Pedro Santos Rodrigues

Bachelor in Computer Science

Accelerating SQL with Complex Visual Querying

Dissertation submitted in partial fulfillment
of the requirements for the degree of

Master of Science in
Computer Science and Informatics Engineering

Adviser:	Teresa Romão, Associate Professor, NOVA University of Lisbon
Co-advisers:	Rui Nóbrega, Assistant Professor, NOVA University of Lisbon
	Tiago Simões, Principal Product Designer, OutSystems

Examination Committee

Chair:	Henrique Domingos, Associate Professor, NOVA University of Lisbon
Rapporteur:	António Coelho, Associate Professor with Habilitation, FEUP
Member:	Teresa Romão, Associate Professor, NOVA University of Lisbon

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

January, 2021

Accelerating SQL with Complex Visual Querying

This document was created using the (pdf)LaTeX processor, based on the "novathesis" template[1], developed at the Dep. Informática of FCT-NOVA [2].
[1] https://github.com/joaomlourenco/novathesis [2] http://www.di.fct.unl.pt

To my dear grandparents Júlio and Fátima.

Acknowledgements

I would like to thank *Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa* and in special to the *Departamento de Informática* for providing me an excelent knowledge base and for setting me challenges that would make me overcome myself, day after day.

Secondly, I have to thank my advisers Tiago Simões, Teresa Romão, and Rui Nóbrega for mentoring me throughout this dissertation. Your support was essential and I learned a lot from you!

Thank you to OutSystems for taking initiative on this interesting research project. And also to all the people I met in the office and in the online meetings, due to this new normal. Everyone was ready to help and the environment was amazing. Also, thank you to all participants of the usability tests.

Furthermore, a huge thank to my friend and partner of this journey Pedro Deodato. We met each other in the first year of university and then we have been walking this journey as a team - P&PCl forever. Thank you for sharing with me the greatest achievements and all the moments we had to overcome during these five years. We will never forget the sleepless nights working on our projects, but the most important is that we did it!

Last but not least, I would also like to express that I'm thankful for all the support my family has given to me. In particular, I would like to thank, my parents, and my siblings. They always encouraged me and supported me whenever I needed it. Without your support, it would have been impossible to get this far. Also, a big thank you to my grandparents, you are an inspiration to me.

Finally, a special thanks to Mariana, who always has been here for everything. This accomplishment would not have been possible without you.

Your work is going to fill a large part of your life, and the only way to be truly satisfied is to do what you believe is great work.

Steve Jobs

Abstract

This dissertation addresses the usability improvement of a graphical user interface that allows query formulation without using textual query languages, such as SQL. This visual tool, called Aggregates, is provided on the OutSystems Low-Code Development Platform, to formulate data queries, through interaction and manipulation of visual components.

Since Aggregates do not support all the existing functionalities of SQL, the OutSystems Platform allows users to build queries using this textual query language. Nonetheless, by evaluating customers' SQL queries, it was revealed that a considerable subset of the queries written in SQL could have been formulated using the visual tool.

The users' interviews and the results of the SQL queries evaluation have foreseen that the cause of the reduced acceptance of the visual approach, could be the existing usability problems on the interface. Furthermore, the interface is inadequate to build more complex queries, which involve more entities and conditions.

Through an iterative design process, this dissertation includes the design, implementation, and evaluation of prototypes with different fidelity levels. The aim is to optimize the effectiveness and efficiency of the process where users communicate to the system what data they intend to extract from the database. Moreover, the readability and comprehension improvement of the query visual representation is intended, reducing the time and the effort required to understand what data will be gathering from the database. The final implemented interface is currently incorporated on the OutSystems Platform to accelerate the query formulation process without harming the learnability of the system.

Keywords: Visual Query Interfaces, Low-Code Development, User-Centered Design, Human-computer Interaction, Iterative Design, Database Querying

Resumo

Esta dissertação apresenta um estudo sobre o melhoramento da usabilidade de uma interface gráfica que permite consultar dados sem recorrer a linguagens de consulta textuais, tais como o *SQL*. A ferramenta visual abordada, denominada *Aggregates*, está inserida na Plataforma de Desenvolvimento *Low-Code OutSystems*, de modo a permitir a formulação de consultas a bases de dados, através da interação e manipulação de componentes visuais.

Tendo em conta que a interface gráfica disponibilizada não suporta todos os tipos de consultas suportadas pelo *SQL*, os utilizadores podem recorrer a esta linguagem textual para construir as suas pesquisas. No entanto, ao avaliar estas consultas criadas textualmente em *SQL*, por clientes da plataforma, percebeu-se que um conjunto considerável de consultas foram construídas usando *SQL*, embora pudessem ter sido construídas usando a ferramenta visual disponibilizada.

Tanto as entrevistas aos utilizadores, como a análise das consultas construídas usando SQL, indicaram que a falta de aceitação do método visual de construção de consultas era causada por problemas de usabilidade na interface. Para além disso, quando as consultas de dados envolvem mais entidades ou condições, os utilizadores sentem dificuldade a usar a interface.

Através de um processo de desenho iterativo, esta dissertação apresenta o desenho, implementação e avaliação de protótipos com diferentes níveis de fidelidade. Foi optimizada a eficácia e a eficiência do processo de utilização da interface para consultar dados. Além disso, também se melhorou a legibilidade da representação visual da consulta, de modo a diminuir o tempo e esforço necessário para compreender que dados pretendem ser extraídos da base de dados. A implementação final da interface encontra-se atualmente incorporada na Plataforma OutSystems, acelerando o processo de criação de consultas de dados sem dificultar a aprendizagem necessária para utilizar o sistema.

Palavras-chave: Interfaces Gráficas de Consulta de Dados, Desenvolvimento *Low-Code*, Desenho Centrado no Utilizador, Interação Pessoa-Máquina, Desenho Iterativo, Consulta de Bases de Dados

Contents

List of Figures

List of Tables

Acronyms

ANSI	American National Standards Institute
DBMS	Database Management System
DQL	Data Query Languages
HCI	Human-computer Interaction
IDE	Integrated Development Environment
ISO	International Organization for Standardization
IT	Information Technology
RDBMS	Relational Database Management System
SQL	Structured Query Language
UX	User Experience
VQI	Visual Query Interface
VQL	Visual Query Language
VQS	Visual Query System

Chapter 1

Introduction

Nowadays, database queries are required not only in computer systems areas but also in most sectors of professional or personal environments. The increase of the database information gathering needs increased the searching for advanced technologies to optimize the time spent and reduce the errors of this data querying process since the most important is to obtain the intended information.

In the decades of the 1960s, Database Management Systems (DBMSs) arose, and later in 1970s new management systems that use relational models, designated as Relational Database Management Systems (RDBMSs), appeared as well as the first Data Query Languagess (DQLs), such as Structured Query Language (SQL) [5] which was considered by the American National Standards Institute (ANSI) and International Organization for Standardization (ISO) as the standard query language [10]. Even though these new technologies provided a structured way to access databases, knowledge of relational logic and DQL was mandatory to fetch data from relational databases. Thereby, only a subset of people could use these powerful querying technologies.

Visual Query Systems (VQSs) were defined by Catarci *et. al.* [3] as "systems for querying databases that use a visual representation to depict the domain of interest and express related requests". These systems use different visual representations and interaction strategies to build database queries. This visual approach could improve the user's learning curve and reduce the mandatory previous knowledge of a DQL, which are more difficult to learn, mainly for people without programming base knowledge. Furthermore, some visual interface mechanisms such as automatisms, accelerators, or feedback messages, can be explored in order to accelerate the querying formulation process and reduce the errors that users can make while they are building queries.

In that way, those visual systems are not only useful to users not familiarized with DQLs. Conversely, some studies have revealed that visual languages might be convenient to the expert users too. For instance, the comparison made by Catarci and Santucci [4] concludes that diagrammatic languages can reduce the error rate of the queries, in comparison with the ones that were build using textual languages, even when they were formulated by expert users. These results have demonstrated that even expert users make mistakes in simple queries (e.g., they may not remember the name of the tables or the precise syntax of some language expressions). Therefore, the Visual Query Interfaces (VQIs) should be built strategically in order to take advantage of their peculiarities that could optimize the querying process not only to the users with a low database querying experience level but also for highly experienced users.

1.1 Motivation

Low-Code Development is a recent development paradigm that seeks to reduce the time and effort spent on tasks that would not have a significant impact on the final product outcome. As high-level languages, APIs and third-party infrastructures have allowed developers to be more productive and focus on the most valued sections of the software they produce. Low-code approaches have followed this endeavor, using visual Integrated Development Environments (IDEs), connectors between components and lifecycle managers to employ an abstraction layer on the high-level languages, removing concerns of infrastructure or pattern reimplementation. In that way, developers could focus on tasks that truly accelerate the growth of the end product, achieving the desired goals with greater efficiency [70].

The OutSystems Platform provides a cloud solution of low-code development, which allows developers to build and deploy enterprise-grade applications though visual interactions optimizing the time, effort, and previous knowledge necessary. Then, the OutSystems Platform aims to provide an application development environment that could be used by users with different backgrounds to build, deploy and manage their applications, using good practices and state-of-the-art technologies, even if they do not need to concern about that. Therefore, the vision and potential of the OutSystems Platform are similar to the VQIs under-mentioned above since both intend to facilitate, accelerate, and optimize processes through visual interaction.

In spite of development in OutSystems is based principally on visual languages, there is the possibility to use low-level code, written in textual languages, such as Java, .NET or SQL, in order to increase the extensibility and the power of solutions. In that way, there are alternatives to performing operations not supported by the low-code visual approaches. This is the difference between Low-Code and No-Code paradigms since in No-Code is not possible to use low-level code [72]. Nonetheless, if the visual languages of low-code platforms are more robust, responding more thoroughly to users' requirements, there is a diminished demand to resort to these textual programming languages which are high error-prone and have a worse learning curve, requiring also, on multiple situations, previous coding experience.

Furthermore, as mentioned by Amaral *et. al.* [12], web and mobile applications produced on OutSystems' technology, have proven an increase in quality. Also, it was concluded that low-code developers are 10.9 times more productive than the standard of Information Technology (IT) Industry, which does not use these rapid software solutions [21]. These results reinforce the importance of the improvement of the visual languages used on these platforms.

Following that vision, the principal motivation is the existence of a platform component that accelerates the query building process and reduces the errors that could occur throughout, keeping the experience simple and understandable by all users.

1.2 Problem Description

The OutSystems Platform [63] provides a Visual Query Language (VQL) that allows users to query data from databases through visual interactions. Using that interface, it is possible to perform some operations that are usually supported by textual DQLs, namely join, filter, sort, and aggregation operations.

Although the existing interface turns the process of query formulation more simple and intuitive, it does not support all SQL functionalities. Due to that lack of expressiveness, the platform allows its users to formulate queries using SQL. However, as *Catarci et al.* referred [4], visual languages could give advantages in query formulation for all users, including the ones that are proficient in SQL. Thus, it is important to provide a powerful and consistent interface in order to give users the possibility to accelerate the formulation process, reducing also the rate of errors that may arise.

The principal purpose of this visual interface is to provide a more visual and dynamic tool that accelerates the query formulation process and turns it easier and less error-prone. So, the existing interface should be a useful and efficient tool for developers due to the potential of that visual approaches above-mentioned. However, OutSystems knew there were developers that were not using the visual querying interface, maintaining their preference for SQL. Therefore, it was necessary to verify the main causes that lead users to not use that visual tool.

At the beginning of this dissertation, the lack of functionalities (e.g., IN, NOT IN, EXISTS, NOT EXISTS, DISTINCT, UNION and the possibility to use subqueries) was indicated as a significant factor for users to use SQL. Nevertheless, the first interface explorations revealed impactful usability problems. That is an important point since it could considerably harm users' query formulation process, reducing the value proposition of the system which intends to accelerate the querying process, keeping it more effective and less error-prone. Moreover, there were metric results and user interviews that confirmed the presumption concluded after interface exploration, highlighting the user experience of the interface as the core subject to research and improve.

Beyond the motivation of turning the query building process faster, effective, and less error-prone, the following questions would guide the problem definition process in order to clearly understand the existing problems of the interface:

- Why do OutSystems developers often use SQL to formulate database queries?
- What are the main causes that users point out to use SQL?
- Who are the users more unsatisfied with the current provided visual approach to retrieve data? What are their reasons?

Under the above-mentioned circumstances, the goal of this thesis is the design, implementation. and evaluation of a new and more powerful VQI to provide an improved User Experience (UX) that:

- Accelerates the query formulation process;
- Improves the readability of queries (i.e., turns easier to understand what data will be fetched from database);
- Maintains the interface simple and intuitive for all users even the ones that do not know SQL or OutSystems, reducing also the existing learnability curve, whenever possible.

1.3 Research Questions

The main research question that is being addressed in this dissertation is:

Can we enable OutSystems developers to easily do complex
database queries without ever using SQL?

Regarding the main question and the diverse background of the system target users, it is important to research how can be developed a solution that covers the requirements of all user types with the improved usability possible.

The following research questions focus on this usability trade-off which depends on the users and the system particularities:

Research Question 1: Does the existing interface have usability problems for the less experienced users?

Considering the VQI already implemented, the most complex queries have not been correctly covered by the tool. However, it is important to analyze also if novice users had similar problems or others that could have an impact on the task performing.

Research Question 2: Can experienced users take advantage in using the visual interface to build queries instead of SQL?

Since the users more experienced who know other textual query languages, such as SQL, can use SQL to perform the queries, are advantages for them in the usage of a VQI? What are the advantages and disadvantages of this type of approach for these users?

Research Question 3: Can expert users' UX be improved without reducing the system's learnability and satisfaction for less experienced users?

The usability attributes trade-off depends on the system's target users' expectations and requirements. Thus it is important to take into account if the development to improve the efficiency, effectiveness, and satisfaction of expert users does not harm the effectiveness and the learnability of the operations performed for the less experienced users.

1.4 Contributions

As a result of the work developed throughout this dissertation, these are the main contributions:

- A detailed study of the usability problems of the interface based on different research methods: self-exploration and analysis of the existing interface, user interviews, quantitative data analysis, ideas posted on the worldwide OutSystems developers' community, and usability tests of the existing interface;

- Design and implementation of a new visual querying interface built using an iterative design process with two iterations: a paper prototype and a final prototype completely integrated in the OutSystems Platform;

- Evaluation of the new interface through several usability tests with specific user testing scenarios and compared the effectiveness, efficiency, satisfaction users have using the new interface against they had using the previous one;

- State-of-the-art study of the major problems associated to the query formulation tasks and the visual query interface approaches used to formulate queries visually without textual query languages.

Furthermore, this dissertation has triggered the beginning of a new initiative at OutSystems to integrate the idea and prototype developed in this dissertation in the final product in order to be released in a further version of the OutSystems Platform.

1.5 Document Structure

The remaining chapters of this thesis are organized as follows:

- Chapter 2 - Background: introduces some design and usability concepts to be used in this work. Besides, it is provided a context of the OutSystems Platform current progress, which explains the functionalities of the existing data querying tool;
- Chapter 3 - Related Work: analyses the users' interaction with database systems to improve the interfaces' suitability to the target users of the system. Also, approaches used by other systems to create interfaces that allow to visually build queries, are described and compared, detailing the interaction strategies used.
- Chapter 4 - Methodologies: presents the methodology planned to design, implement and evaluate the solution built in the context of this dissertation, including the problem exploration process, the user analysis, the iterative design, the testing scenarios and the evaluation approach adopted;
- Chapter 5 - Requirements and Analysis: summarizes the processes used to deeply understand the problem and the difficulties users of the system have in order to provide also in this chapter a collection of the major usability problems of the interface;
- Chapter 6 - Design and Implementation: describes the entirely solution conception process from the first sketches to the implementation of the final prototype. For each prototype it is described the design, implementation and evaluation phase.
- Chapter 7 - Conclusions and Future Work: includes the final remarks of the results obtained through this dissertations and the aspects that should be approached in the future.

Chapter 2

Background

This thesis aims to improve the interface that allows users to build queries in a more efficient and effective way. Therefore, Human-computer Interaction (HCI) is a core subject of this work since such interface can only be improved if its interaction and usability are studied from the user's perspective.

Accordingly, this chapter will introduce key concepts of HCI, as well as, a brief contextualization of the OutSystems Platform that is indispensable for the comprehension and progression of this study.

2.1 Human-Computer Interaction

Although computer systems have been designed by humans, these two parts of HCI do not speak the same language. Nonetheless, these types of systems were created to support, in a transparent way, human tasks and requirements, forgiving careless mistakes [9]. Thus, HCI aims to study the relationship of users and computer systems, in the context of the users' desired tasks, in order to "unfold and reveal challenges and insights, and to instrument appropriate solutions for alleviating the current obstacles to the access and use of advanced information technologies" [27].

2.1.1 Main Concepts

The **Usability** of a system is one of the most important concepts in HCI, that can not be forgotten on the design process, since its attributes must be taken into account performing also a guidance function through all this process. This concept was standardized in ISO-9241 [49] as "extent to which a system, product or service can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use".

However, usability is not a single-dimensional property, being always associated with its attributes, that characterize the user accessibility when is using the system into five different points, such as referred by Nielsen [18]:

- **Learnability**: How easy is the learning process until a novice user (who has not used the system before) achieves a high-level of proficiency using the system [29]. The learnability is higher as the learning process is faster, and the user has to spend less effort to reach his goal. Also, it depends on the tutorials and training provided to users. A system that requires less training has higher learnability than a system that requires more training. The time that a novice user requires to perform some specific tasks can be used to measure learnability. Learnability can be improved using tips while a novice user explores the system doing his first tasks.

- **Efficiency**: Refers to the productivity level of a user who has already learned how to use the system. Efficiency can be measured analyzing the time that expert users spent to do specific tasks on the system. This attribute can be improved, for example, adding shortcuts to accelerate the interaction process.

- **Memorability**: Defines how easy is for a user, that was using a system before but did not use it for a time period, to do his desired tasks on the system. So it's related to how many times the user has not used the system and the time that the user needs to remember how the system works. Therefore, if a system has good memorability the user does not need much time to remember it, even if it has stopped using it for a long period of time. Memorability can be measured, for example, analyzing the interaction process of a user who has been away from the system, while he uses the system again. The use of visual components and metaphors with real-life objects helps, sometimes, the users in this process.

- **Errors**: A system not only must have a low error-rate but if an error occurs, the user should be able to recover from that. Since there are multiple types of errors with different severity levels, catastrophic errors should not occur. This attribute can be measured by the evaluation of the error-rate, taking into account the severity levels of the errors. Furthermore, if a system has errors, that can be reverted and does not have a negative impact on the final result, cannot be forgotten that these errors also harm the efficiency of the system.

- **Satisfaction**: The most subjective attribute of the usability that is related to the overall satisfaction of the user when uses the system. It could be measured by asking the users about the experience while they are using the system, always searching for subjective answers.

Nonetheless, as mentioned by Nielsen [18]: **"it is not always possible to achieve optimal scores for all usability attributes simultaneously"**. Thus, when a system is designed it is necessary to prioritize what are the most important attributes for the users and the domain where the system will be used and applied. These trade-offs are one of the most challenging tasks of the design processes because it depends on user expectations and their backgrounds, as well as, the problem domain and what are the main focus of the system use. Accordingly, it is fundamental that the design process can focus on target users of the systems. Therefore, the main concepts, processes, and techniques for a design process centered on the users will be described.

2.1.2 User-centered Design

Before user-centered design principles and methodologies were adopted, the Waterfall model was commonly adopted as a software development process. This model comprises five sequential phases, from the requirements specification phase to the operation and maintenance phase, and has a good quality control since documentation and planning are a major concern of this methodology [2]. However, the stages of this model are not overlapping stages, so other development methodologies and philosophies arose to mitigate this problem and include the user on the design process, due to their impact on the usability of the system.

Consequently, it was necessary a new model that has the users included in the development process to verify, along with the development, if the approaches adopted are positive and what is the users' acceptability. Nielsen [17] reinforces this saying that "user interfaces should be designed iteratively in almost all cases because it is virtually impossible to design a user interface that has no usability problems from the start. Even the best usability experts cannot design perfect user interfaces in a single attempt, so a usability engineering lifecycle should be built around the concept of iteration".

The Spiral model of iterative design arose as an iteration through design, implementation and evaluation phases where the cost and accuracy increase on each iteration. The first iterations should use low-cost resources, like paper prototypes, and when the results are positive the accuracy should be incremented, changing to high-fidelity prototypes, such as computer prototypes [13].

2.1.2.1 User and Task Analysis

Regarding the concept of usability presented above and the importance of the users on the design process, it is important to define the users and their desired tasks of the system in order to find the best solution as possible to the usability attributes trade-offs. Just a good description of the users and the tasks of the system leads the designers to the best choice of what are the usability attributes most important for the system.

Accordingly, it is important to make a **User Analysis** to understand all users' characteristics that could have an impact on the acceptability of the system. The expected result of this analysis should be a set of structured information that characterize the users of the system in terms of technological expertise, knowledge of the business domain, application experience, educational background, gender, and age, as other aspects that might be useful to comprehend, depending on the system's users and domain [8]. The more traditional process to gather this information is through questionnaires or interviews, but that can also be obtained by conducting market analyses or observational studies [18].

Furthermore, it is essential to enumerate and analyze the tasks the users should perform using the system. The **Task Analysis** process aims to aggregate information about the tasks that should be performed on the system, starting from the system's overall goals and break down these to obtain individual tasks [18]. Moreover, the goal of this analysing process is to obtain more structured information about: how the tasks are performed using the existing systems, what are the pre-conditions and the requirements of each task, why the users need to perform this tasks, and others that might be useful to characterize the tasks of the system.

The techniques used, to extract information to this analysis, aims at figuring out how the tasks should be done instead of how they would perform them. The idea is to resort to examples, as well as possible, in order to understand what type of strategies are used, what type of exceptions from their normal workflow is occurring, and other aspects that can be observed where the communication with users is on a concrete level [18]. In addition, Nielsen [18] points out that "The users' model of the task should also be identified, since it can be used as a source for metaphors for the user interface", which reinforces that these dialogues with users to obtain analysis content, can be useful also to find relevant solutions to latter design process phases.

Therefore, the outcome of this analysis should contain a list of the entire tasks that users what to perform in the system, the information that is required to complete them, the steps and the dependencies between tasks, all the outputs that must be generated, and how is the communication process between the users associated with the system's tasks [18].

2.1.2.2 Sketching and Prototyping

After the user and task analysis process, designers must start sketching and prototyping ideas and approaches, in order to think about how can they solve the problems. Nevertheless, this phase of the design process should start with sketching techniques, as these are not only a good and inexpensive starting point to communicate ideas, but also help to develop structure and enrich the reasoning, leading to the perception of other details as well as of other approaches to solving the related problems.

While sketching techniques are more plentiful, and have a low detail level, being mainly based on suggesting and exploring, rather than retrieving results, the prototyping phases, have more refinement approaches and are used to test the design choices made.

However, prototypes may have different thoroughness degrees, presenting different advantages and disadvantages. Thus, it is important to start the prototyping process with low fidelity prototypes, as paper prototypes, since the objective of these is to evaluate the conceptual model (if the users understand the system), the functionalities presented, the navigation, the screen components distribution, and the terminology used. After the evaluation of these prototypes presents good results, high fidelity prototypes should be used, such as computer prototypes. There are a set of available tools to assist in the building process, of these prototypes, such as Balsamiq [31] and Mockingbird [56]. These different prototype types can detect different issues when tested with users, so it is very important to test prototypes with different granularity levels.

Furthermore, there is another relevant aspect of the prototype designing, that is the scope definition of the prototype. It is important to define what features the prototype undertakes and what is the inherent detail level. Nielsen [18] describes this as two dimensions of prototyping: horizontal prototyping and vertical prototyping, as demonstrated in Figure 2.1. A vertical prototype is characterized as a prototype to test a restricted part of the system but with real users and circumstances. A horizontal prototype is presented as suitable for test the entire system but in a less realistic approach.

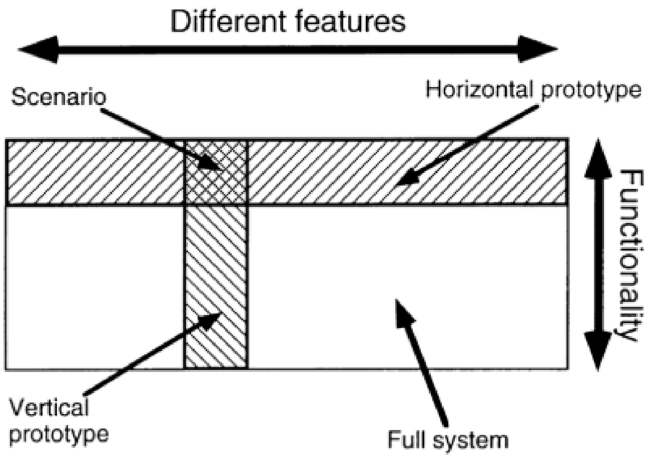


Figure 2.1: The two dimensions of prototyping: Horizontal prototyping keeps the features but eliminates depth of functionality, and vertical prototyping gives full functionality for a few features (source: Nielsen [18])

Finally, regarding the methodology about how to use the prototypes built, Dix et al. [9] refer that are three main approaches:

- **Throw-away**: After the prototype is built and tested, it is used on the final system development, but after that, the prototype is discarded and the rest of the design process continues without relying on the prototype previously built;

- **Incremental**: First, the system is separated into different parts, and each part is built, one at a time. So the prototypes are developed separately, regarding its correspondent part, and finally are combined to build the final system;

- **Evolutionary**: Contrary to the throw-away approach, the prototypes developed are used as the basis for the next iteration of the design.

2.1.2.3 Evaluation Techniques

The evaluation phases are crucial in the design process, since they allow designers to understand the systems' specific problems and the impact of the interfaces on users. So, the expected outcome of these processes is a list of usability issues ordered by priority level, referring to what usability principles and guidelines are not being accomplished and what solutions can be applied to solve the problem [18].

However, there are different approaches to evaluate interfaces. First, one important topic that distinguishes two types of techniques is who performs the evaluation. There are techniques where only the designers and specialists are involved in the process and are another type of evaluation where users participate in [9]. Thus, there are two types of evaluation: evaluation through expert analysis and evaluation through user participation.

Evaluation through expert analysis

In this type of evaluation, designers, or other specialists, evaluate the system, supporting their analysis on cognitive principles, to preview usability problems likely to occur. Moreover, this evaluation not only is cheaper because it does not involve users, as it also can be applied to any phase of the design process, from the design specification to the high fidelity prototypes [9].

Regarding the approach presented above, these two methods are one of the most used:

- **Heuristic Evaluation** [19]: The system is thoroughly analyzed in order to find problems that do not follow importantly and recognized usability heuristics. After a problem has been detected, it should be reported, not only with a description of the problem but also the indication of the heuristics that have not been accomplished, the severity level of the problem and possible solutions to solve it [18].

- **Cognitive Walkthrough** [22]: This method uses a sequence of actions as the principal resource to guide the evaluation process. For each action, the evaluator tries to understand if all steps are clear and visible, as well as, if the system gives clear feedback, confirming if the action has been completed. Usually, the main focus of this method is to analyze the learnability of the system. Mainly, to understand if the system provides a good learning mechanism through exploration, rather than using manuals, training or other types of *a priori* learning processes [9].

A study made by Desurvire *et al.* [7] concluded that Heuristic Evaluation made by specialists is better to predict some problems before the user testing process than Cognitive Walkthrough. The reason pointed out for this result is that heuristic evaluation can often help to remind the designers of problems since this method analyses more dimensions of the system than Cognitive Walkthrough.

Evaluation through user participation

Although there exist methods that do not need the users to evaluate the system usability, it is difficult to predict all the behaviors of the users when they interact with a system. Therefore, there are multiple methods to make usability tests with people that are the system's target. Some methods of user testing which can be resourceful on the context of this work, are the following, respecting the terminology used by Dix *et al.* [9]:

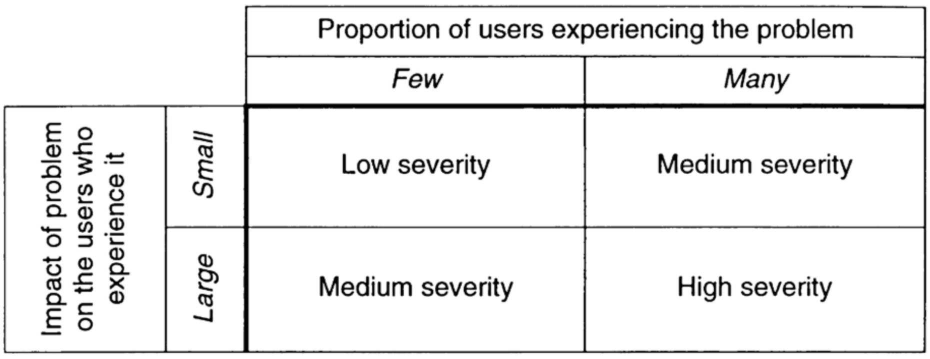
- **Observational Methods**: the main principle of these methods is observing users using the system to conclude important usability information about it. Usually, it is requested to the user to 'thinks aloud' since it might be possible to obtain more insights that can be useful, not only to understand why the user might have made an error, but also it can be a good strategy to find starting points for other possible solutions. Moreover, although usually, these tests have a set of predetermined tasks, since it is easier to find the user reaction to the system part the need to be tested, also can be executed tests only by evaluating the normal tasks of the users on their work;

- **Experimental Methods**: starting from a properly defined test hypothesis, it is selected a set of users to perform an experimental test to verify if the test hypothesis proves to be true or not. Thus, it is necessary to define previously all the experimental environment, which includes: the test hypothesis that wants to be verified, the users that will perform the test, and the independent and dependent variables. The dependent variables are the ones that express the result of the test in function of the independent variables, such as the task execution time or the number of the errors that occurred. The independent variables are chosen by the test designer to produce different conditions for comparison. Examples of independent variables can be the size of an interface component or the use of an interaction technique. This method is very useful to verify through a test hypothesis which of the possible solutions presented (independent variables) have a better performance for the intended application context;

- **Query Methods**: these methods focus on what the users think about the system, collecting information from interviews or questionnaires to analyze their opinion. One advantage of this method is that it might reveal issues not observed previously complained before, but contrary a lot of cases are not tested, since in the interaction field, many times, the user only finds a problem when it occurs for the first time. So it is not possible to extract concrete information from users that never have passed for this situation.

Finally, independently of the evaluation process adopted (through expert analysis or user participation), severity ratings should be attributed to the problems identified in order to define the main priorities and understand which might have a larger impact on the system acceptability. However, although these levels should be attributed by specialists, if they use not only cognitive principles, but also use the results observed from the user testing phase to sustain the classification of the problems detected, the result can be more accurate.

Besides, the Figure 2.2, extracted from [18], displays two influential factors that should be taken into account to attribute a severity level to a problem: how many users are experiencing this problem and what is the impact level on the user.



		Proportion of users experiencing the problem	
		Few	*Many*
Impact of problem on the users who experience it	*Small*	Low severity	Medium severity
	Large	Medium severity	High severity

Figure 2.2: Severity Levels of the Problems based on their impact on the users (source: Nielsen [18])

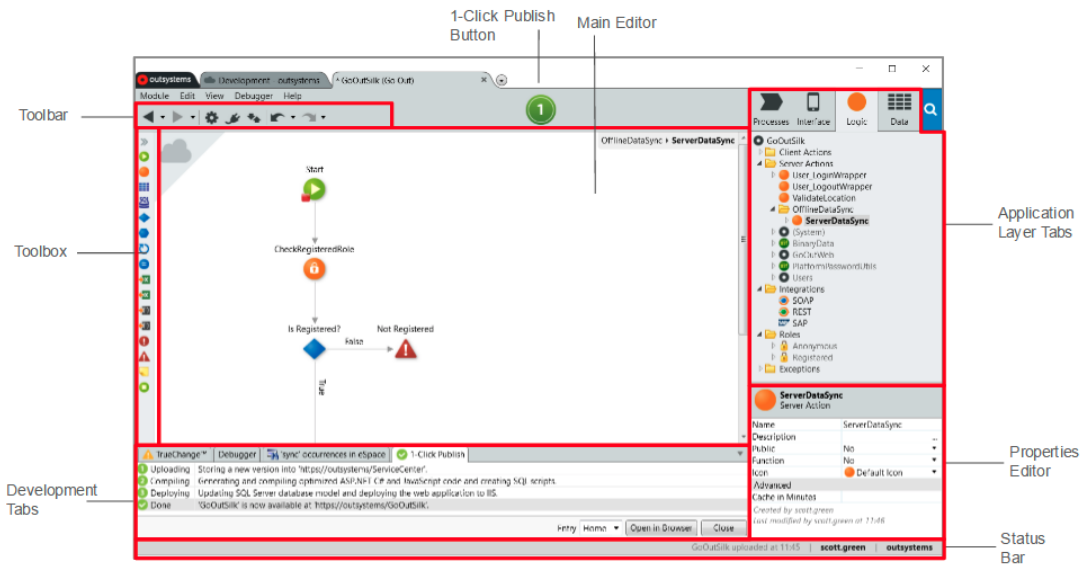
2.1.2.4 Errors Classification

The errors that occurred when a user interacts with a system are excellent indicators for designers because the understanding of the reason that led to error situations is a good strategy to classify them. Therefore, errors can be classified as slips and mistakes, as will be presented below according to Dix *et al.* [9]:

- **Slips**: in these types of errors, the user knows how to do the intended task on the system, however, he presses a wrong button or closes one window accidentally. So, he understands the action, but a misaction does not allow that he reaches his goal;

- **Mistakes**: these errors occur when the user does not understand the system, formulating a wrong goal. An example of a mistake is when the user does not understand the action associated with an icon, performing a not intended action.

Therefore, the strategy to mitigate the problems associated with these two types of errors could be different, as mentioned by Dix *et. al.* [9]: “Slips may be corrected by, for instance, better screen design, perhaps putting more space between buttons. However, mistakes need users to have a better understanding of the systems, so it will require far more radical redesign or improved training, perhaps a totally different metaphor for use.”



```
Uploading   Storing a new version into 'https://outsystems/ServiceCenter'.
Compiling   Generating and compiling optimized ASP.NET C# and JavaScript code and creating SQL scripts.
Deploying   Updating SQL Server database model and deploying the web application to IIS.
Done        'GoOutSilk' is now available at 'https://outsystems/GoOutSilk'.
```

Figure 2.3: Main areas of Service Studio (source: OutSystems[65])

2.2 OutSystems Background

The OutSystems Platform has the mission of simplifying and accelerating the development and management of digital enterprise solutions, no matter the dimension and domain of the applications. It covers the entire development lifecycle which aims to promote rapid development and integration, to facilitate and speed up the deployment stages, to keep track of the status and health of the applications produced, and to expedite the management of daily operations and configurations on the final products [62].

2.2.1 Visual Development Environment

Service Studio is the low-code development environment of the platform, which allows the developers to create complete applications using visual elements to perform drag and drop actions. Figure 2.3 presents an overview of the IDE, which highlights the different areas of the workspace.

Using the widgets and icons provided in the toolbox, the main area is dedicated to designing the applications' interface and logic. So, in the main area, there are visual elements, which can be set using the properties editor, placed on the bottom right corner of the screen.

Also, it includes other sections whose main purpose is not the product development, but are related to key actions of the software development process. Therefore, on the window's top, there is a toolbar which has shortcuts to some of the most common operations, and a green circle button, denominated "1-Click Publish button", to start the automated deployment process provided. Besides, the bottom of the window is dedicated not only to the presentation of messages, errors, and warnings but also to debugging the application.

Since the development environment can be used to develop a complete full-stack application, the elements, which can be manipulated in the Service Studio, can be related to different parts of the application. The Application Layer Tabs, which contain a tree view of its elements, are described next:

In the **Processes** tab is possible to create and manage business processes of the systems through a flow that can be composed by human or automatic activities, time waits, conditional decisions and indications to execute processes. Also, this section can be used to configure the timers of the application. Then, it is possible to indicate when a timer should start, what is it period and what action should be performed when it is triggered.

The **Interface** tab can be used to manage the components related to the visual interface of the final application. It is possible to observe all applications' screens, as well as, the variables and the actions related to each one. Moreover, flows between the various screens can be also defined. If one screen or action is selected, it will be possible to add new visual components to the interface or assign new elements to the action flow in order to define all the client-side logic of the screen.

The **Logic** tab is where the core logic of the application could be defined. It includes not only the server actions of the application but also the exceptions specification, the existing user's roles and also the integration with external services. Although there is a data section on the application layer tabs, which will be described below, the actions which require data querying are managed in this section.

In the **Data** section is covered the database modeling, making possible the creation of diagrams to represent the schema of the database. Moreover, in the tree view of this tab is allowed to establish new entities and static entities in order to define the data model of the application.

2.2.2 Visual Data Querying

The main topic of this work is the improvement of the visual data querying process on the low-code development of applications, using OutSystems. Consequently, the headway of visual querying components of the platform is an essential factor to properly understand the dissertation. Regarding the last version of the OutSystems platform, the actual visual data querying tool, which is the starting point of this study, will be described.

2.2.2.1 Previous Work

Since 2002, which is the release date of the first OutSystems low-code development environment, the Hub Edition 1.0, allows two manners to create database queries [71]:

- **Simple Queries**: visual query builder that allows the creation of some less complex queries, interacting with a graphical user interface;

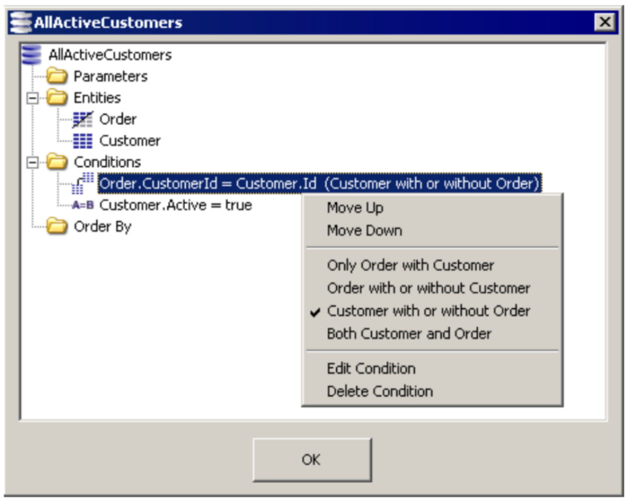


Figure 2.4: Simple Query example in Hub Edition 2.0 (source: OutSystems [66])

- **Advanced Queries**: feature to specify queries textually, using a language based on SQL, but includes some extra syntax to reference variables used of the application development;

Then, since the first versions of the IDE, it is provided two ways to build queries. The first uses the visual language, which does not need so many learning requirements but also diminishes the necessity to remember the entities' name or the language syntax. The second provided relies on SQL, which is the standardized textual query language, known for the developers' majority.

The first simple queries versions have accelerated the query building process finding automatically the relationships between the entities chosen. The main idea is that the developer only needs to select the entities intended and the respective join conditions would appear automatically in the query view interface. After this, it will be possible to change the join type, as well as, to add, edit and remove filtering or sorting conditions. Figure 2.4 presents a simple query example in Hub Edition 2.0 (2003) when the developer was changing the join type to an Outer Join.

This visual querying approach continued in the next versions, adding some minor improvements, such as the support to structures in order to store temporary information without changing the entity definition [67], and the inclusion of a properties pane to view and change the properties of all query elements in a single window [68].

However, at the launch of the OutSystems Platform 9 (2013), an entirely new way to manipulate data and express database queries has been released. These new components of the system, called **Aggregates**, have replaced Simple Queries, promoting a new interaction strategy to query databases, where the main focus is the data, instead of query design. Also, new features have been introduced to improve the expressiveness of the VQL, namely grouping functions and the ability to add calculated columns easier.

2.2.2.2 Current Progress

Since the implementation of Aggregates, the query process without textual languages is more visual and more focused on the query outcome. Aggregates can be used to query data from the server or mobile local storage, and can be created in the following ways:

- If someone is designing the user interface and wants to present some data gathered from the database (server or local storage), he can right-click on the screen where data will be displayed and select “Fetch Data from Database”;
- When an action flow is designed, there is an Aggregate icon in the toolbox that can be dragged and dropped to the main area;

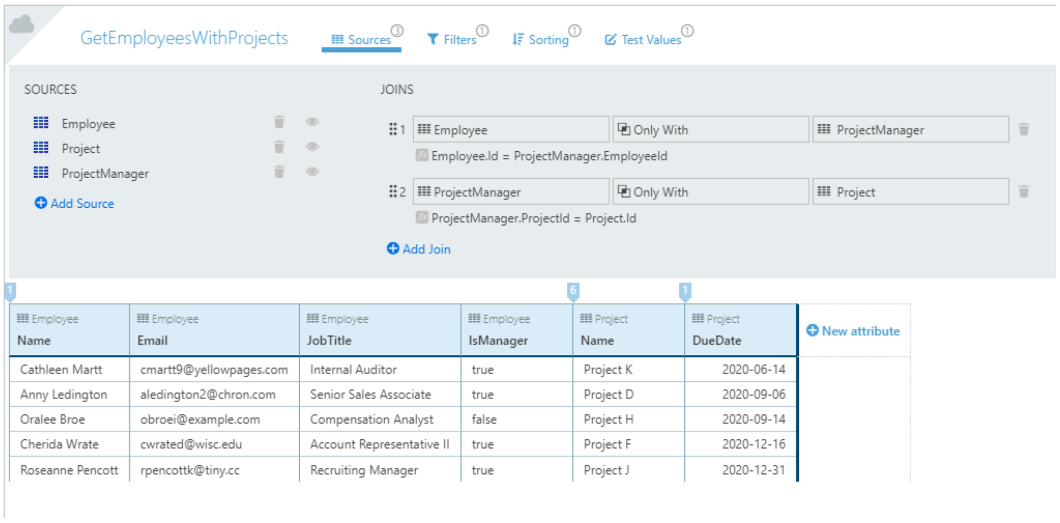
When the Aggregate is created, the first step is the data source selection in order to specify which entities should be included in the query. Then, the developer should click in the main area and choose the entities he wants or drag and drop the entities to add them to the query. When an entity is added, all its attributes are automatically included in Aggregate.

Since an Aggregate can include one or more entities, added on the beginning or later, if it has more than one, it will be analyzed to verify if there are relationships between them. If the entities are related, according do the database model defined, then the system uses the existing relationship to automatically formulate the join between them. Basically, if the relationship between the two entities is mandatory, (i.e., the foreign key that is an identifier of the other entity is mandatory) then an inner join is added automatically, otherwise a left join is added. For the cases there is no relationship which links the two entities added, then the Aggregate will allow users to specify the join condition manually.

Hereupon, the Aggregate has already been created and its data source specified, so the visual querying process can be started, in a progressive way, seeing at the same time the query output. Figure 2.5 demonstrates an Aggregate on the referred state, to be possible to understand the structure of the interface.

First, this component of the OutSystems Platform presents two main capabilities, represented visually in two distinct areas: the query design area, and the viewer of the query result. The former, located at the top of the window, is composed of a set of four tabs, where each selection action changes the grey area to the respective form-based interface. The latter is a table-based interface, which is similar to spreadsheets applications, such as Microsoft Excel [51] of Google Sheets [47] and its principal aim is to provide a direct and visual approach to show the query output.

Focusing on the query design area, the sources tab, illustrated in Figure 2.6, can be used to add, change or remove the entities of the quey, defining also, the join types between them. There are three join types available: "only with", "with or without", and "with". The respective joins in SQL are: "inner join", "left join", and "full outer join". Additionally, each one appears with a visual representation similar to Venn Diagrams



GetEmployeesWithProjects — Sources 3 | Filters 0 | Sorting 0 | Test Values 0

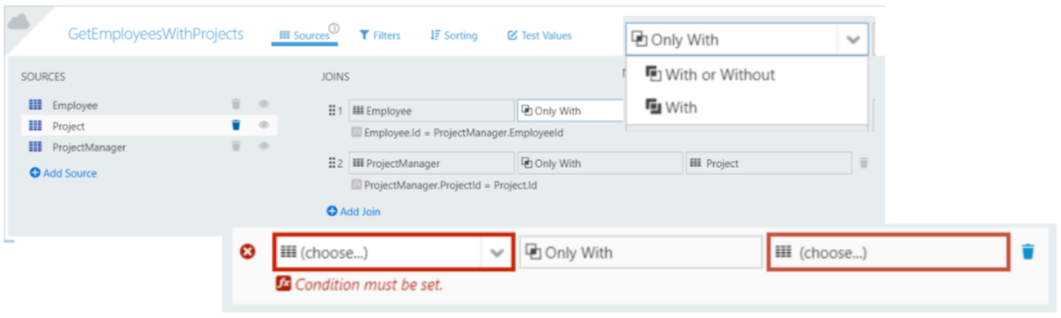
SOURCES
- Employee
- Project
- ProjectManager
- Add Source

JOINS
- 1 Employee | Only With | ProjectManager — Employee.Id = ProjectManager.EmployeeId
- 2 ProjectManager | Only With | Project — ProjectManager.ProjectId = Project.Id
- Add Join

Employee Name	Employee Email	Employee JobTitle	Employee IsManager	Project Name	Project DueDate
Cathleen Martt	cmartt9@yellowpages.com	Internal Auditor	true	Project K	2020-06-14
Anny Ledington	aledington2@chron.com	Senior Sales Associate	true	Project D	2020-09-06
Oralee Broe	obroei@example.com	Compensation Analyst	false	Project H	2020-09-14
Cherida Wrate	cwrated@wisc.edu	Account Representative II	true	Project F	2020-12-16
Roseanne Pencott	rpencottk@tiny.cc	Recruiting Manager	true	Project J	2020-12-31

New attribute

Figure 2.5: Aggregate Example



GetEmployeesWithProjects — Sources | Filters | Sorting | Test Values

SOURCES
- Employee
- Project
- ProjectManager
- Add Source

JOINS
- 1 Employee | Only With — Employee.Id = ProjectManager.EmployeeId
- 2 ProjectManager | Only With | Project — ProjectManager.ProjectId = Project.Id
- Add Join

Only With
- With or Without
- With

(choose...) | Only With | (choose...)

Condition must be set.

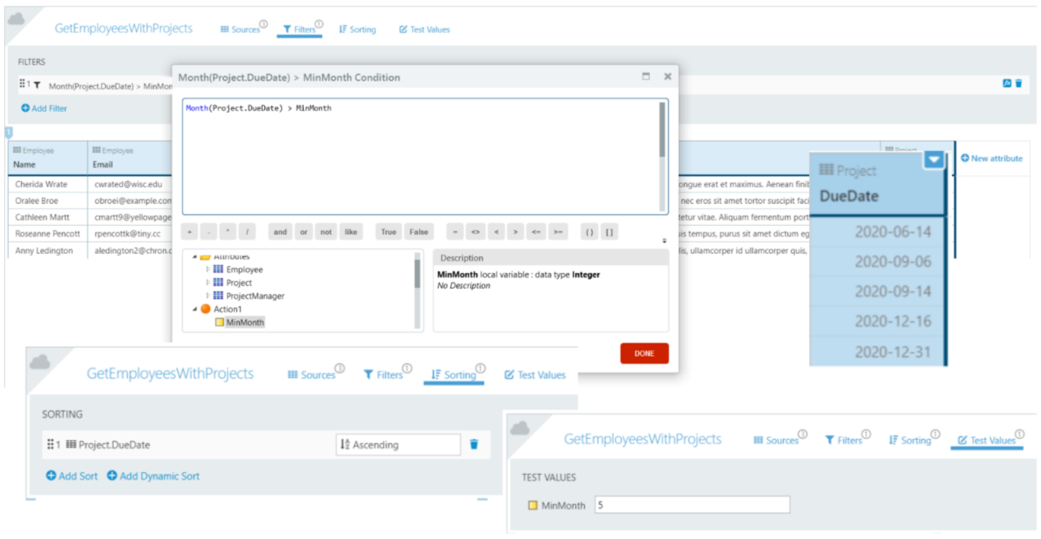
Figure 2.6: Aggregate - Defining Sources

[16] to be easier to identify which data is selected on each join type. Moreover, it is possible to edit the join condition textually, using the platform expression editor.

The filtering tab can be used to apply filters in the query likewise the WHERE statements in SQL. These filters can be defined through boolean conditions inserted in the expression editor. The conditions are specified textually with the assistance of some auto-completes and shortcuts available in a tree view.

The sort conditions can be added in the sorting tab choosing the "add sort"or the "add dynamic sort"options available. The difference between them is that dynamic sort relies on a variable of the system, contrary to the other that depends only on an entity attribute. To add a sort, the user has to select what entity wants to sort and specify what are the sort criteria, for example, ascending or descending. Moreover, more than one sort can be inserted and they can be ordered to establish the priorities between them.

Finally, the last tab has a different behavior when compared with the rest of the options available in this interface area. The main goal of this feature is the testing of query output when concrete values are assigned to the variables referred to in the query. Thus, it does not contribute to the query design in the same way as the other tabs, presenting only a

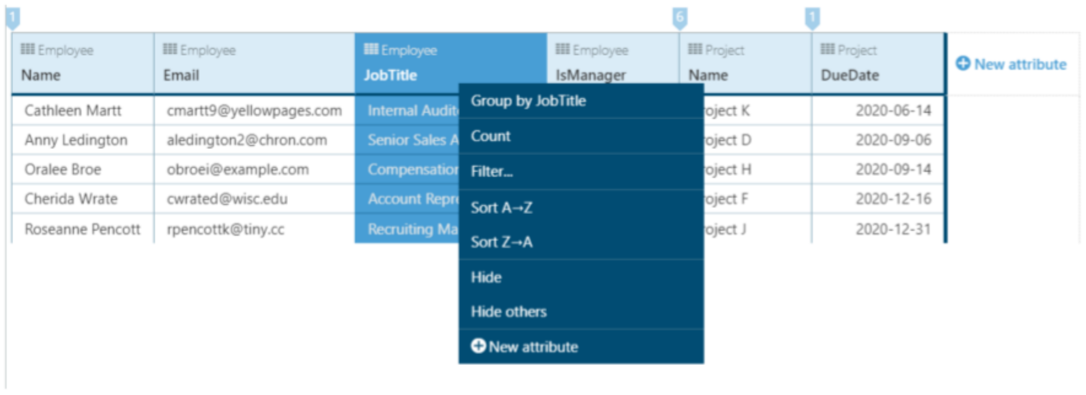


```
Month(Project.DueDate) > MinMonth
```

Project DueDate
2020-06-14
2020-09-06
2020-09-14
2020-12-16
2020-12-31

MinMonth
5

Figure 2.7: Aggregates - Filtering, Sorting and Test Values in an example of querying DueDates after a month indicated in a variable



Employee Name	Employee Email	Employee JobTitle	Employee IsManager	Project Name	Project DueDate
Cathleen Martt	cmartt9@yellowpages.com	Internal Audit		oject K	2020-06-14
Anny Ledington	aledington2@chron.com	Senior Sales A		oject D	2020-09-06
Oralee Broe	obroei@example.com	Compensation		oject H	2020-09-14
Cherida Wrate	cwrated@wisc.edu	Account Repr		oject F	2020-12-16
Roseanne Pencott	rpencottk@tiny.cc	Recruiting Ma		oject J	2020-12-31

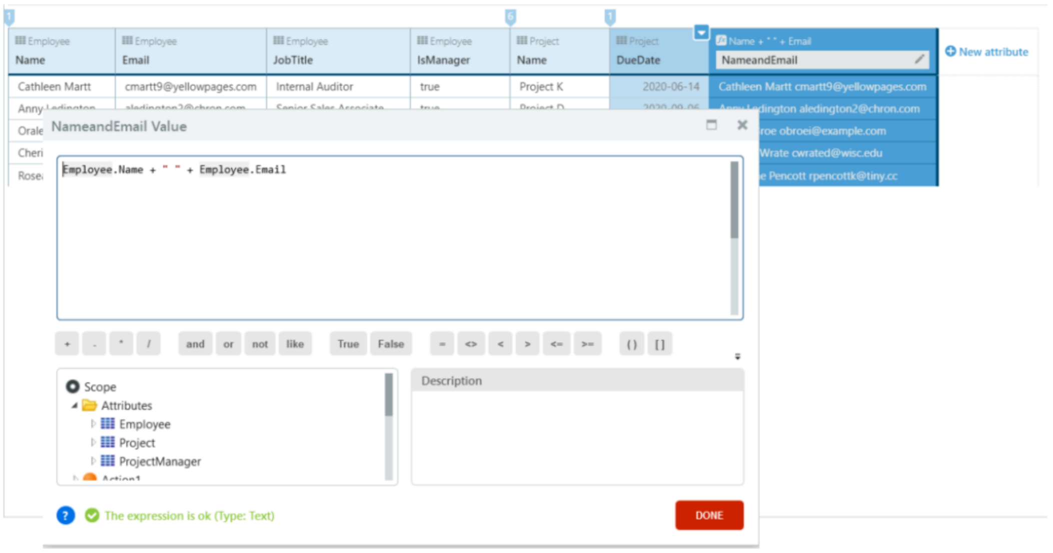
Figure 2.8: Query Design functions while interacting with Query Result

test purpose in the context of the query result visualization.

Figure 2.7 presents a usage example of the last functionalities mentioned, to create a query to filter and sort dates, regarding the value of a variable.

On the other side, the area dedicated to viewing the query output provides also some functionalities to design the query while the user is interacting and exploring the query result. Therefore, as represented by an example in Figure 2.8, the user can change the query when he performs a right-click on a column or when clicks on the new attribute. The only options in the list presented that does not change the query are the hide options since they just change the result in the presentation layer. So, if the user hides a set of columns, he will not see them in the result preview visualization area, however, the query output did not change. Besides, the user can add other attributes based on the existing ones, so Figure 2.9 shows an example of this functionality.

When a group by or an aggregation function (COUNT, SUM, MAX, MIN, AVERAGE)



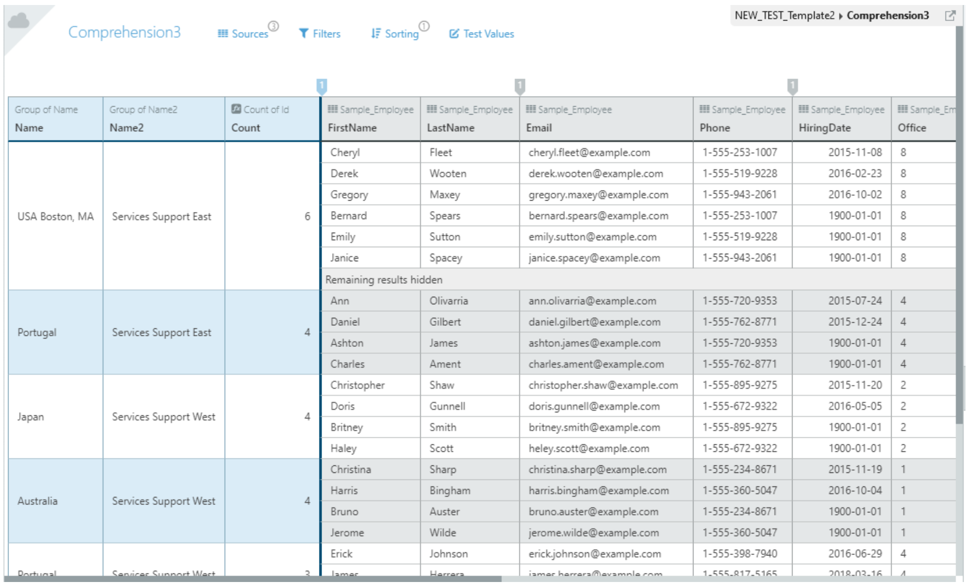
Employee Name	Employee Email	Employee JobTitle	Employee IsManager	Project Name	Project DueDate	Name + " " + Email NameandEmail
Cathleen Martt	cmartt9@yellowpages.com	Internal Auditor	true	Project K	2020-06-14	Cathleen Martt cmartt9@yellowpages.com
Anny Ledington	aledington2@chron.com	Senior Sales Associate	true	Project D	2020-09-06	Anny Ledington aledington2@chron.com
Orale						roe obroei@example.com
Cheri						Wrate cwrated@wisc.edu
Rose						e Pencott rpencottk@tiny.cc

NameandEmail Value

```
Employee.Name + " " + Employee.Email
```

The expression is ok (Type: Text)

Figure 2.9: Calculated Attribute Insertion



Group of Name Name	Group of Name2 Name2	Count of Id Count	Sample_Employee FirstName	Sample_Employee LastName	Sample_Employee Email	Sample_Employee Phone	Sample_Employee HiringDate	Sample_Em Office
USA Boston, MA	Services Support East	6	Cheryl	Fleet	cheryl.fleet@example.com	1-555-253-1007	2015-11-08	8
			Derek	Wooten	derek.wooten@example.com	1-555-519-9228	2016-02-23	8
			Gregory	Maxey	gregory.maxey@example.com	1-555-943-2061	2016-10-02	8
			Bernard	Spears	bernard.spears@example.com	1-555-253-1007	1900-01-01	8
			Emily	Sutton	emily.sutton@example.com	1-555-519-9228	1900-01-01	8
			Janice	Spacey	janice.spacey@example.com	1-555-943-2061	1900-01-01	8
			Remaining results hidden					
Portugal	Services Support East	4	Ann	Olivarria	ann.olivarria@example.com	1-555-720-9353	2015-07-24	4
			Daniel	Gilbert	daniel.gilbert@example.com	1-555-762-8771	2015-12-24	4
			Ashton	James	ashton.james@example.com	1-555-720-9353	1900-01-01	4
			Charles	Ament	charles.ament@example.com	1-555-762-8771	1900-01-01	4
Japan	Services Support West	4	Christopher	Shaw	christopher.shaw@example.com	1-555-895-9275	2015-11-20	2
			Doris	Gunnell	doris.gunnell@example.com	1-555-672-9322	2016-05-05	2
			Britney	Smith	britney.smith@example.com	1-555-895-9275	1900-01-01	2
			Haley	Scott	heley.scott@example.com	1-555-672-9322	1900-01-01	2
Australia	Services Support West	4	Christina	Sharp	christina.sharp@example.com	1-555-234-8671	2015-11-19	1
			Harris	Bingham	harris.bingham@example.com	1-555-360-5047	2016-10-04	1
			Bruno	Auster	bruno.auster@example.com	1-555-234-8671	1900-01-01	1
			Jerome	Wilde	jerome.wilde@example.com	1-555-360-5047	1900-01-01	1
Portugal	Services Support West	3	Erick	Johnson	erick.johnson@example.com	1-555-398-7940	2016-06-29	4
			James	Herrera	james.herrera@example.com	1-555-817-5165	2018-03-16	4

Figure 2.10: Representation of the attributes aggregated through Group By operations or other aggregation functions (COUNT, SUM, MAX, MIN, AVERAGE)

is applied through the context menu exemplified in Figure 2.8 the data is aggregated automatically and is presented as the example shown in Figure 2.10 illustrates. The aggregated attributes resulted from the application of these operations are the ones that belong to the query output as well as occur in SQL. However, the query output preview of Aggregates represents also in a different color a preview of the corresponding data in the right side. For example, in Figure 2.10, it was counted the number of employees (COUNT) by each office and department (2 Group Bys applied), but it is possible to read at the right side of these three first columns the data that was aggregated.

Chapter 3

Related Work

Since users were a fundamental component of the study, their relationship with data and what are their expectations of these query systems were investigated and they will be discussed in this chapter. Moreover, a set of technologies and techniques will be described and compared. This information can turn to be useful to explore the solution, and understand different points of view to manage problems, regarding the project's scope.

3.1 Query Conceptual Models

The perception of the user's conceptual model is important to understand how the user reason while interacting with the system the perform the intended actions. A query is built to gather data. To transmit what is the intended data, the user needs to think about how it could express the data required in the query. The understanding of the user's conceptual model could be useful to remove the existing gap between what the user wants to query from the database and what system register that the user wants to retrieve.

Some studies have analyzed this reasoning process of the users when they were writing queries. Siau *et al.* [25] have referred that "The semantics communicated through the interface can be classified according to abstraction levels, such as the conceptual and logical levels". Also, there is one more level, the physical level, where is considered the system details, such as physical storage and access structures. Since the physical level is low level, usually, the conceptual and logical levels are most used. The logical level takes into account abstract structures for data and operations, and the conceptual level uses real-world objects and concepts to communicate. Through an empirical method of evaluation, the conceptual level has revealed a higher accuracy. Also, this abstraction level makes the users more confident in their answers than the physical level or logical levels. Moreover, the time that users need to design the queries is reduced using conceptual levels [25].

Reisner [23] provided a model of query writing from the reading of the query intention in an English statement to the query writing in SQL (Figure 3.1a). After understanding what data is required, the user applies two parallel steps. In the **Template Generation** phase, the user formulates a template identifying the SQL keywords necessary, such as SELECT, FROM, and WHERE. In the other step, called **Lexical Transformation**, the user identifies the name of the tables and columns involved. Finally, the results of these two steps are combined in the last step, denominated **Insertion**, in order to produce the final query [23]. The recall of table and column names represents a significant use of long-term memory, being a concern that should be taken into account [26].

In the same field, Ogden [20] presented a three-stage cognitive model of the query process (Figure 3.1b):

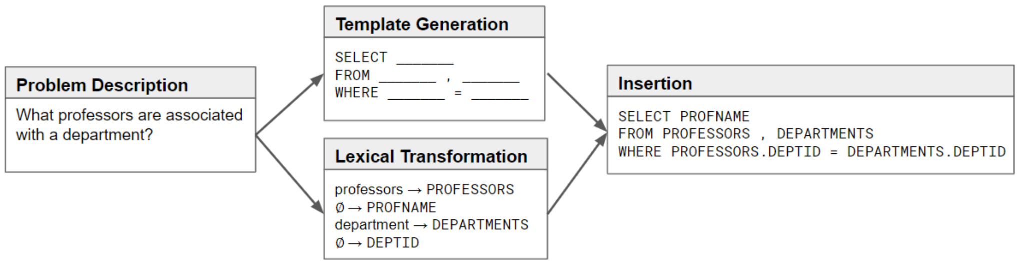
1. **Query Formulation**: according to the existing data, the user specifies, in natural language, what data is required;

2. **Query Translation**: regarding the existing data model, the operations and relations necessary are defined, in order to adapt the natural language request to the pragmatics of the intended query;

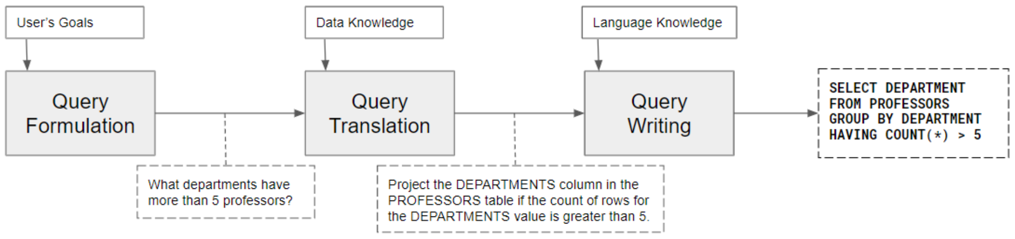
3. **Query Writing**: the information of the previous steps is used to built the query, using the syntactic and the semantics of the query language.

Comparing the two previous models, the Lexical Transformation phase is present in the Query Translation phase of the latter model, as well as, Template Generation and insertion are part of the Query Writing phase [6]. Moreover, although the query writing and comprehension are the focus of this work, it was verified in a comparison between three different models (relational model, extended-entity-relationship model, and object-oriented model) that the data model influence the query writing and comprehension [6].

The query comprehension is one of the important concerns of this work, since it is important to consider if the query representation, no matter if it is visual or textual, indicates clearly what data will be gathered. Chan *et al.* [6] have postulated that the query comprehension could be covered by the reverse of the stages included in the Ogden Model. First, the user should identify the data structure and operations, to translate them, in the next step, to the natural language. After this, the user needs to read and understand what data gathering is intended. Moreover, in this evaluation, it was concluded that although data modeling influence query writing, it does not influence the query comprehension. The explanation is provided by the authors: "Both query writing and comprehension require an understanding of the query language syntax. This is a component not needed in the data modeling task."



(a) Reisner Model (adapted: Reisner [23])


User's Goals
Data Knowledge
Language Knowledge
Query Formulation
Query Translation
Query Writing
What departments have more than 5 professors?
Project the DEPARTMENTS column in the PROFESSORS table if the count of rows for the DEPARTMENTS value is greater than 5.
SELECT DEPARTMENT
FROM PROFESSORS
GROUP BY DEPARTMENT
HAVING COUNT(*) > 5

(b) Ogden Model (adapted: Ogden [20])

Figure 3.1: Models of Query Writing Process.

The experience with the data involved is another factor that influences the query comprehension. If novice users do not understand the data involved, they cannot validate if the query result is correct. This situation was analyzed by Robb *et al.* [24] that were distinguished the query process between new users and experienced users (Figure 3.2). Besides, they concluded that if the novice users were alerted to the details of the data queried, the query effectiveness will increase.

3.2 Query Formulation Problems

Since the goal of this work is the improvement of an interface that allows its user to build queries, it is important to summarize a set of significant problems that usually occur in query formulation. The problems that will be presented are related to SQL queries. However, as the visual tool of this work aims to substitute some functionalities of SQL, the comprehension of the interaction problems that exist in the textual language can be considered and mitigated in the development of the new interface.

Lu et al. [15] have evaluated the SQL usage in a diverse population, which includes people of different enterprise areas with different levels of experience in the database systems domain. The authors concluded that the comprehension of the queries is difficult, as well as the logical errors are difficult to detect. Moreover, the joins and aggregation functions are the other problems pointed out.

The query errors could be syntactic or semantic. The **Syntactic Errors** are related to the grammar rules of the language and are detected by the compiler. Therefore, the impact of these errors is reduced since the user can see that the query is incorrect through the compiler alert. The **Semantic Errors** are a major concern because they occur when

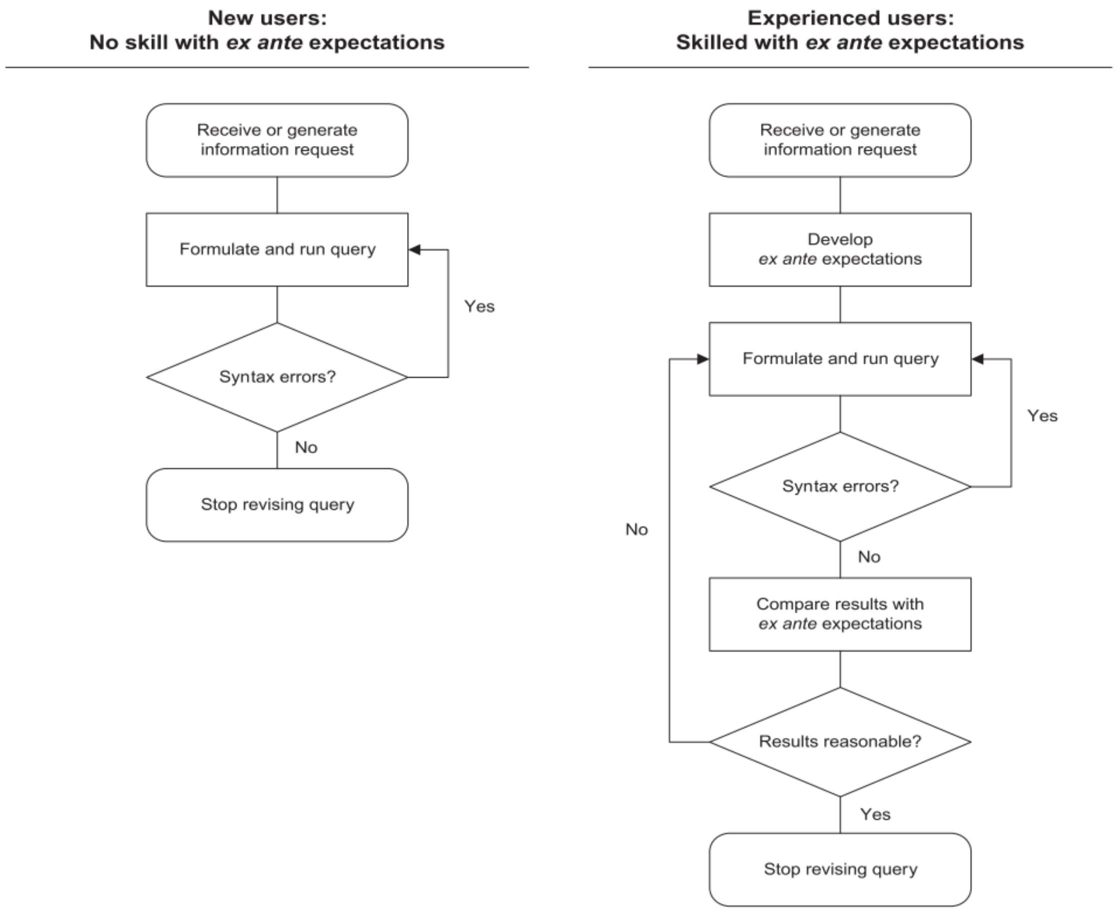

New users:
No skill with *ex ante* expectations
Receive or generate information request
Formulate and run query
Syntax errors?
Yes
No
Stop revising query
Experienced users:
Skilled with *ex ante* expectations
Receive or generate information request
Develop *ex ante* expectations
Formulate and run query
Syntax errors?
Yes
No
Compare results with *ex ante* expectations
Results reasonable?
No
Yes
Stop revising query

Figure 3.2: New and experienced users' query processes (source: Robb *et al.* [24]).

the returned information is not intended by the user, even if the query does not have compilation errors [26]. These errors could affect the correctness of the results.

In SQL, the join clauses are used when is necessary to merge data from different tables in one column in order to specify which data of each table will be considered. Several studies have demonstrated that the indication of the join clause is one of the most representative semantic errors [1, 15]. Smeller [26] has studied what are the cognitive causes that lead the user to forget the join clauses:

- **Working memory overload**: if the user needs to recall the table and column names, and the conditions necessary after the identification of the join's requirement, the required join clauses could be forgotten in this period;

- **Absence of the clue**: when the statement that explains what data is required do not present clues for the join necessity;

- **Procedural fixedness**: when a query that only extracts data from one table is reused for another that requires the join clauses but this join is forgotten;

- **Ignorance**: when the user does not know how to merge the tables and specify the join clauses.

The cognitive causes of the problems are important to develop interfaces that could mitigate the existing problems in the query formulation. For instance, if the join clues are provided in the interface, the user does not need to remember them. This approach, which follows one of the usability heuristics presented by Nielsen [18], minimizes the user memory load.

A study that evaluated novice programmers' semantic mistakes concluded also that omissions are the principal semantic error, mainly in the WHERE clause [1]. Besides, the authors have referred also the problem of working memory overload: "This error may occur when the capacity of a student's working memory is surpassed".

Another study has analyzed a large dataset of queries composed by university students enrolled in an introductory database course. However, this study is more extensive and includes a list of the principal errors committed by the students [28]. Continuing the focus on the semantic errors, the authors have pointed out the following error categories: inconsistent expressions, inconsistent joins, joins omission, duplicate rows, redundant column outputs [28].

3.3 Visual Query Composition

The interfaces to build queries resort to visual representations to communicate with the user. Catarci *et al.* [3] presented an interesting classification according to the visual formalism which the interface is based on:

- **Form-based**: based on forms, which can be seen as a rectangular grid of other components (subforms, groups of cells, a combination of cells, etc.) that group objects in a named collection regarding its structure. Forms and tables are similar, but contrary to the tables, forms allow nesting. Thus, forms can be seen as a generalization of tables. In this approach, the relationships can be represented among cells, cells subsets, or even the overall set, providing to the user three information levels;

- **Diagram-based**: usage of graphical representations, such as graphs, charts, and diagrams to better transmit the relationships among data. The aim is to use visual representations to help the understanding of the relationships between concepts which are represented by textual labels;

- **Icon-based**: as the opposite of the diagram-based, this type of interface tries to facilitate the understanding of the concepts instead of relationships. So, Icons are used, which are visual segmented objects to transmit a message or information, using analogies and metaphors with the real-world objects, or even conventions that are used to express no tangible objects, as computer processes;

Table 3.1: Query Language Requisites

Specification	Description	SQL Indication
Data Source	Entities and attributes which will be presented in the query	Using SELECT and FROM statements
Merge Type	Define how will be merged attributes of different entities	Using JOIN clauses
Filtering Criteria	Criteria that can be used to filter records, presenting in the result only those that fulfil a set of conditions	Using WHERE or HAVING clauses
Sorting Criteria	Define what are the criteria to sort the records of the result	Using ORDER BY
Aggregation Functons	Group a set of records by comparison or using mathematical functions	Using GROUP BY statements or SQL functions, such as MIN, MAX, COUNT, AVG and SUM
Calculated Attributes	Attributes added, based on existing ones	Using SELECT statement
Distinct Values	If only different values will be considered in the result (removing duplicated values)	Using SELECT DISTINCT statement
Unions	Combine the result of two different queries	Using UNION operator
Subqueries	Defining a query that uses other queries, for example, to filter the result	Nesting SELECT statements

- **Hybrid**: these approaches can combine the previous visual formalisms in order to select the best combination of advantages to the application usage domain.

In order to compare different interfaces, it is essential to analyze what a query language has to support to build the query. Accordingly, Table 3.1 presents the query creation required specifications, comparing them with the respective indication in SQL.

Nevertheless, there are two relevant aspects, according to the last requisites presented: the interaction process to indicate the query specifications, the overview of what data wants to be retrieved using the current query. Both are fundamental since a good visual query language aims to simplify not only, the query formulation process but also, the query readability, promoting an efficient and effective recognition of what are the desired data.

Data Source:

Chartio [33] has two components to query databases visually: using the Data Explorer [34] or using the new Visual SQL [36]. Regarding the data source specification, these two systems use different strategies to select and present the entities and attributes related to the query. In the Data Explorer, the user can expand the items of a list of tables in a scrollable and searchable tree view, which is pinned in one side of the window, to choose the desired attributes. This system divides the attributes into two different types: Measures and Dimensions. Usually, measure refers to quantitative data and dimensions to categorical data. So, to insert the attributes in the query, users can drag and drop the required attributes to the form-based interface that contains the Measures, Dimensions, and Filters of the query (Figure 3.3a) [34].

On the other hand, the new component of Chartio, Visual SQL provides a different interface to select the data sources. Contrary to the previous approach, in this interface, there is no fixed list to choose the attributes. In this way, there is only a search text component that is activated when the user clicks on "add column" action. When this action occurs, a pop-up style component that has a list, similar to the referred above, where it is possible to preview some data entries of the table, is presented (Figure 3.3b) [36].

In the systems referred above the columns are added one by one sequentially, but other systems have different methods to select the table's columns. For example, in Tableau Prep [79] there is a checkbox list to chose the intended attributes (Figure 3.3c), and in Microsoft Power BI [52] the table is chosen using a list, and all its attributes are added automatically. Also, users can remove, the columns not desired afterwards [54, 80].

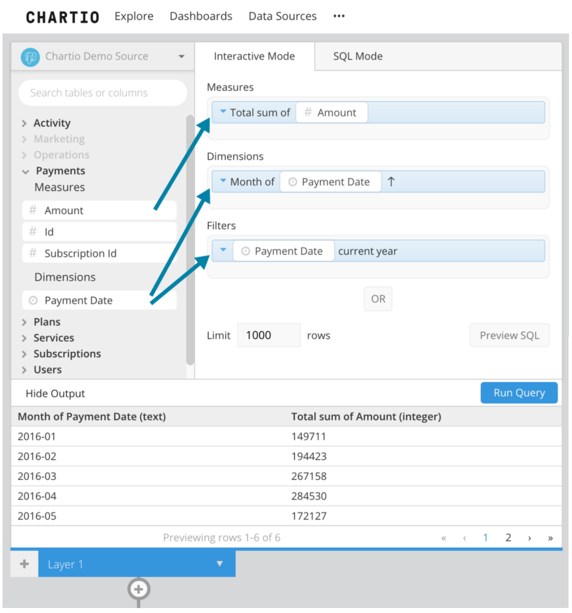
Other systems, as Devart dbForge Query Builder [42], uses a diagram-based interface to select the entities and attributes of the query. In this system, the user can drag and drop the desired tables to the diagram area, and select through checkboxes the intended attributes, that are presented in the database schema diagram (Figure 3.3d).

Merge Type:

Merges are used when it is necessary to extract data from different tables, so it is necessary to establish what is the join kind to merge the data. Therefore, the interface needs to adopt an interaction and representation technique to specify it. To define a join in Devart dbForge Query Builder [42], the user can only select the attributes' checkboxes of the different tables and the system generates an inner join automatically. In this system, there are buttons on the toolbar to select all rows of one table, of another, or both, allowing to perform left, right and outer joins respectively [41].

Another approach used by some systems, such as Chartio Data Explorer [34] and Microsoft Power BI [52], is a form-based interface to insert a join. In the former, two queries can be merged clicking on a button to popup a form that can be used to select the merge type and the first columns that will be merged using dropdowns [34] (Figure 3.4a). Also, if there are null values on the merge related columns, there is an option to include or not the null values match rows [35]. Similarly, the latter provides a button to merge queries that opens a modal where the attributes that will be used on the merge (viewing also a table preview) and the join kind can be chosen, using a dropdown [54] (Figure 3.4b).

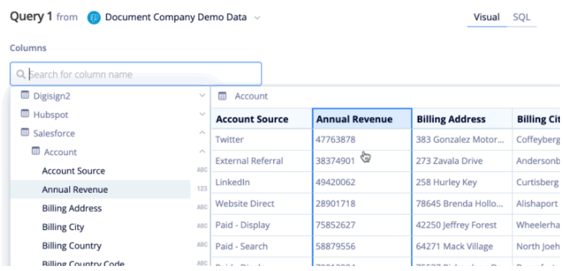
Tableau Prep [79] provides two options to start a join between two tables: clicking



Month of Payment Date (text)	Total sum of Amount (integer)
2016-01	149711
2016-02	194423
2016-03	267158
2016-04	284530
2016-05	172127

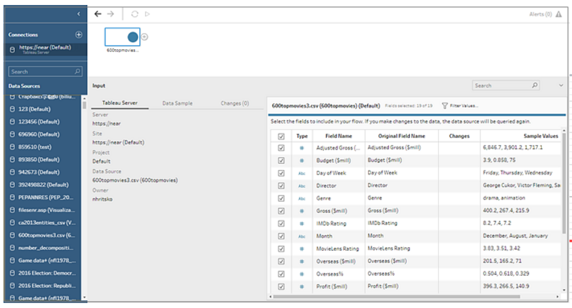
Previewing rows 1-6 of 6

(a) Chartio Data Explorer (source: Chartio[34])

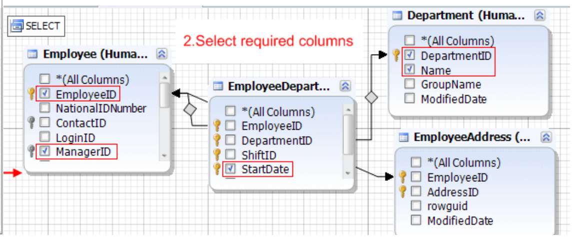


Account Source	Annual Revenue	Billing Address	Billing Cit
Twitter	47763878	383 Gonzalez Motor...	Coffeyberg
External Referral	38374901	273 Zavala Drive	Andersonb
LinkedIn	49420062	258 Hurley Key	Curtisberg
Website Direct	28901718	78645 Brenda Hollo...	Alishaport
Paid - Display	75852627	42250 Jeffrey Forest	Wheelerha
Paid - Search	58879556	64271 Mack Village	North Joeh

(b) Chartio Visual SQL (source: Chartio [36])



(c) Tableau Prep (source: Tableau [80])



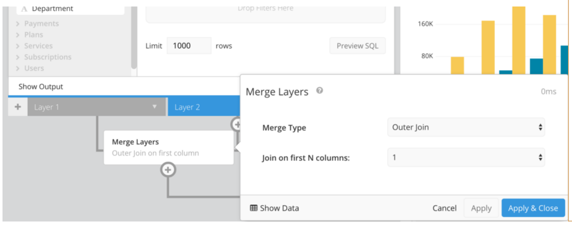
(d) Devart dbForge Query Builder (source: Devart [42])

Figure 3.3: Different approaches to select the entities of the query.

on a "add join"hover button above the table visual representation with the suggestion of related tables, or merge the visual representation of two tables using a drag and drop action. After this selection, the inner join type is selected automatically by the system according to the tables' relationship [74, 75]. However, the user can configure the join in a dedicated section (Figure 3.4c) where it is possible to define the join type using a Venn Diagram, to manage the join clauses using dropdown lists to select the fields, including also some join clause recommendations based on the database schema. Moreover, a summary of the join result that contains counters with the values included and excluded by each table, in a visual way using diagrams is provided. Finally, a list of the values included and excluded, where the red values represent the values excluded is presented, as well as a preview of the join result [75].

Filtering Criteria:

To represent and manage the filtering criteria of the queries, usually it is used text to indicate the logical conditions. However, some systems are trying to optimize the



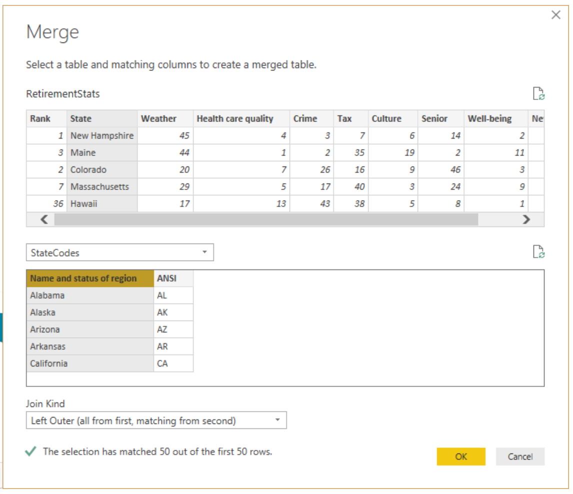
Limit 1000 rows

Merge Layers
Outer Join on first column

Merge Layers 0ms

Merge Type: Outer Join

Join on first N columns: 1


Merge

Select a table and matching columns to create a merged table.

RetirementStats

Rank	State	Weather	Health care quality	Crime	Tax	Culture	Senior	Well-being	Ne
1	New Hampshire	45	4	3	7	6	14	2	
3	Maine	44	1	2	35	19	2	11	
2	Colorado	20	7	26	16	9	46	3	
7	Massachusetts	29	5	17	40	3	24	9	
36	Hawaii	17	13	43	38	5	8	1	

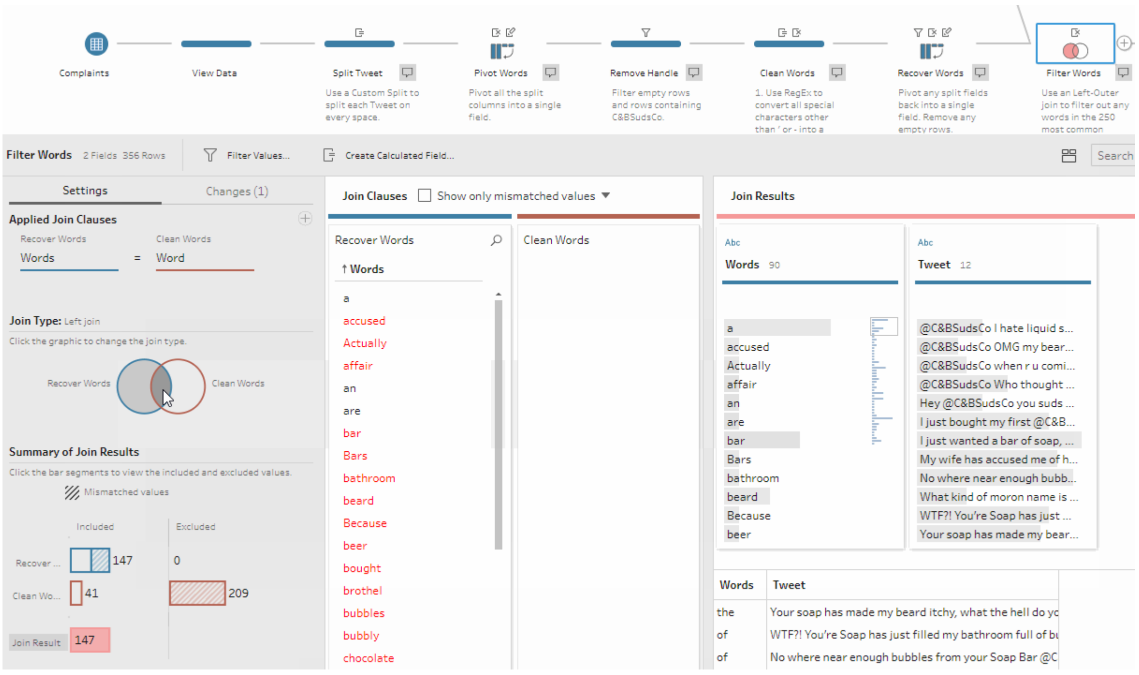
StateCodes

Name and status of region	ANSI
Alabama	AL
Alaska	AK
Arizona	AZ
Arkansas	AR
California	CA

Join Kind: Left Outer (all from first, matching from second)

✓ The selection has matched 50 out of the first 50 rows.

(a) Chartio Data Explorer (source: Chartio [34]) (b) Microsoft Power BI (source: Microsoft [54])



Split Tweet: Use a Custom Split to split each Tweet on every space.

Pivot Words: Pivot all the split columns into a single field.

Remove Handle: Filter empty rows and rows containing C&BSudsCo.

Clean Words: 1. Use RegEx to convert all special characters other than ' or - into a

Recover Words: Pivot any split fields back into a single field. Remove any empty rows.

Filter Words: Use an Left-Outer join to filter out any words in the 250 most common

Filter Words 2 Fields 356 Rows

Applied Join Clauses

Recover Words Words = Clean Words Word

Join Type: Left join

Click the graphic to change the join type.

Summary of Join Results

Click the bar segments to view the included and excluded values.

	Included	Excluded
Recover ...	147	0
Clean Wo...	41	209
Join Result	147	

Join Clauses ☐ Show only mismatched values

Recover Words ↑ Words: a, accused, Actually, affair, an, are, bar, Bars, bathroom, beard, Because, beer, bought, brothel, bubbles, bubbly, chocolate

Clean Words

Words 90: a, accused, Actually, affair, an, are, bar, Bars, bathroom, beard, Because, beer

Tweet 12: @C&BSudsCo I hate liquid s..., @C&BSudsCo OMG my bear..., @C&BSudsCo when r u comi..., @C&BSudsCo Who thought ..., Hey @C&BSudsCo you suds ..., I just bought my first @C&B..., I just wanted a bar of soap, ..., My wife has accused me of h..., No where near enough bubb..., What kind of moron name is ..., WTF?! You're Soap has just ..., Your soap has made my bear...

Words	Tweet
the	Your soap has made my beard itchy, what the hell do yc
of	WTF?! You're Soap has just filled my bathroom full of bu
of	No where near enough bubbles from your Soap Bar @C

(c) Tableau Prep (source: Tableau [30])

Figure 3.4: Data merging approaches.

usability, helping the user in the specification process through some autocompletes. These diminish the necessity to remember all the syntax and the name of the functions. Besides, other systems present the logical conditions using a more structured layout, although keeping resorting in a textual representation. An example is Devart dbForge Query Builder [42] that represents the WHERE and HAVING clauses in a tree where the user can organize the conditions into groups [39].

Furthermore, some query systems also allow the user to view the query result, providing a shortcut in the column's name to insert filters. So, the user can change the query while is viewing the results. In this way, the user can apply a filter and view its effect almost immediately. Power BI Query Editor [52] is an example of a system that uses this approach.

Different interfaces can be used to select filters regarding the field data type. The advantages of the graphical interfaces could support the filtering criteria definition. For example, if a filter is applied to constraint dates, then a date input box with a visual calendar can be useful to simplify the date typing. In this way, some software, such as Chartio Visual SQL [36] and Chartio Data Explorer [34], not only helps in the data typing but also provide dropdown lists that contain operators that can be applied to the referred data type (e.g. less than, contains, like, etc) to helps the user in the boolean operator specification [34, 38].

Tableau Prep [79] follows this more strictly, distinguishing between data types when filtering criteria are indicated. It provides different forms depending on the data type. The main difference of this system is that not only provides a more diversity of controls, as integrating range selectors, radio button, and the option to include or exclude fields through an action accessible near its value, but also gives to the user the option to use a calculation form where the interaction style is more textually and more extensible [77].

Sorting Criteria:

Usually, in the actual VQIs, the sorting criteria could be indicated in two ways: a right-click action on the column header of the table that represents the query result, or using a form-based interface to define the sort criteria of each entity. Devart dbForge Query Editor [42] is a pure example of the first approach [43]. Chartio Data Explorer [34] adopts the second approach, providing a pop-up form to apply sorting criteria to the query. The user can use this form to select the intended attributes and the criteria to apply [32]. Moreover, Chartio Visual SQL [36], Tableau Prep [79], and Microsoft Power BI [52] combines the previous solutions with the possibility to redefine the priority of the sorting criteria, through drag-and-drop actions [38, 54, 78].

Aggregation Functions:

In order to perform aggregations functions, such as MIN, MAX, COUNT, AVG, SUM, or GROUP BY, some systems provide these functionalities through interaction with the query result table. Devart dbForge Query Editor [42] provides this option to create an aggregation function. Moreover, in this editor exists another aggregation dedicated view which contains the aggregation functions in a tree view. In this view, users can group or ungroup the elements present in the window [40]. Microsoft Power BI [52] allows also the users to add aggregations through the right-clicking on the column header, but in this case, it is open a form to enter the intended function and columns [53]. Chartio Visual SQL [36] and Chartio Data Explorer [34] presents a pop-up form where could be selected the columns and the aggregations intended [37]. Using a different interaction strategy, in Tableau Prep [79], the user drag and drop the desired columns to a specific area that is divided into two: Grouped Fields and Aggregated Fields. The first is to add the SQL corresponding GROUP BY, and the second to add the other aggregation functions, such as COUNT, MIN, MAX, AVG, and SUM.

Other Specifications:

Regarding the option to add new calculated attributes, all the systems referred above allow inserting calculated attributes to the query, excepting the Devart dbForge Query Editor which does not support [38, 53, 76].

The option to show only distinct values is provided in Tableau Prep [79], Microsoft Power BI [52], and Devart dbForge Query Editor [42], through a button or a checkbox to does not see in the result the duplicated values. However, Chartio Visual SQL [36] and Chartio Data Explorer [34] does not support a specific interaction method to specify this. Nonetheless, the distinct effect can be applied, using the group by in all the columns of the query.

Some system provides visual options to build queries that contain UNIONs. For example, Chartio Visual SQL [36] and Chartio Data Explorer [34] provides this option in the same components of the joins. In these systems, when the user chooses the join type, the union is one of the join types available, although there is a different operation in SQL. Tableau Prep [79] and Microsoft Power BI [52] present different options between the joins and unions, but the interface and the interaction strategies are similar [54, 75].

Moreover, a visual way to perform subqueries is provided by the diagrammatic-based interface of Devart dbForge Query Editor [42], using tabs to alternate between the selected query. Links are used as assistants and shortcuts to view and change between the queries [44, 45].

3.4 Discussion

The conceptual models of query writing presented in this chapter will be taken into account along the design of the solution since these are a good baseline to understand how users reason while are interacting with the system to achieve their goals. The system's tasks will be optimized according to the presented users' conceptual models presented, in order to reduce the semantic errors, as much as possible. Therefore, the most relevant semantic errors that occur using SQL were presented. In the design of the solution, these errors are part of the problems to solve using a new user interface and UX. As referred in section 3.2, the semantic errors will be the main concern since these could have a negative impact on the results. The syntactic errors also will be addressed in this work but require less study about the users' conceptual model. The major concern in this type of error is to provide the maximum feedback to the user. In that way, the user will understand the error and correct it.

Furthermore, since the design of an interface needs to tackle with the usability attributes trade-off, it is important to characterize and consider the problems of the application's target users. This characterization is essential to define the usability priorities of the interface. The population used in the studies presented in this chapter will be an excellent reference to the target user analysis. The target users of the intended system are low-code developers. Since low-code development has advantages not only for not

Table 3.2: VQSs Summary

Feature \System		Chartio Data Explorer	Chartio Visual SQL	Tableau Prep	Power BI Query Editor	Devart dbForge Query Builder
Tables and Columns	Only the required	✓	✓	✓	✗	✓
	All the attributes at once	✗	✗	✓	✓	✓
	Remove Columns	✓	✓	✓	✓	✓
Merge	Inner Join	✓	✓	✓	✓	✓
	Left Join	✓	✓	✓	✓	✓
	Right Join	✗	✓	✓	✓	✓
	Full Outer Join	✓	✓	✓	✓	✓
	Cross Join	✓	✓	Using Calculated Attributes	Using Calculated Attributes	Textually
	Null Option	✓	✓	✗	✓	✗
	Define Join Condition	✓	✓	✓	Selecting Columns Visually	Textually
Filtering Criteria		✓	✓	✓	✓	✓
Sorting Criteria		✓	✓	✓	✓	✓
Aggregation Functions		✓	✓	✓	✓	✓
Unions		✓	✓	✓	✓	✗
Calculated Attributes		✓	✓	✓	✓	✗
Distinct		✗	✗	✓	✓	✓
Subqueries		✗	✗	✗	✗	✓

programmers but to programmers with different levels of expertise, the target users of the system are wide. Thus, the studies that have evaluated a wide diversity of users, such as the evaluation of Lu *et al.* [15], could reveal important results to the work development. Moreover, the studies presented that have analyzed students of database systems [1], [28], could add more important data, since some details only occur if the queries formulated are more complex.

Finally, a set of actual graphical user interfaces used to formulate queries was presented and compared. The approaches used by other systems are a relevant object study for the design and the development of the new interface. Table 3.2 represents a summary of the database query operations supported by each one of the presented systems.

Chapter 4

Methodologies

The problem approached in this dissertation required a reformulation of the visual interface used to query data. As such, the methodology used to design, implement and evaluate each phase of the solution development was regarded as a key factor to keep the right focus until reach the final stage.

In that way, this chapter will present an overview of the methodology adopted as well as how users were integrated into the process in order to maintain a user-centered design approach.

4.1 Problem Exploration

Even though there was a general idea of the problems of the existing interface, a complete plan was designed to perform a deep exploration of the aforementioned problems. The main concern was to obtain a global and complete view of the problems through different gathering strategies in order to examine the problem from different perspectives rather than using only a subset of users or data, which could bias the problems' perception.

Firstly, the existing problems were identified in a wide sense from the perspective of a user who does not have context about how the query builder works, in order to realize the difficulties that new users who tried to start using the interface could feel. After that, other gathering techniques were applied in order to perceive also the problems felt by expert users and progressively understand the most detailed aspects of the problems presented. Therefore, the methods presented below, which will be further detailed in section 5.1, were applied, in this order, to obtain a wide and diverse view of the existing problems:

- **Analysis:** Visualization of the OutSystems tutorials regarding data querying and self-exploration of the interface in order to analyze and identify a preliminary set of preliminary problems;

- **User Interviews:** Simple and direct dialogues with end-users of the OutSystems platform to listen their opinion concerning the query builder. The main aim was to notice in which reasons they use it, what are their major difficulties using it, or even the struggles or barriers which make them to not use it;
- **Data Analysis:** Analysis of the queries formulated in SQL through the OutSystems Platform extensibility feature which allow developers to add queries in SQL to perceive if SQL is used only for advanced used cases not supported by the visual query, or if it is often used to cover simple use cases;
- **Community Ideas:** Exploration of the discussions in the forums of the OutSystems Community regarding data querying to aggregate more information about problems users feel and suggestions to improve it;
- **Existing Interface Evaluation:** Usability tests to the existing interface in order to visualize directly how users interact with the system as well as what they feel when trying to understand and formulate database queries using the visual query builder. The results of this problem information gathering would promote also important results to validate the further prototypes since it would be possible to compare the usability of the new prototypes with the existing interface.

4.2 User Analysis

As mentioned, an user-centered design approach was applied and hence an analysis of the target users of the system, which will be further presented in section5.2, was performed to perceive and categorize the users who use the query builder. In this analysis, the users were classified and divided into three groups: OutSystems Developer, Software Developer, and Citizen Developer.

In that way, the topics that could most branch out the users' requirements and expectations were pointed out to perceive how users could be divided into different groups. These groups were used throughout the design process to characterize the target users of the system. Each group represents a subset of the system's users, which should be separated from the remaining ones due to their user's profiles. Henceforth, all usability tests were performed in an equitable manner with users of all groups in order to obtain a representative sample of the end-users of the system.

4.3 Iterative Design

After comprehending the existing problems and the target users of the system, the phases for the sake of the design and development of the final solution were planned. Accordingly, the design strategy adopted, which will be further detailed in Chapter 6, will be briefly presented. The chosen methodology used an iterative design strategy in order to keep a user-centric design, which prioritizes the users' needs according to the mentioned in section 2.1.2.

Sketching: The design process started with an initial sketching phase, where the first solution ideas were explored and drafted. In this technique, the integration of new ideas into the existing interface is favored given the fact it is an interactive process where the ideas are tested progressively. The most important aspect was to think about system transversal changes and not about particular details of specific components, since these details could be refined later. The outcome of this phase sets a more concrete idea of what can be inserted in the prototype, even if it is necessary to think more about how to implement it later.

Keeping in mind the ideas explored in the sketching phase, it is possible to start to build the prototypes that will be tested by users. The evolutionary prototyping principle (described in section 2.1.2.2) was applied. Accordingly, the last prototype was used as a baseline to develop the prototype of the next iteration. As mentioned in section 2.1.2.2, the first prototypes should be low-fidelity prototypes, and iteration by iteration, this level should increase in order to refine details in the interface. In that way, it was decided to build two prototypes:

- **Paper Prototype:** Simple low-fidelity prototype implemented in paper using a ruler, a set square, and writing materials. Through that approach, it is possible to implement the first ideas faster and reduce the risk of adoption failure, since the implemented ideas were already tested. The main concern of this type of prototype is that the design of the major interface changes might affect users mental model which enables the early detection of which choices should continue to the next iterations or if they need to be redesigned.

- **Service Studio Prototype:** This is the final prototype, developed using C#, Typescript [55], and React [69], which is integrated in the new Design of the Service Studio. Through this prototype, the solutions implemented were validated and compared with the previous existing implementation in order to validate if the usability of the system has improved with the implemented changes.

In each one of the two above-mentioned prototypes, it was included the following phases: design, implementation, and evaluation. In the Design phase solution ideas that could be applied were set up as well as the priorities to be tackled in the prototype. The Implementation phase refers to the concrete prototype development process. Finally, in the Evaluation phase, the prototypes were tested by end-users, according to the testing approach further explained in section 4.4 and section 4.5.

Regarding evaluation, it was necessary to establish how many users should be tested in each evaluating phase. In these phases, not only the two mentioned prototypes were considered, but also the evaluation of the existing interface mentioned in section 5.1.5.

Table 4.1: Number of users tested by each user group and by each solution evaluated

	Previous Implementation	Paper Prototype	Service Studio Prototype	Total
OutSystems Developer	10	5	10	25
Software Developer	10	5	10	25
Citizen Developer	10	5	10	25
Total	**30**	**15**	**30**	**75**

The data extracted from the evaluation of the existing interface provides an opportunity to compare the existing usability against the usability of the final solution. Thereby, the number of users tested in the first prototype is also an important factor.

Nielsen performed several studies quantifying how many users should be tested in a usability study, leading to the conclusion that 5 users were a sufficient number for qualitative studies since it is possible to get the maximum benefit-cost ratio - "Testing with 5 people lets you find almost as many usability problems as you'd find using many more test participants"[58] [59]. However, to perform quantitative analysis it is necessary to get at least 20 users in order to get statistical relevance [59].

According to the studies mentioned, at least 5 users of each user group (described in section 5.2.) should be tested since only a qualitative analysis will be performed to validate user's interaction with the applied changes. Nevertheless, some statistical results should be also subject of analysis in order to compare if the new solution provide improved usability than the existing version. Hence, it was decided to test more users in the existing implementation and in the final prototype in order to obtain enough elements to conduct a quantitative analysis. In that way, Table 4.1 shows how many users were tested by each user group and by each solution evaluated.

To compare the different phases, all tests were performed using the same testing scenarios, which are presented in the following section.

4.4 Testing Scenarios

Given the wide scope of the usability problems identified, a plan of the testing approach was required in order to evaluate the most important aspects of the user-interface communication, given the short period available to test the users. Following the plan, it was possible to keep the testing focus on the prioritized details in a feasible way.

Firstly, it was defined the type of testing scenarios that would be presented. For that decision, it was taken into account the specific aspects that should be improved in the interface usability. As mentioned, not only the optimization of the efficiency, effectiveness, learnability, and user satisfaction of the entire query formulation process was the goal, but also the improvement of the query comprehension. In such a way that it was important to evaluate if users could understand the query purpose (i.e., what data intends to be fetched from the database) as well as the time they required to realize that.

Considering those evaluation requirements, two types of testing scenarios were prepared: scenarios where users explore an existing query and try to realize what was its purpose, and the other ones where users try to formulate it on their own. Through that approach, it was possible to analyze the usability of the interfaces tested for both points of view: comprehension and formulation.

Nevertheless, the complexity of the queries presents a crucial factor when the scenarios were devised. For example, an interface could be useful and pleasant to use in simple use cases but it could lead to a decrease of that quality in more complex queries. Accordingly, the requirements that were considered relevant to be covered by user testing scenarios are listed below:

- **Query Comprehension:** Relevant aspects that should be present in the queries and consequently in the interface to evaluate the queries' comprehension:
  - **Interface Elements Exploration:** Include use cases that contain the majority of the query components supported by the system: database entities and joins of different types, filtering and sorting criteria applied to different data types, and other query components added throughout the query formulation process, such as Group Bys, Aggregation Functions (SUM, MIN, MAX, AVG, COUNT) or Calculated Attributes [1];
  - **Joins Representation:** Representation of different joins in order to analyze if users could successfully identify them. First of all, there was a concern to consider in the scenarios joins of different types, such as inner joins or left joins. It was important to consider not only the most common joins (i.e., joins where the unique foreign key referencing another table is equals to the other table primary key) but also some queries that contain joins with more advanced conditions. For instance, when the join between two tables could be made using different foreign keys, as they are multiple relationships between both tables, or when the join condition contains logical operators.
- **Query Formulation and Modification:** At the same time, the following aspects were considered essential to be approached in the user testing scenarios from a query formulation or modification point of view:
  - **Add Data Sources and Joins:** Identify the options chosen to add query sources and analyze what are the users' reactions to the system automatisms to simplify the joins specification;
  - **Edit Query Filters:** Evaluate if the query filters edition is intuitive and what barriers could exist in the interfaces regarding this aspect;

[1] These operations were presented in section 2.2.2.2

  - **Insert Calculated Attributes, Group Bys, and Aggregation Functions:** Check if users could understand the cases where they would need to use the referred functionalities and if they can discover how to apply them without difficulty;
  - **Use Hidden Columns:** Verify the main difficulties identified by users when they need to apply actions in attributes that were not entirely visible in the interface, either because they are hidden, or because they are not visible due to the lack of available space in the interface (scroll required).

The aspects mentioned were considered the most relevant to analyze since they allow a widespread use of the tool but they also cover multiple cases identified as critical in terms of usability, according to the aspects further detailed in section 5.1.

Nevertheless, the data model could have also a significant impact on user testing results. For instance, if a user does not understand the data model, he could miss the purpose of the query, even if the query is presented in a simple and readable manner. In that way, the selection of the data model used for user testing was performed attending to the following points:

- **Simple Business Domain:** The data stored in the database should represent an example of a day-to-day application that could be understandable by any user regardless of his background.
- **Different data types:** The data model should contain a wide variety of data types since it could become useful in order to reflect if the design approaches chosen are able to work with different types of data;
- **Multiple relationships between two entities:** The interface aims to accelerate and simplify the querying process not only for simple cases. For this reason, it is important to perceive how the interface could help users in cases where there are more than one relationship between two entities. Moreover, SQL does not have any particular syntax that helps users in these cases, then there is an improvement opportunity here.

Accordingly, 4.1 illustrates the data model of the database adopted to perform all usability tests.

After choosing the data model used as support for the usability tests, the list of test scenarios was elaborated. Three different types of scenarios were designed:

- **Query Comprehension:** The user explores a visual query already built and tries to indicate what are the query components presented as well as the data that would be fetched from the database through that query;
- **Query Modification:** After a query comprehension example, the user tries to apply some modification on the existing query previously explored;

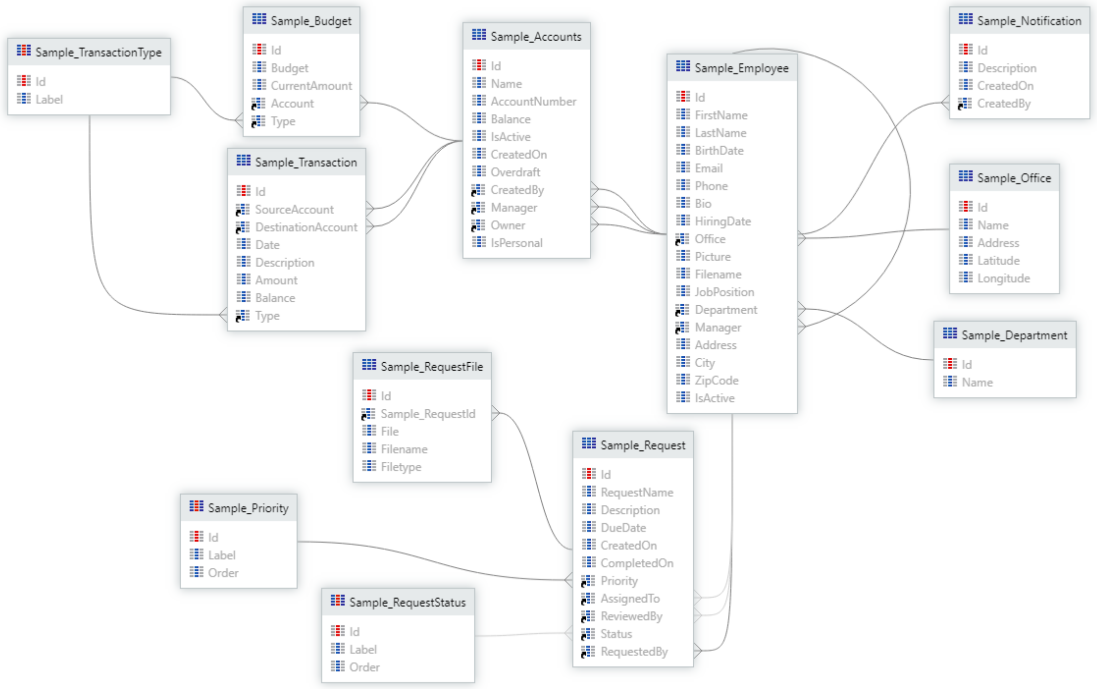


Figure 4.1: Data Model used for User Testing

- **Query Formulation:** Given a natural language statement that explains what data is intended to be fetched from the database, the user tries to formulate a new visual query from scratch in order to retrieve the data intended.

Through that approach, it was possible to keep the focus on different aspects according to the scenario used. On the one hand, the comprehension scenarios give the focus to the query readability, which promote a global exploration of the query components and gave the opportunity to understand if users clearly identified the purpose of each interface section. On the other hand, modification and formulation scenarios were used to understand if users could built queries through the interface presented.

In that way, the scenarios below, built to fetch data to the data model presented in Figure 4.1, were presented to the user in this order:

1. **Query Comprehension 1 (C1):** Employees of "Portugal" or "Japan" of departments "Services Support West" or "Services Support East" who have a job position different from "Services Representative" and never have created any notification. The employees must be presented ordered by their last name (descending order);

2. **Query Modification 1 (M1):** Change the existing query to consider only the employees of offices "Australia"or "Japan", their department must be "Marketing"or "Services Support East", and they must have any job position. Moreover, create an attribute to present employees' full names;

3. **Query Comprehension 2 (C2):** List for each AccountNumber the amount sum of transactions of type "Eating Out". Moreover, the transaction is only shown if its source account is managed by employees of department "Credit Control" and office "United Kingdom". Account Numbers are sorted by their amount sum in descending order;

4. **Query Comprehension 3 (C3):** List the number of employees by department and office. The number of employees is presented in descending order;

5. **Query Formulation 1 (F1):** Notifications created by employees who are owners of a set of accounts which, at least, must combinedly average a balance of 20.000 (average of balances of accounts owned by each employee);

6. **Query Comprehension 4 (C4):** List all requests assigned to employees of department "Services Support West" ordered by priority in the first place ("High" first), and secondly by creation date (oldest dates first);

7. **Query Modification 4 (M4):** Considering the previous requests, change the query to show only the ones that were requested by employees from other departments (not the same).

Nevertheless, there were several results to be obtained from the user testing scenarios and it was difficult to reach an approach to test all aspects related to the existing problems in a short period. Accordingly, Table 4.2 was used to distribute the requirements mentioned across all scenarios, without turning the usability test exhaustive and heavy for users.

Table 4.2 clarifications regarding the factors integrated in testing scenarios:

- **Simple Joins:** Joins that are automatically generated by the system without requiring human intervention;

- **Complex Joins:** Joins that need to be partially configured manually (e.g., to specify the foreign key used to merge both tables or to change the join condition);

- **Left Join with Null:** Left join between entities A and B, to consider only the entities A that are not related to B (e.g., the employees who have not created any notification.);

- **Group by (without reference):** The system has an automatism that generates automatically a name to a new attribute grouped by. However, this name is not self-explanatory and there was no reference to the source of its attribute. That way, it is important to highlight this case;

- **Aggregation Functions:** The aggregation functions supported by aggregates are Max, Min, Average, Sum, and Count. As the representation in the interface as well as the insertion method is similar, only a few of these were used;

Table 4.2: Distribution of the relevant testing aspects among the testing scenarios designed.

Scenarios		**C1**	**M1**	**C2**	**C3**	**F1**	**C4**	**M4**
Relevant for Comprehension and Formulation	Number of Entities	4	4	6	3	3	5	7
	Simple Joins	2	2	4	2	1	3	3
	Complex Joins	-	-	1	-	1	1	3
	Left Join with Null	1	1	-	-	-	-	-
	Filters	3	2	3	-	-	1	2
	Group Filters	-	-	-	-	1	-	-
	Sorting (Text)	1	1	-	-	-	1	1
	Sorting (Number)	-	-	1	1	-	-	-
	Sorting (Date)	-	-	-	-	-	1	1
	Group By	-	-	1	2	4	-	-
	Group By (without reference)	-	-	-	3	-	-	-
	Max	-	-	-	-	-	-	-
	Min	-	-	-	-	-	-	-
	Average	-	-	-	-	1	-	-
	Sum	-	-	1	-	-	-	-
	Count	-	-	-	1	-	-	-
Relevant for Query Formulation or Modification	Use of not visible columns					X		
	Use of hidden columns					X		
	Insert Calculated Attribute		X					
	Add Aggregation Function					X		
	Add not automatic join					X		X
	Add same entity twice (alias)							X
	Edit some filters		X					

Due to the complexity of each scenario, the last two scenarios (Comprehension 4 and Modification 4) were not tested with Citizen Developers, those users were only tested in 5 scenarios while the other user types were tested in 7.

4.5 Evaluation Method

Having all testing scenarios established, it was necessary to plan how to identify the users, which enables an accurate evaluation of the interaction with the system, allowing the gathering of qualitative and quantitative results throughout the usability tests.

Figure 4.2 presents an overview of the evaluation method adopted which have 4 steps:

1. **Understaning the user's profile:** User's answer a survey with questions regarding

their background in order to identify the users' profile according to the user groups previously defined;

2. **Preparation of the test:** There is an explanation of the test to clarify users the purpose of the test is only to evaluate the interface and not their capabilities. After that, the data model is presented since it is necessary to test the interface according to the testing scenarios established;

3. **Observation during the test:** While users were interacting with the interface to accomplish the tasks proposed, their user's reactions were registered as well as difficulties they felt and other usability evaluation points important to classify the usability of the interface;

4. **Results:** The information obtained during the test was processed in order to analyze the usability of the interface tested.

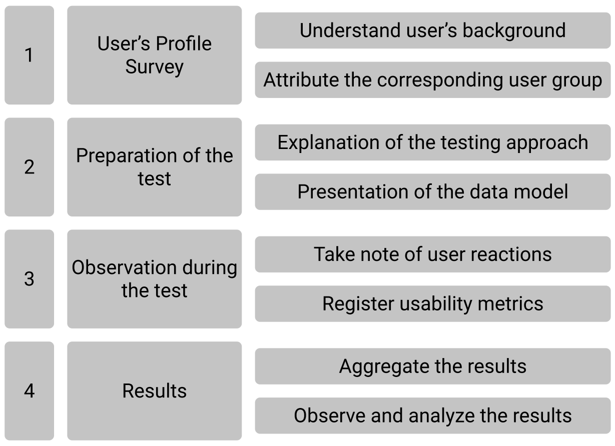

1
User's Profile Survey
Understand user's background
Attribute the corresponding user group
2
Preparation of the test
Explanation of the testing approach
Presentation of the data model
3
Observation during the test
Take note of user reactions
Register usability metrics
4
Results
Aggregate the results
Observe and analyze the results

Figure 4.2: Evaluation method overview.

Understanding the user's profile:

All users tested, started by filling a survey, which included the questions presented in Table 4.3, in order to perceive what is their background regarding software development, relational databases, and data management and visualization tools.

The answers to these questions were used to identify the profile of each user according to the three user groups defined:

- **Citizen Developer:** users who do not have extensive software development background neither using traditional programming languages nor with low-code platforms;

- **Software Developer:** users experienced using traditional programming languages but not accustomed to use low-code development solutions;

Table 4.3: Survey to undertand users' profile

ID	Question	Possible answers				
1	What is your professional occupation?	Open question				
2	Have you a degree in Computer Science or similar?	Yes	No			
3	Do you have other academic backgrounds?	Open question				
4	Have you already used OutSystems?	Never use	Almost never	Occasionally / Sometimes	Almost every time	Frequently use
5	How long do you use the platform?	No experience	<= 6 months	<= 1 year	1-3 years	>4 years
6	Have you used a query language (SQL or other)?	Never use	Almost never	Occasionally / Sometimes	Almost every time	Frequently use
7	When was the last time that you have used SQL to build queries?	Never use	Some weeks ago	Some months ago	Last year	Some years ago
8	From 1 to 5, how do you define your SQL expertise level?	1	2	3	4	5
9	Have you already used relational databases?	Never use	Almost never	Occasionally / Sometimes	Almost every time	Frequently use
10	Are you familiar with relational operators (joins)?	Yes	No			
11	How often do you use spreadsheet applications?	Never use	Almost never	Occasionally / Sometimes	Almost every time	Frequently use
12	How often do you use business intelligence software?	Never use	Almost never	Occasionally / Sometimes	Almost every time	Frequently use

- **OutSystems Developer:** users who have experience developing software with OutSystems, independently of their background.

More details regarding the user groups definition will be further provided in section 5.2.2.

Preparation of the test:

After perceiving the user's profile, the data model was presented, focusing on the most relevant aspects of the tests. In that way, it was explained how the entities of the model were related with each other and the most used attributes in the scenarios were highlighted.

Users observed the database diagram illustrated in Figure 4.1 and a presentation of each entity was provided through a dialogue where the focus was the explanation of the entities included in the data model and how they are related (i.e., explaining which foreign keys were used to link the entities presented). The language used was adapted according to the users' background regarding relational databases. For users who have previous knowledge regarding join operations, it was reinforced the foreign keys used in each join. On the other hand, the explanation to users who did not have previous knowledge in this subject was simplified, explaining for other words what meant the links between entities, providing whenever necessary examples to ensure users understood the data model adopted for tests. As soon as users confirmed that they understood the overview of the existing entities, the testing process started.

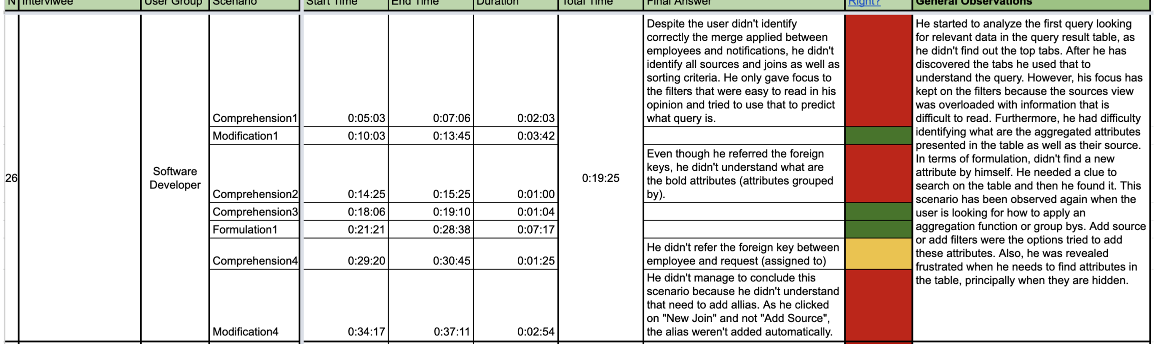
Observation during the test:

While users were interacting with the system in order to accomplish the proposed scenarios, it was observed the approaches used to solve the problems and the difficulties felt.

However, following users' reasoning and opinions while they interact with the interface was not considered sufficient to collect all necessary results. Therefore, an evaluation methodology was designed to collect the required data using the same approach for all tested interfaces. In that way, Table 4.4 enumerates all aspected registered and evaluated in usability tests for each scenario. The aspects listed in this table were registered manually inserted the observation results in a spreadsheet as exemplified in Figure 4.3.

For all scenarios, it was registered the time they needed to perform it as well as if they reached the task goal according to the classification presented in Table 4.5.

In addition, a text highlighting the most relevant usability test topics, including users' opinions, reactions, expectations, and suggestions has been included for each user. It was registered the qualitative result of the usability tests, which are not only important for the design of the next iterations but also to assess users' satisfaction.



N	Interviwee	User Group	Scenario	Start Time	End Time	Duration	Total Time	Final Answer	Completely Right?	General Observations
26		Software Developer	Comprehension1	0:05:03	0:07:06	0:02:03	0:19:25	Despite the user didn't identify correctly the merge applied between employees and notifications, he didn't identify all sources and joins as well as sorting criteria. He only gave focus to the filters that were easy to read in his opinion and tried to use that to predict what query is.	🟥	He started to analyze the first query looking for relevant data in the query result table, as he didn't find out the top tabs. After he has discovered the tabs he used that to understand the query. However, his focus has kept on the filters because the sources view was overloaded with information that is difficult to read. Furthermore, he had difficulty identifying what are the aggregated attributes presented in the table as well as their source. In terms of formulation, didn't find a new attribute by himself. He needed a clue to search on the table and then he found it. This scenario has been observed again when the user is looking for how to apply an aggregation function or group bys. Add source or add filters were the options tried to add these attributes. Also, he was revealed frustrated when he needs to find attributes in the table, principally when they are hidden.
			Modification1	0:10:03	0:13:45	0:03:42			🟩	
			Comprehension2	0:14:25	0:15:25	0:01:00		Even though he referred the foreign keys, he didn't understand what are the bold attributes (attributes grouped by).	🟥	
			Comprehension3	0:18:06	0:19:10	0:01:04			🟩	
			Formulation1	0:21:21	0:28:38	0:07:17			🟩	
			Comprehension4	0:29:20	0:30:45	0:01:25		He didn't refer the foreign key between employee and request (assigned to)	🟨	
			Modification4	0:34:17	0:37:11	0:02:54		He didn't manage to conclude this scenario because he didn't understand that need to add allias. As he clicked on "New Join" and not "Add Source", the alias weren't added automatically.	🟥	

Figure 4.3: Example of a general annotation registered after a usability test.

Table 4.4: Evaluation points registered in each scenario.

ID	Scenario	C1	M1	C2	C3	F1	C4	M4
	General observation aspects							
S1	Task completion	X	X	X	X	X	X	X
S2	Time used to perform the scenario	X	X	X	X	X	X	X
S3	Components of the interface explored	X	X	X	X	X	X	X
	Regarding comprehension scenarios							
S4	First area of the interface used to explore the query	X		X	X		X	
S5	List of the entities identified	X		X	X		X	
S6	List of the joins identified	X		X	X		X	
S7	In the cases the join is a left join with a condition where the right entity identifier must be null	X						
S8	In the cases the join between two entities could be made using different foreign keys, identification if user identified correctly the foreign key used			X			X	
S9	List of the filters identified	X		X			X	
S10	List of the sorting criteria identified	X		X	X		X	
	Regarding modification scenarios							
S11	Task completion of the required modification of the existing filters		X					
S12	Time needed to perform the required modification of the existing filters		X					
S13	Registered if users made mistaked when they were changing the required filters		X					
S14	Task completion of the insertion of the required calculated attribute		X					
S15	Time spent to add the required calculated attribute		X					
S16	Option used to add the required calculated attribute		X					
S17	Alternative options tried in order to find how to add the required calculated attribute		X					

Table 4.4 continued from previous page

ID	Scenario	C1	M1	C2	C3	F1	C4	M4
S18	Registered if users had difficulties to find how to add the required calculated attribute		X					
S19	Registered if users needed a clue in order to find how to add the required calculated attribute		X					
	Regarding formulation scenario							
S20	List of entities inserted					X		
S21	List of joins inserted					X		
S22	Selection of the required foreign key in the cases where the join between two entities could be made using different foreign keys					X		
S23	Registered if users managed to add the required group filter					X		
S24	Task completion of the addition of the required aggregated attributes					X		
S25	Time needed to add the required aggregated attributes					X		
S26	Option used to add the required aggregated attributes					X		
S27	Alternative options tried in order to find how to add the required aggregated attributes					X		
S28	Registered if users had difficulties to find how to add the required aggregated attributes					X		
S29	Registered if users needed a clue in order to find how to add the required aggregated attributes					X		

Evaluation of the users' satisfaction after using the interface:

Finally, to gain a more extensive notion and some quantitative metrics about the interaction with the interfaces, the users were asked to fill in a System Usability Scale (SUS) [73] as their final task in the usability test, where users needed to classify the questions bellow with a number from 1 (strongly disagree) to 5 (strongly agree):

Table 4.5: Description of the effectiveness states considered.

Legend	Comprehension	Modification	Formulation
Achieved	When the user mentioned all components of the queries (sources, joins with foreign keys, filters, sorting, aggregation functions). The only thing that would not be considered is if the user didn't refer with or without joins when they were put automatically.	All modifications	Completely Right
Partially Achieved	If the user didn't refer to the foreign key between two tables where there is more than one foreign key between two tables. The another possibility is if the user did not specify only the sorting criteria.	Forgot to remove some filter	Group data using the wrong identifier
Not Achieved	Anything else	Anything else	Anything else

1. I think that I would like to use this system frequently;

2. I found the system unnecessarily complex;

3. I thought the system was easy to use;

4. I think that I would need the support of a technical person to be able to use this system;

5. I found the various functions in this system were well integrated;

6. I thought there was too much inconsistency in this system;

7. I would imagine that most people would learn to use this system very quickly;

8. I found the system very cumbersome to use;

9. I felt very confident using the system;

10. I needed to learn a lot of things before I could get going with this system.

The process detailed in this chapter was used in all interfaces tested and addressed in this dissertation: the existing interface of Aggregates, the paper prototype, and the final prototype which was integrated into Service Studio (the Visual IDE of the OutSystems Platform).

4.6 Summary

Figure 4.4 summarizes the methodology described in this chapter, from the problem definition to the final prototype implemented. As can be observed, there were two major phases throughout this dissertation:

- **Requirements and Analysis:** The stage where the problem was defined as well as the target users of the system were identified and classified into different groups according to their requirements and expectations. The output of these two analyses was taken into account to prepare the user testing environment so that the scenarios and the evaluation method were defined;
- **Design and Implementation:** The process to reach the redesigned interface, from the first sketchings to the final prototype integrated into the Service Studio code. The results of each phase were taken into account in the following phase, in order to evaluate the design choices made gradually.

Finally, the results of the user testing of the final prototype were compared against the results obtained when users tested the existing interface in order to measure the success of the prototype built.

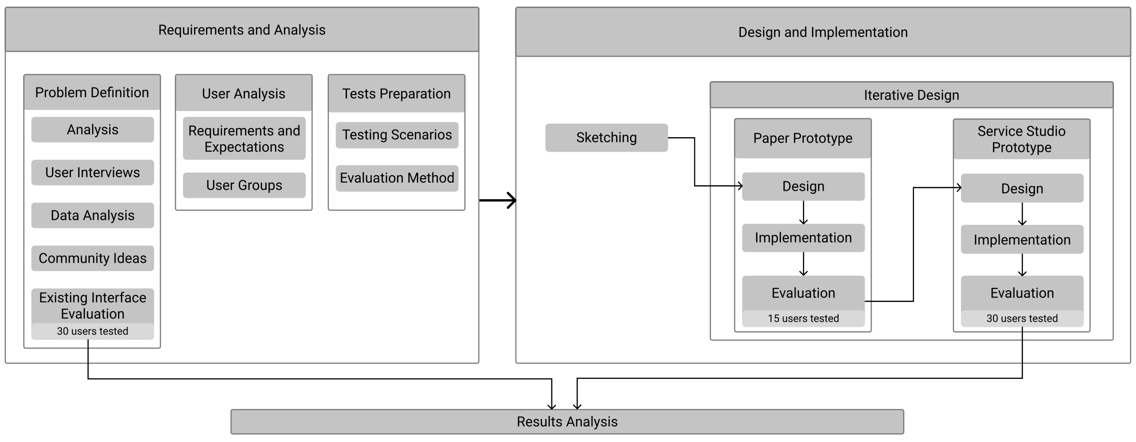


Figure 4.4: Methodology Overview

Chapter 5

Requirements and Analysis

As previously stated, the problem comprehension and the users' requirements were considered key points throughout the design process in order to build a user-centered solution that could tackle the existing most impactful problems. For this to happen, this chapter describes the processes used to structure the problems of the current visual interface, using the methodology described in the previous chapter. Furthermore, target users of the system are presented and categorized, accordingly with their background, expectations, and necessities.

5.1 Problem Analysis

As the existing VQI of the OutSystems Platform has not been having the expected acceptability due to its interaction problems, multiple approaches were used to collect the existing problems, understanding for each of them what are the tasks involved, and the most harmed set of users.

Hereinafter, the approaches and strategies used to collect and organize data regarding the existing problems of the interface will be presented. As usability issues depend on interaction with users, the problems were holistically analyzed, considering, in each problem, the impact caused on each type of user.

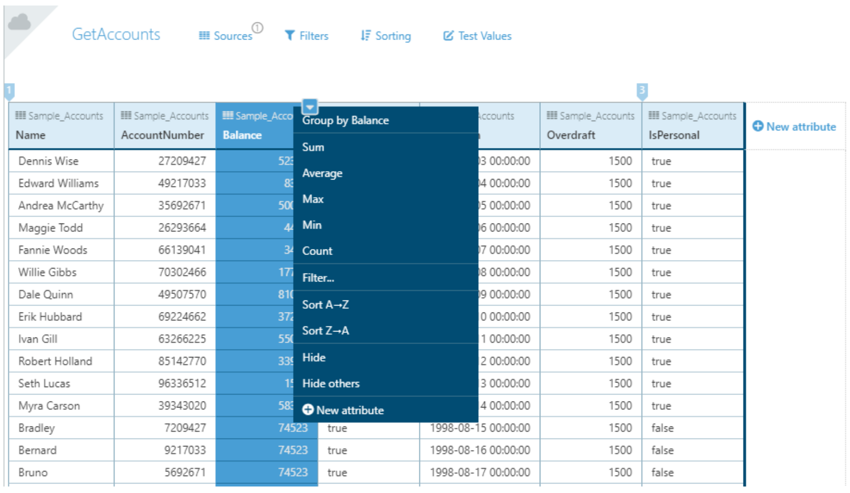
5.1.1 Analysis

The analysis of the query formulation interface through self-exploration was the first method used to comprehend the existing problems. The process started with the visualization of two OutSystems tutorials [60, 61] about visual data querying. The tutorials included some hands-on exploration that provided an initial contextualization of the visual query builder and its functionalities. After that, other scenarios of query formulation were explored to comprehend the system barriers and difficulties.

As pointed out in section 1.2, the lack of some advanced functionalities were observed during that exploration process. Nevertheless, other problems, which have a negative effect on user experience and task efficiency and effectiveness, were also identified.

First of all, it was detected that some functionalities were hidden, damaging the learnability of the system, not fulfilling the "Recognition rather than recall"principle of the Nielsen Heuristics [57]. Notwithstanding that the learnability issue mainly affects the novice users, there are peculiarities of this system and its environment that aggravate this problem. Since SQL formulation is an alternative approach to build queries, if SQL users do not find the intended functionalities in the visual query builder due to its hiddenness, they could use SQL to perform their tasks, avoiding the use of the visual system.

The hidden functionalities are not only advanced features of query formulation. For instance, there is no visible option in the existing interface to add an aggregation function, such as Group By, SUM, MIN, MAX, AVERAGE, or COUNT. These functions are accessible for an exclusive interaction path which requires a right-click on the column header of the attribute where the user intends to apply the aggregation function. Figure 5.1 illustrates an example of the application of an aggregation function. Other options are not always hidden but are not prepared to keep visible due to interface components modification. For example, despite the option to add calculated attributes is visible (after all columns of the query result table), if the query result has several columns it is necessary to scroll horizontally until the end to find that option.



Sample_Accounts Name	Sample_Accounts AccountNumber	Sample_Acco Balance		ccounts	Sample_Accounts Overdraft	Sample_Accounts IsPersonal
Dennis Wise	27209427	523		3 00:00:00	1500	true
Edward Williams	49217033	83		4 00:00:00	1500	true
Andrea McCarthy	35692671	500		5 00:00:00	1500	true
Maggie Todd	26293664	44		6 00:00:00	1500	true
Fannie Woods	66139041	34		7 00:00:00	1500	true
Willie Gibbs	70302466	177		8 00:00:00	1500	true
Dale Quinn	49507570	810		9 00:00:00	1500	true
Erik Hubbard	69224662	372		0 00:00:00	1500	true
Ivan Gill	63266225	550		1 00:00:00	1500	true
Robert Holland	85142770	339		2 00:00:00	1500	true
Seth Lucas	96336512	15		3 00:00:00	1500	true
Myra Carson	39343020	583		4 00:00:00	1500	true
Bradley	7209427	74523	true	1998-08-15 00:00:00	1500	false
Bernard	9217033	74523	true	1998-08-16 00:00:00	1500	false
Bruno	5692671	74523	true	1998-08-17 00:00:00	1500	false

Figure 5.1: Hidden option to add an aggregation function, since it is enclosed in a right-click on the query result table column header.

Furthermore, the interface has not revealed to be flexible, efficient, and effective for professional users' purposes. Obstacles have been identified as preventing users from accelerating their query formulation process and causing increases in the queries' error rates. Some examples of these aspects which were identified through the interface's analysis are presented, hindering query formulation and query comprehension, and reducing the most valued proposition of the data querying visual approach:

- Search engines: there is no option to efficiently search for an attribute neither to look for some data in the query result or to apply some aggregation function (as illustrated in Figure 5.1). Moreover, there is no general search in all queries, which could be useful when users need to find if some entity or variable is present in the query.

- Hidden columns: in the existing interface, primary and foreign keys are automatically hidden in the query result table since they are considered not relevant data for the query output. That strategy has been implemented since the first release of the interface to provide the most clear output for users with lack of technical background, assuming that they could not understand the meaning of those attributes. Since those attributes may contain relevant information, it is possible to unhide them. However, it must be highlighted that there is no efficient way to expand all hidden columns at once, difficulting significantly the comprehension and formulation of all queries who do not have a reduced number of entities and attributes. Figure 5.2 illustrates how the hidden attributes are presented in the interface.

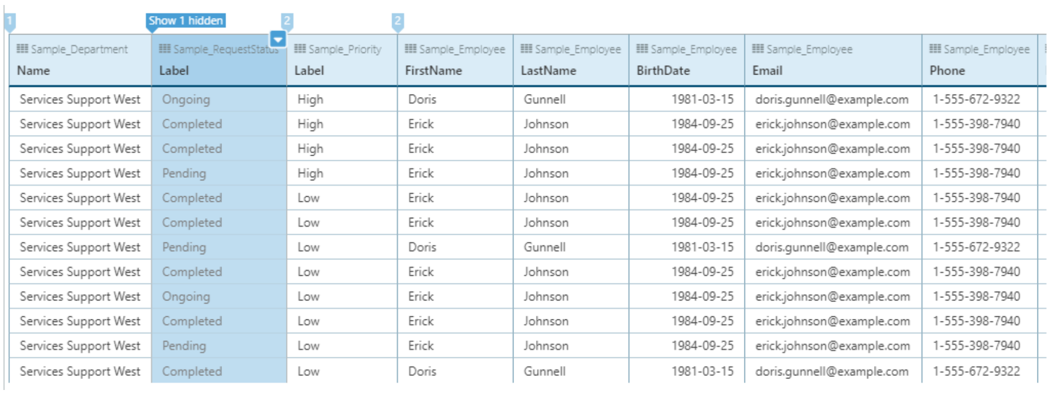
- Filter edition: the adaptability of existing query filters is possible using the expression editor modal, as demonstrated in Figure 5.3. This modal is useful because it provides some guidance in the expression formulation. Users, can select the intended language expressions or the entities, attributes, or variables that need without having to recall those names. However, if users do not need that assistance, the interaction strategy implemented could led to time and visual context wastefulness every time that a modal is opened. It should be noted that the modal is definitely a good feature but it should not be the unique way to edit filters since it can increase the time spent to built queries unnecessarily. Moreover, there is no way to copy and past filter or to read effectively the content since there is no color highlighting in filters' expressions.

- Accelerators feedback: this visual query interface has multiple accelerators that turn the query building process faster. Nevertheless, sometimes the automatic mechanisms are silent and users may not comprehend they were applied. For example, when two entities are added and are related, the system automatically joins those entities. However, there is no highlight or other visual interaction mechanism that provides feedback to the user and shows him which action was applied in the aggregation output.

- Inconsistency problems: some functionalities were accessible through a determined context and option but other alternative options do not have the same behavior. For instance, in some use cases it is not possible to add a new entity to the query using drag and drop but if the user click in another add button the entity is added.

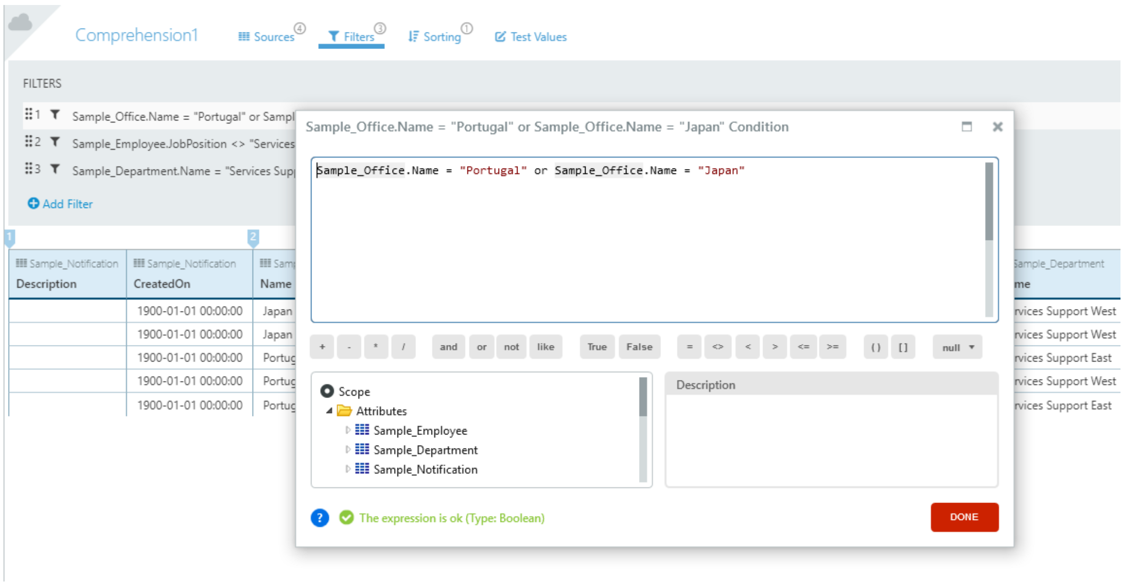
- Multiple actions at once: sometimes it could be useful to do several actions at once in order to reduce the time spent to build the query. For instance, it should be possible to add multiple entities at once instead of adding one at a time;

In a nutshell, the first analysis and exploration of the interface lead to conclude that the interface is useful for users without technical background and has implemented good strategies that could optimize the query formulation process even more than SQL if those strategies would be reconsidered and redesigned. Moreover, it was concluded that it is necessary to provide search engines and other accelerators to enhance the efficiency value proposition of the system. The value of the system increases if users could build queries in that system as fast as SQL or even more.



Sample_Department Name	Sample_RequestStatus Label	Sample_Priority Label	Sample_Employee FirstName	Sample_Employee LastName	Sample_Employee BirthDate	Sample_Employee Email	Sample_Employee Phone
Services Support West	Ongoing	High	Doris	Gunnell	1981-03-15	doris.gunnell@example.com	1-555-672-9322
Services Support West	Completed	High	Erick	Johnson	1984-09-25	erick.johnson@example.com	1-555-398-7940
Services Support West	Completed	High	Erick	Johnson	1984-09-25	erick.johnson@example.com	1-555-398-7940
Services Support West	Pending	High	Erick	Johnson	1984-09-25	erick.johnson@example.com	1-555-398-7940
Services Support West	Completed	Low	Erick	Johnson	1984-09-25	erick.johnson@example.com	1-555-398-7940
Services Support West	Completed	Low	Erick	Johnson	1984-09-25	erick.johnson@example.com	1-555-398-7940
Services Support West	Pending	Low	Doris	Gunnell	1981-03-15	doris.gunnell@example.com	1-555-672-9322
Services Support West	Completed	Low	Erick	Johnson	1984-09-25	erick.johnson@example.com	1-555-398-7940
Services Support West	Ongoing	Low	Erick	Johnson	1984-09-25	erick.johnson@example.com	1-555-398-7940
Services Support West	Completed	Low	Erick	Johnson	1984-09-25	erick.johnson@example.com	1-555-398-7940
Services Support West	Pending	Low	Erick	Johnson	1984-09-25	erick.johnson@example.com	1-555-398-7940
Services Support West	Completed	Low	Doris	Gunnell	1981-03-15	doris.gunnell@example.com	1-555-672-9322

Figure 5.2: Hidden attributes - primary and foreign keys are hidden by default.



FILTERS

1 Sample_Office.Name = "Portugal" or Sampl
2 Sample_Employee.JobPosition <> "Services
3 Sample_Department.Name = "Services Sup

Sample_Office.Name = "Portugal" or Sample_Office.Name = "Japan" Condition

```
Sample_Office.Name = "Portugal" or Sample_Office.Name = "Japan"
```

The expression is ok (Type: Boolean)

Sample_Notification Description	Sample_Notification CreatedOn	Sam Name
	1900-01-01 00:00:00	Japan
	1900-01-01 00:00:00	Japan
	1900-01-01 00:00:00	Portug
	1900-01-01 00:00:00	Portug
	1900-01-01 00:00:00	Portug

ample_Department me
rvices Support West
rvices Support West
rvices Support East
rvices Support West
rvices Support East

Figure 5.3: Filter edition modal - example of a filter edition after selection of the intended filter (the first one in that case).

5.1.2 User Interviews

After the analysis and study of the existing interface, there was a necessity to realize how are the usage of the visual query system for users. There was a demand to comprehend what are the most impacting problems of the interface for users' tasks, what are the first users' reactions when asked about the utility of that interface, and if it could be applied, what are the reasons to use SQL instead of OutSystems' visual query builder. Asking those general questions to users was the technique used to understand what is their in-depth and sincere opinion about the system utility and its most impacting problems.

Therefore, 10 user interviews were performed to explore the tool's limits and understand their impact on user actions. The interviews started with some brief questions to perceive the background of the participant. After that, more directed questions were asked in order to understand the users' opinions about the advantages and disadvantages of the visual query tool. Besides, it was asked in which situations users prefer to use SQL instead of the visual tool. The answer reveled to be important since it provided information about the main reasons to resort on SQL alternative and also the main causes which led the user to stop using the visual query builder interface. Participants identified and demonstrated in-loco examples of limitations of the visual query system, highlighting causes and the impact of the problems. Also, users revealed other problems which have not been already identified.

Furthermore, the set of interviewed users, answered about which advanced functionalities would benefit further adoption and some usability problems previously identified. The purpose of these last questions was to ensure that the user answered truthfully about the aspects that most affect him negatively while using the visual query system. By letting him explain all the details about his vison, on further adjustments to the system, is possible to establish aspects of common groud between the conclusions of the primary analysis and the user reasoning about major limitations.

The results revealed that the most novice users, the users with less than six months of experience, consider Aggregates simple to use and stated that it covered their necessities, referring that it is more simple to learn than SQL. The most experienced users, who use the OutSystems platform to develop applications, every day, for professional purposes, and have a technical technological background, reported that in the visual tool they cannot have a clear query understanding at a first glance. Moreover, they referred that they work with queries that contain several tables, attributes, and business rules. In those cases, they have considered that it was difficult to formulate queries using the visual querying interface. Besides, being required to switch between tabs in the interface to view the data sources, and the filters and sorts criteria, other issues were presented such as the lack of control on the query output[1]. When asked about the aggregation functions[2], they do not refer any problem with the approach interaction strategy adopted, but they

[1]As referred on section 2.2.2.2. When a user adds an entity to an Aggregate, all its attributes are added automatically and if the user hides them the output of the query does not change.

[2]Funtionality added when Simple Queries have been replaced by Aggregates (section 2.2.2.1)

have emphasized that it is very difficult to find the intended column they need, since there is no search or navigation engine. Other usability problems that decreases the users' satisfaction, such as difficulty to search in all query or to copy and paste query components, were also pointed out.

In conclusion, the results of the problem definition phase indicated that the interface is simple to use but presents a considerable set of limitations in its components, mainly in the domain of professional purpose tasks. In that way, the suggestions to create an improved overview of the query, which would allow a faster comprehension of the query, as well as the implementation of interface accelerators to make the query editor powerful such as search engines and several access alternatives to the same functionality were the most stated aspects. Regardless of the suggestions, the user experience of the interface should be kept simple in order to continue to be easy to be learned and used by users without a technical technological background.

5.1.3 Data Analysis

Even though the performed analysis and the first user interviews have indicated user experience problems of the interface as the factor that has been impacting the user acceptance of the visual query builder, a quantitative analysis was performed to perceive what are the operations most used by developers when they formulate queries textually. Thereby, the analysis was complemented with a metric study on queries executed on the Outsystems cloud, in order to find patterns that could justify users reasons to use SQL instead of Aggregates.

The queries analyzed, which have been extracted in July 2019 from customers' projects, were built using Advanced Queries[3]. The data set used is composed of 214.400 statements. However, only 60.8% were used in this study since only the queries are important for the results and not other SQL statements, such as inserts, updates, deletes, and transactions. Nevertheless, that set of 125.613 queries has duplicated results, so that these ones were removed resulting in a final data set of 67.828 queries. The operators and clauses were identified using a SQL Parser developed in JavaScript [50]. After obtaining the abstract syntax trees of the queries in a JSON file, that data was analyzed in a program to count the operators and the clauses that were present in the queries.

Firstly, it was measured the percentage of SQL queries that contained operations not supported by the visual tool. Table 5.1 summarizes the number and the percentage of those queries containing not supported operations. It is important to refer that the intersection of the subsets is not null, so there are queries that have two or more of the indicated operations. Nevertheless, it can be observed that most of the operations have no significant representation in the results. For example, the DISTINCT operation was the

[3]The option of the OutSystems Platform, invoked in section 2.2.2.1, that allows the query design in a textual way using a language based on SQL.

Table 5.1: Queries that contain operations not supported by Aggregates

	IN	NOT IN	EXIST	NOT EXIST	UNIONS	DISTINCT	SUBQUERIES	Total
Queries	7538	2697	132	118	2385	7987	6827	67828
Percentage	11.11%	3.98%	0.19%	0.17%	3.52%	11.78%	10.07%	100%

Table 5.2: Queries that could be designed using Aggregates and the queries which the tool does not support

	Not Supported	Supported (Simpler)	Supported (More Complex)	Total
Queries	28130	24026	15672	67828
Percentage	41.5%	35.4%	23.1%	100%

not supported operation with the highest percentage, included in 11.78% of the queries analyzed.

Secondly, it was measured how many queries were built using the textual language but could be designed using the visual query builder. Table 5.2 shows the results obtained separating the queries performed in three categories:

- Not Supported: Queries which include operations not supported by Aggregates, such as IN, NOT IN, EXIST, NOT EXIST, Unions, Distincts and Subqueries;

- Supported by Aggregates: Queries that could be designed totally using Aggregates. These are divided into two subcategories:

  - Simpler: Queries which include only operations supported by aggregates excluding the indication of sorting criteria and the use of aggregation functions (e.g. GROUP BY or SUM, AVG, MIN, MAX, COUNT);

  - More Complex: Queries that are supported by Aggregates excluding the above (simplers).

User interviews suggested that aggregation functions and sorting criteria were not the main problems. Accordingly, the queries supported by Aggregates were divided into two groups in order to compare the quantitative analysis with the qualitative analysis extracted in interviews. This was considered an important element since these operations could be obtained using a different interaction technique where the user changes the query when he is interacting with the query result, as mentioned in section 2.2.2.2.

In the set of queries analyzed, around 58.5% could be designed using Aggregates, is evident that the lack of support of some SQL expressions might not be the main problem. Under the circumstances, the results were discussed together with the stakeholders. It was determined that the main problem was the usability of the system.

In conclusion, the results of the quantitative analysis have confirmed the assumptions pointed out after the analysis and exploration of the interface and the first user interviews. All the studies made have concluded that the main priority of the project should be the usability improvement of the visual querying tool interface. Thereby, there are metrics that sustain the conclusions made in the problem definition phase.

5.1.4 Community Ideas

Since OutSystems has a wide worldwide Community of developers, the users can use the OutSystems Community Website [64] to express their problems or difficulties or even to communicate new ideas for the product. Through that website, it was possible to extract information in order to align the final solution with the ideas and problems they shared. Accordingly, all 306 posts of category "Aggregates & Queries"were analyzed in order to extract useful information for the design phase of this dissertation.

First of all, it has been taken into account for each post if the topic is related to the lack of some functionalities or related to some problems in the interface. That approach leads to perceive that the predominance of the posts were about usability problems of the interface, whereas there are a reduce quantity of posts about functionalities not supported.

Therefore, the users' problems and suggestions regarding usability or enhancements of the interface were processed, transforming that information to relevant input to the next design phases. The problems presented below were the most important problems and suggestions indicated in the OutSystems Community:

- Search engines: users requested in multiple posts alternatives to search for an entity, attribute, or filter inside the query. Not only they refer that the readability in the interface is difficult primarily due to the non-existing color highlight in the text but also there is no possibility to search for the intended fields. They have also referred that this feature was extremely important for their work since they need to build queries with a large set of entities, attributes, and conditions;

- Filters Edition: it was pointed out in several posts suggestions to improve the interface in order to support filters comments with color highlighting, since they could be useful to explain the intention of the filter, or even the possibility to disable filters instead of deleting them;

- Result Count: it was referred that there is no visible count of how many rows the query result has;

- Accelerators and Utilities: it has been suggested to add different accelerators in order to accelerate and streamline the query formulation process as well as different ways to present the query structure since, in the user's point of view, the query readability, provided with the existing interface, should be improved.

The description of all problems and suggestions posted in the OutSystems community are detailed in Appendix A - Taxonomy of Problems - Existing Interface in a table that aggregates all problems identified throughout this problem analysis process.

In summary, beyond the necessity of new advanced functionalities, the developers on the OutSystems Community have indicated different situations where the interface turns out not to be as powerful as it could be, since there is a lack of utilities or accelerators that could assist users keeping their task on track.

5.1.5 Existing Interface Evaluation

The last phase of the problem definition was the usability tests to the existing visual query interface, which were performed with two main goals:

1. **Identify problems:** Explore more usability problems that could exist in the interface, by directly observing users interacting with it, under the tasks defined in the prepared testing scenarios;

2. **Gather usability metrics:** Record users' insights about the system as well as the difficulties they had to perform the proposed tasks. This information is going to be processed leading to the attainment of qualitative information to characterize the general opinion of users but also quantitative usability metrics regarding the completeness of the tasks, the time they required to perform it, the operations or elements they did not understand, and their satisfaction using the system.

By this means, **30 users have performed usability tests in a remote environment using Zoom** [82], where the screen was shared and recorded and the visual query builder of Service Studio was tested using the remote control functionality of this meeting platform. Besides, each user allowed to record the session in order to further collect usability metrics and analyze the feedback provided.

Understanding the user's profile:

The profile of each user was identified through a survey where users answered some questions concerning their background namely regarding OutSystems, SQL, and data tools. This information was useful to attribute each user to his respective user group according to his knowledge in relational databases and OutSystems development.[4]

It was tested 10 users of each group and the detailed information about the usability test participants' profile is presented in Table B.1 of Appendix B - Usability Tests Results.

Preparation of the test:

The testing process started with a brief explanation of the data model used across the tests. The communication used to explain the data model was adapted according to the relational database knowledge each user has. When users confirmed they comprehended the data model and the aim of the tasks they started using the interface according to the testing scenarios. More details about the testing scenarios presented and the evaluation method adopted were already described in sections 4.4 and 4.5 respectively.

[4]A complete definition of each user group is provided in section 5.2.

Table 5.3: **Evaluation point S14:** Success rate of users when they needed to add a calculated attribute (Existing Interface usability tests - 30 users).

Insert Calculated Attribute	Added Successfully	Difficulty to Add	Could not Add
OutSystems Developer	91.67%	8.33%	0.00%
Software Developer	20.00%	80.00%	0.00%
Citizen Developer	20.00%	60.00%	20.00%
Total	43.89%	49.44%	6.67%

Observation during the test:

During the test, it was performed a direct observation of users interacting with the system to accomplish the tasks proposed in the scenarios planned. Thereby, a spreadsheet was used to manually register all aspects evaluated in each scenario that was previously enumerated in Table 4.4. The results considered relevant to the usability improvement will be presented below.

In summary, the achievement of the goals of each scenario was registered (effectiveness metrics) as well as the time users spent in each scenario (efficiency metrics). Moreover, as there was scenarios built to test specific problems and functionalities of the interface, specific notes and metrics regarding these functionalities studied were also registered.

Results:

Regarding the learnability of the system, the results confirmed that the lack of visibility of some functionalities have undermined users' ability to find out how to query data in some cases. For example, in the scenario M1, where users needed to create a new calculated attribute (evaluation point S14 according to Table 4.4), there was a set of users who had difficulty to find how to create it due to the hiddenness of the option. Table 5.3, presents the percentage of users who managed to add this attribute without difficulty, the ones who struggle to find it, and the remaining who did not manage to add it although perceived the goal of the proposed task.

Furthermore, the users revealed difficulty to apply group bys or aggregation functions (evaluation point S28 according to Table 4.4). In the formulation scenario F1, the users needed to apply one group by and one average aggregation function in order to reach the task goal. The results revealed that only around 40% of the users managed to group the required data in this scenario, which confirmed the problem predicted by analysis of the interface about the lack of visibility of these functionalities.

On the other hand, the tests revealed some problems concerning the effectiveness reaching the intended result. Although the value proposition of the visual interface is to empower users on understanding and formulating queries, it was identified occasions where users misinterpreter the represented queries or made slips in query formulation cases.

Table 5.4: **Evaluation point S7:** Readability rate of the join which represents the case where two entities were merged using a left join and the conditions specified the that the primary key of the right enitity must be null (Existing Interface usability tests - 30 users).

Left Join with Null Identifier Comprehension	Identified	Did not Identify	Did not see the Join
OutSystems Developer	16.67%	83.33%	0.00%
Software Developer	10.00%	40.00%	50.00%
Citizen Developer	0.00%	50.00%	50.00%
Total	8.89%	57.78%	33.33%

Table 5.5: **Evaluation point S8:** Readability of the foreign keys used to join entities that could be joined using different attributes (Existing Interface usability tests - 30 users).

Foreign Key Identification	Identified	Did not Identify	Did not See the Join
OutSystems Developer	27.78%	72.22%	0.00%
Software Developer	20.00%	36.67%	43.33%
Citizen Developer	0.00%	35.00%	65.00%
Total	15.93%	47.96%	36.11%

For instance, the scenario C1, which includes a join between ”$Sample\_Employee$” and ”$Sample\_Notification$” with the intend of fetching only the employees who have never created any notification, demonstrated that the representation used was misleading users to correctly interpret the query. Observing the results presented in Table 5.4, regardless of their groups, all users did not manage to entirely understand this join (evaluation point S7 of Table 4.4), even the users who are accustomed to use the visual query builder (OutSystems Developers).

In addition, the users have also exhibited other problems regarding the comprehension and formulation of the queries which contain joins between two entities that could be made using different foreign keys (evaluation point S8 of Table 4.4). For example, considering the data model presented in Figure4.1, an account has three foreign keys referencing employees: creators, managers and owners of the account. In those cases it is important to easily perceive which foreign key was used to made each join between the two entities. In this respect, only 15.93% of the users identified this join aspect, which revealed that the interface was not supporting users to further comprehend the query's details.

From a query formulation point of view, only 36.67% of the users selected the intended foreign key when they added entities that could be joined using different attributes (evaluation point S22 of table 4.4). Tables 5.5 and 5.6 detail the results regarding the query aspect described. It was concluded that the interface was not properly guiding users throughout the data querying process and neither facilitating their work in these specific cases.

Table 5.6: **Evaluation point S22:** Selection of the foreign key, in the formulation scenario, used to join entities that could be joined using different attributes (Existing interface usability tests - 30 users).

Foreign Key Selection	Selected	Did not Select	Did not Add Join
OutSystems Developer	50.00%	50.00%	0.00%
Software Developer	40.00%	40.00%	20.00%
Citizen Developer	20.00%	40.00%	40.00%
Total	36.67%	43.33%	20.00%

Table 5.7: **Evaluation point S1:** Average success rate of all scenarios by user group (Existing Interface usability tests - 30 users).

User Group	**Not Achieved**	**Partially Achieved**	**Achieved**
OutSystems Developer	27.14%	20.00%	52.86%
Software Developer	50.00%	20.00%	30.00%
Citizen Developer	70.00%	8.00%	22.00%
Total	**46.84%**	**16.84%**	**36.32%**

Finally, it was summarized metrics to evaluate the users' performance regarding effectiveness, efficiency, and satisfaction for all testing scenarios performed. Table 5.7 represents the average of the success of each user group for all scenarios performed (evaluation point S1 of Table 4.4). The success was classified according to the Table 4.5 already presented and the description of user testing scenarios and user groups which are detailed in section 4.4 and in section 5.2 respectively. The detailed results of each scenario are detailed in Table B.4 of Appendix B - Usability Tests Results.

Even though there were negative results, it was visible how the SQL knowledge and the experience using the OutSystems platform impacted the success rate. The percentage of scenarios which were not achieved by Citizen Developers was 70%. When comparing this value with the value of Software Developers, who have in general more SQL knowledge, the not achieved scenarios represented 50%. In that way, there was a significant difference between these two types of users, meaning that, the existing interface is highly dependent on the users' SQL knowledge. Lastly, if the results of the Software Developers are compared with the OutSystems Developers results, it can be observed that the not achieved percentage decreases from 50% to 27.14%, revealing also that users who are accustomed to using the interface had a superior success rate.

Figure 5.4 illustrates the difference of the effectiveness according to the user group (evaluation point S1 of table 4.4), where it is highly visible the usability problems of the interface which are preventing users to reach the tasks' goals, allowing to conclude that for users who use the query builder for the first time, the learnability curve could be flattened if the user has SQL knowledge.

The information about the background of each user regarding SQL and OutSystems

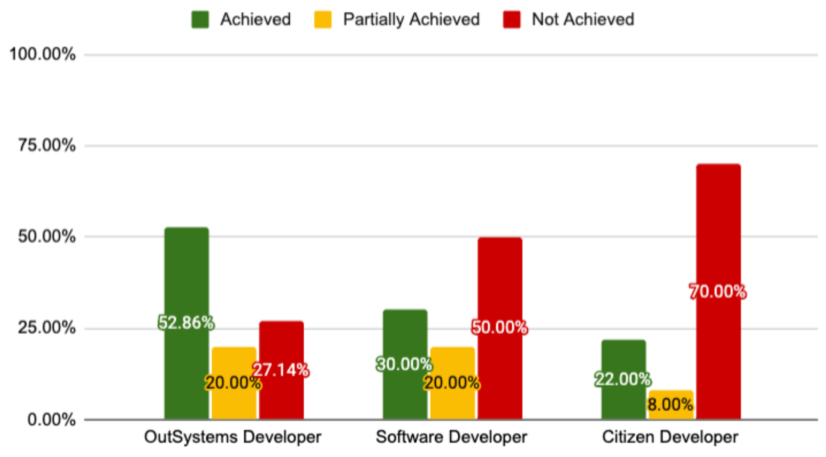


Figure 5.4: **Evaluation point S1:** Average of the sucess rate of all scenarios by each user group (Existing Interface usability tests - 30 users).

were filled in the survey, asked before starting the tests. Analyzing the relation between the SQL knowledge and the OutSystems experience with the scenarios success results, presented in Figure 5.5, led to concluded that in average, a user who never or almost never have used a query language did not achieve 80% and 67%, respectively, of scenarios' goals.

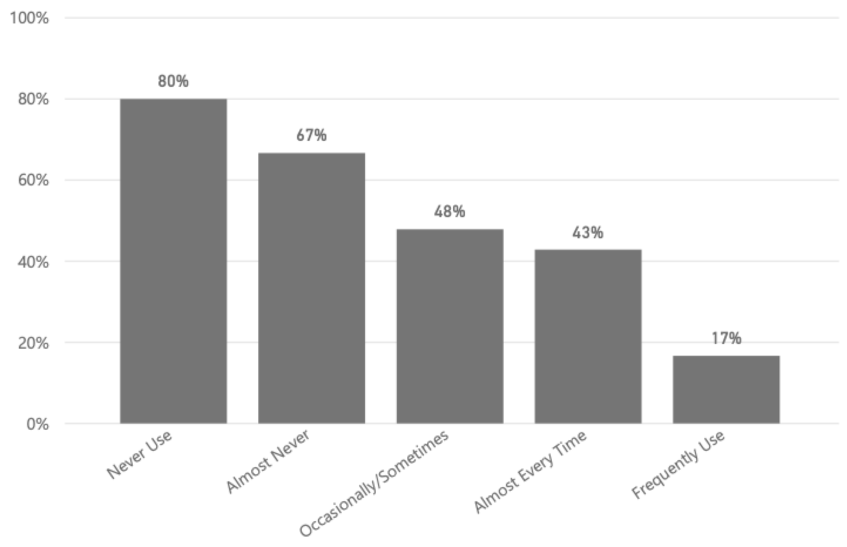
During the tests, users have mentioned other details of the interface that could be a useful input for the next design iterations. In that way, that feedback was also taken into account and registered as well as all the other issues identified.

The result of those analyses described in this section was combined in a table that characterizes each identified problem of the existing interface. This table is presented in the Appendix A - Taxonomy of Problems - Existing Interface and describes all the problems identified. For each problem, it is detailed the interface components involved, the methods used to identify the problem, and the Nielsen Heuristics [57] affected. Moreover, the issues are classified according to the artifact and task attributes of a Framework adapted from Usability-ODC Framework [11], as well as it contains information regarding OutSystems Community[64] posts and likes related to each problem.

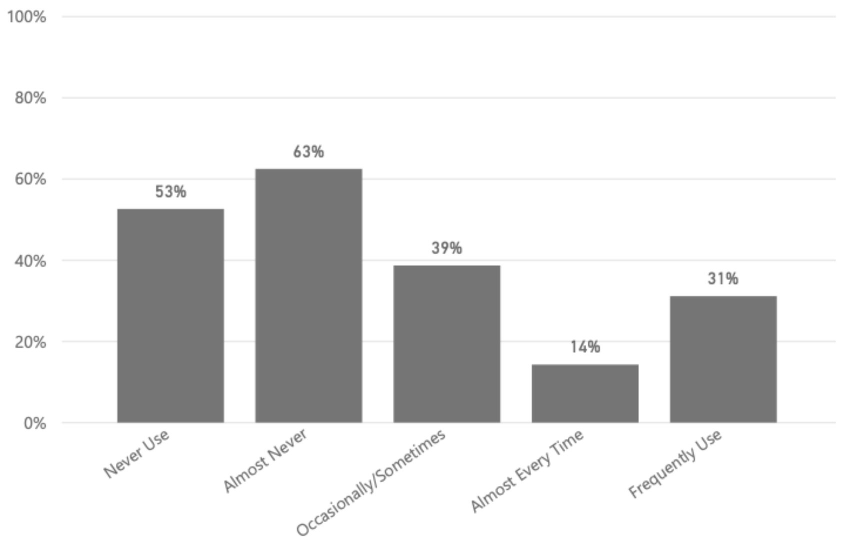
5.2 Target Users

After obtaining a detailed and wide view of the existing problems, the following step was the exploration of how those problems can be solved. As the main goal is the development of an improved interface that gives to the users a solution to manage data queries efficiently, and effectively through intuitive interaction strategies, users are a crucial factor that must be taken into account throughout the entire solution development. Each user has his peculiarities, then the user experience of the solution should be adapted as much as possible to the target users of the query formulation system.

Considering that users will only use the visual query system if they use the OutSystems Platform, the low-code development context where the query system is inserted



(a) "Insuccess rate vs. Have you used a query language (SQL or other)?"



(b) "Insuccess rate vs. Have you already used OutSystems?"

Figure 5.5: Comparison between the users' profile characteristics of the users have not achieved scenarios. The charts illustrates the percentage of scenarios not achieved in average for the users who answered the represented in the horizontal axis to the question presented in the caption. (Existing interface usability tests - 30 users).

cannot be dissociated from the user analysis. Thereby, the user analysis process started by a study to categorize the profile of OutSystems Platform users according to the specificities of the data querying domain.

Since the low-code development paradigm has integrated more people who do not have the strict software engineer profile into software development tasks, the users of the OutSystems Platform do not have the same backgrounds, requirements, and expectations. If, on the one hand, there are OutSystems Developers that have Computer Science academic backgrounds or similar, and experience working with low-level programming languages, on the other hand, there are business experts or specialized in other engineering fields that are also developing in OutSystems. In that way, the provided query building experience should be a hybrid approach that covers the traditional software developers' demands without turning the development and the language less intuitive for people that are not familiarized with classical development patterns, terminologies, or processes.

5.2.1 Requirements and Expectations

As the set of users that may use the visual query system is so broad and heterogeneous, it was necessary to analyze and register the aspects of the user profile which could branch out their needs and expectations in different directions. In this sense, the following three topics of the users' background were considered the ones that could accurately cluster the users' expectations and demands, considering the usage context of the interface, which combines software development, low-code development, and relational databases querying domains:

Software Development Background: The previous knowledge and experiences of users in the software development scope can lead significantly to their expectations while they are interacting with a graphical user interface. Being the object of study an interface that allows users to formulate queries, the factor mentioned becomes even more relevant. In software development, most users often associate database querying to languages that are considered a standard to perform those tasks, such as SQL. In that way, most users end up unconsciously relating the visual query building process to the languages they are familiar with. Therefore, if the visual query interface will be used by users that are familiar with technical languages, it is important to not forget to apply a language that could invoke similar principles and reasonings.

OutSystems Development Experience: The user experience on low-code development, in particular using the OutSystems Platform, has also an impact on how the user will experience the visual data querying tool, even if the user does not build queries regularly through the Platform. As a complete solution for low-code development, the Service Studio has a consistent design across its sections. Consequently, a user familiarized with the Platform could have more facility to find options and understand interface language than a user that is using the Platform for the first time, since there are multiple design patterns and built-in behaviors across all the product.

Nevertheless, users who have considerable experience using the existing solution of OutSystems to build queries could be highly adapted to the existing design. Therefore, this fact should be taken into account throughout the solution design process in order to realize how modifications could impact regular users of the systems. Even knowing that frequent users could be skeptical of changes, it is necessary to properly assess whether the changes only require adaptation or if users could reject them. Therefore, it is important to improve the user experience of the existing interface without removing all its most representative features, in order to keep its singularity.

Data Tools Expertise: Besides textual DQLs, other tools allow users to manage and visualize data through graphical user interfaces. For instance, spreadsheet applications such as Microsoft Excel [51] and Google Sheets [47] presents an intuitive interface where users can manage, organize, and edit data. Using these interfaces, users do not need to know how relational databases work since data is presented in tables that could be manipulated directly under simple controls. For this reason, the language used in those systems is frequently less technical, in such a way that is important to consider that these users are expecting direct manipulation of data and simple language to query and manage data.

5.2.2 User Groups

Being the relational database knowledge a crucial point to frame users' mental model of query formulation, it is important to split up the users who perceive relational databases to the other ones. On the other hand, low-code programming experience can affect how users interact with a visual programming system, thus it is important to study users with experience in low-code development through a different perspective. Considering the aspects mentioned, three user groups were created in order to cluster users that have similar profiles. Thereby, each user could integrate one of the following groups according to their characteristics:

Table 5.8: User Groups

Software Developer	Software Developer or Engineer who has a solid previous knowledge of programming and databases. These users are familiarized with textual programming languages, such as C#, SQL, and others. In that way, they cannot abstract their previous knowledge in such a way that communication with these users should be more technical and specific. Finally, this user group is not an expert in low-code development. However, as they have solid programming, logic, and database knowledge, they could develop some applications using low-code if necessary.
OutSystems Developer	Independently of their background, these users are experienced in OutSystems. That way, they are proficient and faster in low-code development, regardless of their experience in other traditional software development paradigms.
Citizen Developer	Users who do not have an extensive programming or software development background. Even though they are not software developers, they could develop some simple apps using low-code due to its simplicity. As some of these users may not know how relational databases work, they may use other applications, such as Microsoft Excel, Google Sheets, Salesforce, and others to manage data.

Chapter 6

Design and Implementation

The design process adopted was an iterative design approach to iteratively build the solution according to the users' acceptance and feedback. Throughout this chapter, the design and implementation decisions taken in order to reach the final prototype are detailed.

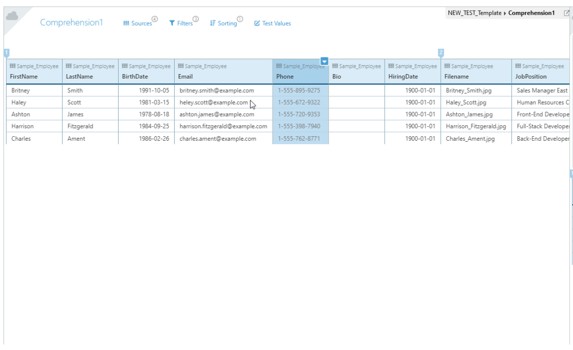
6.1 Sketching

Before starting the development of the interface prototypes, which will be tested with users, some sketches were performed, in order to organize and explore ideas to tackle the existing usability problems. The primary goal expected to attain at the end of this phase is not a functional or complete interface, but a practical idea of what could be designed in the following prototypes.

Considering the extensive list of problems identified, prioritization was an important aspect taken into account throughout all design phases. The first problem explored was the difficulty to comprehend the database query purpose.

In the existing interface, users do not have a unique and clear view, which can facilitate the comprehension of what data could be fetched from the database through the presented query. As illustrated in Figure 6.1, the users can only see some components of the query at a time, due to the existence of tabs to individualize each type of query components (i.e., a tab for source, filter and sort options). In addition, as can be observed in Figure 6.1a, when users open the query, only the query output preview was shown. That was considered a problem in the analysis of the interface for two different points of view:

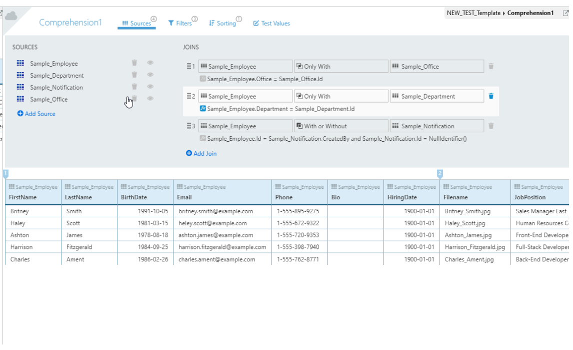
- When the existing interface was tested with users who do not usually use the Platform, they could not easily find the tabs, which increased the chances of wrongly



Comprehension1 · Sources 4 · Filters 3 · Sorting 1 · Test Values · NEW_TEST_Template › **Comprehension1**

Sample_Employee FirstName	Sample_Employee LastName	Sample_Employee BirthDate	Sample_Employee Email	Sample_Employee Phone	Sample_Employee Bio	Sample_Employee HiringDate	Sample_Employee Filename	Sample_Employee JobPosition
Britney	Smith	1991-10-05	britney.smith@example.com	1-555-895-9275		1900-01-01	Britney_Smith.jpg	Sales Manager East
Haley	Scott	1981-03-15	heley.scott@example.com	1-555-672-9322		1900-01-01	Haley_Scott.jpg	Human Resources C
Ashton	James	1978-08-18	ashton.james@example.com	1-555-720-9353		1900-01-01	Ashton_James.jpg	Front-End Develope
Harrison	Fitzgerald	1984-09-25	harrison.fitzgerald@example.com	1-555-398-7940		1900-01-01	Harrison_Fitzgerald.jpg	Full-Stack Developer
Charles	Ament	1986-02-26	charles.ament@example.com	1-555-762-8771		1900-01-01	Charles_Ament.jpg	Back-End Developer

(a) Starting point (only the result is visible)



Comprehension1 · Sources 4 · Filters 3 · Sorting 1 · Test Values · NEW_TEST_Template › **Comprehension1**

SOURCES
- Sample_Employee
- Sample_Department
- Sample_Notification
- Sample_Office

Add Source

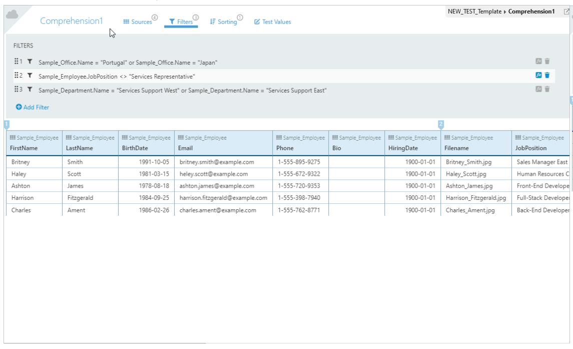
JOINS

1	Sample_Employee	Only With	Sample_Office
	Sample_Employee.Office = Sample_Office.Id		
2	Sample_Employee	Only With	Sample_Department
	Sample_Employee.Department = Sample_Department.Id		
3	Sample_Employee	With or Without	Sample_Notification
	Sample_Employee.Id = Sample_Notification.CreatedBy and Sample_Notification.Id = NullIdentifier()		

Add Join

Sample_Employee FirstName	Sample_Employee LastName	Sample_Employee BirthDate	Sample_Employee Email	Sample_Employee Phone	Sample_Employee Bio	Sample_Employee HiringDate	Sample_Employee Filename	Sample_Employee JobPosition
Britney	Smith	1991-10-05	britney.smith@example.com	1-555-895-9275		1900-01-01	Britney_Smith.jpg	Sales Manager East
Haley	Scott	1981-03-15	heley.scott@example.com	1-555-672-9322		1900-01-01	Haley_Scott.jpg	Human Resources C
Ashton	James	1978-08-18	ashton.james@example.com	1-555-720-9353		1900-01-01	Ashton_James.jpg	Front-End Develope
Harrison	Fitzgerald	1984-09-25	harrison.fitzgerald@example.com	1-555-398-7940		1900-01-01	Harrison_Fitzgerald.jpg	Full-Stack Developer
Charles	Ament	1986-02-26	charles.ament@example.com	1-555-762-8771		1900-01-01	Charles_Ament.jpg	Back-End Developer

(b) Sources Tab



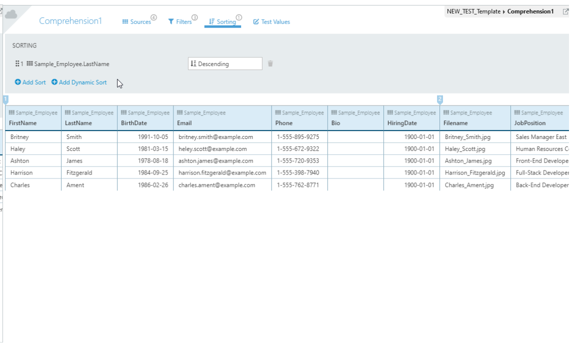
Comprehension1 · Sources 4 · Filters 3 · Sorting 1 · Test Values · NEW_TEST_Template › **Comprehension1**

FILTERS
1. Sample_Office.Name = "Portugal" or Sample_Office.Name = "Japan"
2. Sample_Employee.JobPosition <> "Services Representative"
3. Sample_Department.Name = "Services Support West" or Sample_Department.Name = "Services Support East"

Add Filter

Sample_Employee FirstName	Sample_Employee LastName	Sample_Employee BirthDate	Sample_Employee Email	Sample_Employee Phone	Sample_Employee Bio	Sample_Employee HiringDate	Sample_Employee Filename	Sample_Employee JobPosition
Britney	Smith	1991-10-05	britney.smith@example.com	1-555-895-9275		1900-01-01	Britney_Smith.jpg	Sales Manager East
Haley	Scott	1981-03-15	heley.scott@example.com	1-555-672-9322		1900-01-01	Haley_Scott.jpg	Human Resources C
Ashton	James	1978-08-18	ashton.james@example.com	1-555-720-9353		1900-01-01	Ashton_James.jpg	Front-End Develope
Harrison	Fitzgerald	1984-09-25	harrison.fitzgerald@example.com	1-555-398-7940		1900-01-01	Harrison_Fitzgerald.jpg	Full-Stack Developer
Charles	Ament	1986-02-26	charles.ament@example.com	1-555-762-8771		1900-01-01	Charles_Ament.jpg	Back-End Developer

(c) Filters Tab



Comprehension1 · Sources 4 · Filters 3 · Sorting 1 · Test Values · NEW_TEST_Template › **Comprehension1**

SORTING
1. Sample_Employee.LastName — Descending

Add Sort · Add Dynamic Sort

Sample_Employee FirstName	Sample_Employee LastName	Sample_Employee BirthDate	Sample_Employee Email	Sample_Employee Phone	Sample_Employee Bio	Sample_Employee HiringDate	Sample_Employee Filename	Sample_Employee JobPosition
Britney	Smith	1991-10-05	britney.smith@example.com	1-555-895-9275		1900-01-01	Britney_Smith.jpg	Sales Manager East
Haley	Scott	1981-03-15	heley.scott@example.com	1-555-672-9322		1900-01-01	Haley_Scott.jpg	Human Resources C
Ashton	James	1978-08-18	ashton.james@example.com	1-555-720-9353		1900-01-01	Ashton_James.jpg	Front-End Develope
Harrison	Fitzgerald	1984-09-25	harrison.fitzgerald@example.com	1-555-398-7940		1900-01-01	Harrison_Fitzgerald.jpg	Full-Stack Developer
Charles	Ament	1986-02-26	charles.ament@example.com	1-555-762-8771		1900-01-01	Charles_Ament.jpg	Back-End Developer

(d) Sorting Tab

Figure 6.1: Example of a database query representation through the existing visual interface.

guessing the query purpose, since they have only observed the query output preview. Besides not being an effective understanding technique, it can be even more difficult when the output contains several columns;

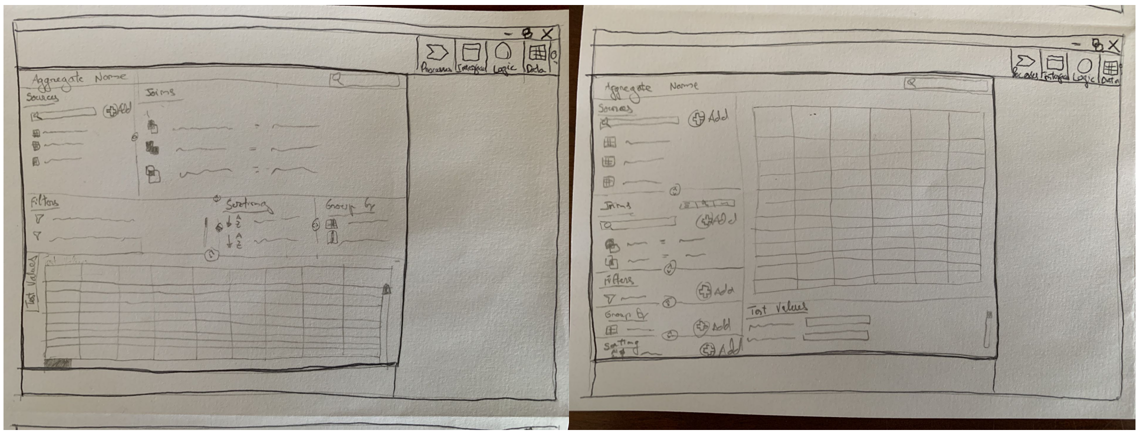
- The users accustomed to this data tool revealed that is cumbersome to open and navigate between tabs in order to understand the query.

Therefore, the designing of a new general layout, where it is possible to view the most important query components at once, was considered a major requirement. A solid improvement regarding this component, could optimize not only the time and effort required to comprehend queries but also to formulate them, since all information is more visible and accessible.

From other perspective, there was also the concern to build a layout that would not compromise the system usability for more complex cases, such as queries that contain a relevant number of entities or conditions.

Accordingly, some wireframes were sketched in order to perceive how the query components could be jointly combined in a unique view. Figure 6.2 illustrates the two options elected from multiple approaches explored.

In both options, there are two principal areas in the interface as presented in the current version of the interface: the query editor area and the preview of the query result.



(a) Option A (b) Option B

Figure 6.2: Interface layout sketches.

Yet, two new manners were explored to display all information of editors jointly with the query result preview. In Option A, the sub-editors remained in the top area and the query result preview below. The Option B illustrates another sketched possibility where all editors were presented on the left side of the screen and the visualization of the results presented on the right side.

Regardless of the option, the layouts presented have in a single view the most important aspects to understand what query is built.

The existing interface does not have any mechanisms that allows the user to easily find an entity or attribute. The attributes included in the query are not represented in any region of the interface beyond the query result table. Thereby, the unique way to find out the attribute is to look for it in the table header of the query output, using the horizontal scroll. That process is cumbersome and slow, so an alternative was sketched in order to include the attribute in the sources view. The users could click on the attributes and the attribute would be automatically highlighted in the query result preview, as illustrated in Figure 6.3. The idea of a search engine was also considered to prompt users to search for an entity or attribute.

Furthermore, other approaches, more compact and functional, were also explored in order to display the joins used in the query. Figure 6.4 shows the sketched ideas to represent joins: a simple list of all join operations inside the query, and two other ones aggregated by the entities involved by the join kind.

Lastly, an idea to accelerate the searching process of a specific element inside the query was sketched. The sketch represented in Figure 6.5 exhibits an idea, taken into account, to find all references of an entity. This general search would allow users to find entities, joins, filters, or other query elements faster.

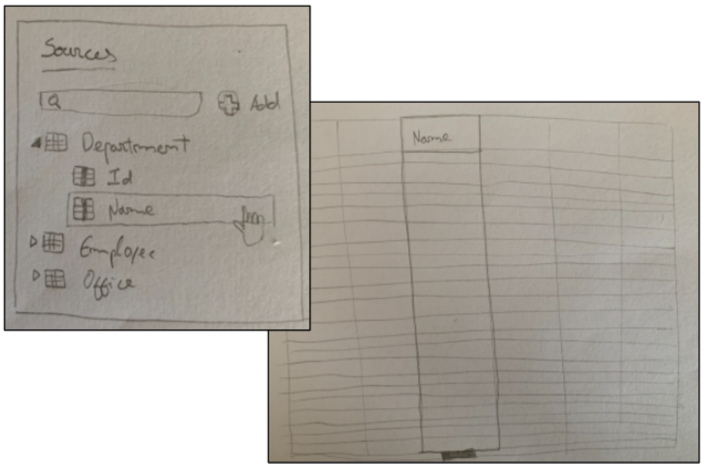


Figure 6.3: Searching for an attribute data on the query output preview.

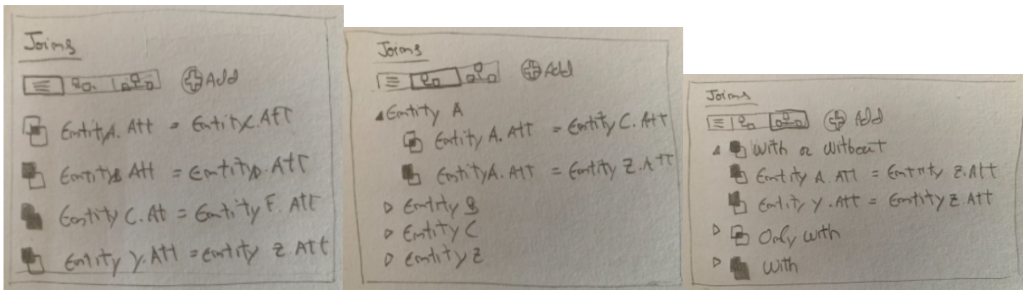

Joins
Add
EntityA. Att = EntityC.Att
EntityB.Att = EntityD.Att
Entity C.Att = Entity F. Att
Entity Y.Att = Entity Z Att
Joins
Add
Entity A
Entity A. Att = Entity C. Att
EntityA. Att = Entity Z.Att
Entity B
Entity C
Entity Z
Joins
Add
With or Without
Entity A. Att = Entity Z.Att
Entity Y. Att = Entity Z Att
Only with
With
(a) Joins flat list (b) Joins associated to each entity (c) Joins by each join kind

Figure 6.4: Sketches elaborated to explore other approaches to represent the join operations present in the query.

6.2 Paper Prototype

The iterative design process started entirely with the first prototype developed: the paper prototype. The main goal of this phase was to build a functional prototype using paper, ruler, try square, and writing materials, in order to create faster and with a low-risk level (i.e., reduced implementation effort spent) a prototype that could be tested by users.

However, due to the COVID-19 pandemic, the prototype was virtually adapted, since it was not possible to test the prototype in person. Accordingly, the prototype was scanned and the interactions were configured using the digital product design platform InVision [48].

6.2.1 Design

The building process of this low-fidelity prototype started with the design phase, where a brainstorming of ideas, explored previously, took place, in order to establish the design priorities for the paper prototype as well as concrete ideas to apply in the solutions.

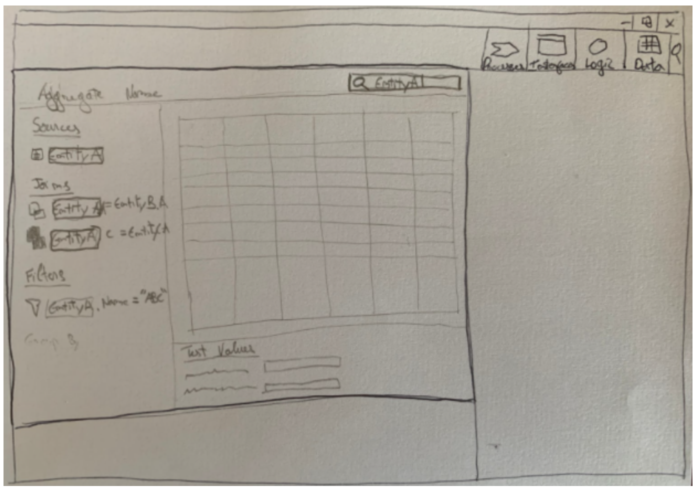


Figure 6.5: Sketch of a general search to allow users to find query elements represented in the interface.

Sub-editors arrangement:

Taking into account the sketches mentioned in the last section, the first question evaluated was which rearrangement of the sub-editors and the result preview views would be implemented in the next phase. In order to take action in this regard, there was a point in the visual query builder background considered. As mentioned in section 2.2.2.1, this querying interface was completely redesigned seven years ago, which had a negative repercussion on users since they were accustomed to the previous interface. Therefore, there was a concern to change the interface without removing the main points that characterize and identifies it. In that way, the goal was to ensure that users consider the new interface as an improved version of the existing one, instead of a completely different interface.

Comparing the two options sketched (Figure 6.2) with the existing interface (Figure 6.1), the Option A is the most similar because the editors area keeps on the top region of the interface and the query result data below. For this reason, this arrangement was considered to be presented in the next phase.

Having chosen where would take place the edition area of the interface, the second aspect taken into consideration was how to organize its sub-editors. This decision was taken, bearing in mind, what each query aspect represents in the user's mental model when they formulate queries as well as the physical space of the interface they could occupy.

In this regard, the way users think about the query elements was taken into account. This reasoning was performed taking into consideration the formulation using SQL since one of the main goals is the improvement of the interface experience for users that are proficient in SQL. According to the conceptual models presented in section 3.1, the first aspect the user thinks, after understanding what data is required, was assessed to be how to translate them to the query language. In SQL, the core statements are presented in the following order:

1. **SELECT:** Indicates which attributes will be selected to be present in the query output table;
2. **FROM:** Sets out which entities are used to query data. Thereby, even it is necessary to merge tables, the join operations are specified through that statement;
3. **WHERE:** Contains the boolean conditions that would filter the results.
4. **ORDER BY:** Specify the criteria to order the data gathered.

In this visual query building tool users do not select attributes because there is an optimizing background task that will inspect where the query is used and only select the attributes that will be used. Therefore, in this system, the information specified through the SELECT statement will not be specified by the user.

However, the three other aspects of the query are clearly specified by users in three different interface areas: Sources, Filters, and Sorting respectively. Accordingly, the arrangement chosen to display these three sub-editors, the following logic was considered: the query edition area were divided into two columns, the left side was elected to represent the sources of the query and the right side lists the filters and sorting criteria.

New approach to represent sources and joins:

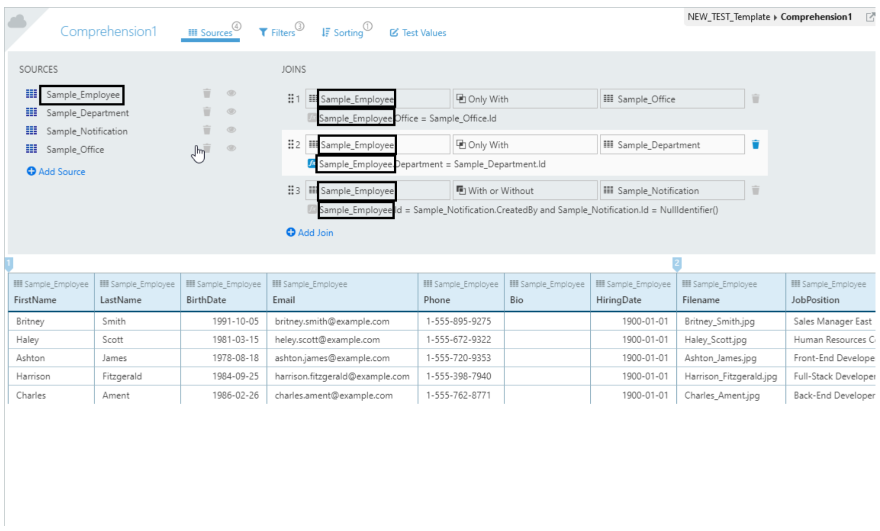
Nevertheless, the way entities and joins were presented in the interface, in the previous sources tab, required to be redesigned from scratch. As can be observed in Figure 6.1b, a simple list of the entities used was presented in the left side of the editor area and the joins used to merge them in its right side.

Even though some designs regarding that were elaborated in the sketching phase (Figure 6.4), it was concluded that the main issues remain:

- Difficulty to comprehend, fast and with a low-effort, what entities were integrated into the query. Being a visual interface, the comprehension of the entities used as source to formulate the query should be easier to understand. However, the potential of the visual interface has not been leveraged, mainly due to the following aspects:
  - **Textual language overloading:** Not only all join conditions were completely presented in full-textual way, but also the same entity could be written in a repeated way. For example, in the example shown in Figure 6.6, the entity ”*Sample_Employee*” was presented seven times to indicate that the query uses this entity and this entity was joined with three other entities;
  - **Lack of guidance to understand what entities are related to each other:** As the entities and sources are listed in the interface, the design should help users to understand which entities are joined. For instance, if joins were put

between the entities involved, it would be simpler to identify if they were merged through a join operation.

- The users who are not familiarized with relational databases and the terminology used to join tables pointed out that the existing view is complex and difficult to understand.



SOURCES

- Sample_Employee
- Sample_Department
- Sample_Notification
- Sample_Office

JOINS

1	Sample_Employee	Only With	Sample_Office
	Sample_Employee.Office = Sample_Office.Id		
2	Sample_Employee	Only With	Sample_Department
	Sample_Employee.Department = Sample_Department.Id		
3	Sample_Employee	With or Without	Sample_Notification
	Sample_Employee.Id = Sample_Notification.CreatedBy and Sample_Notification.Id = NullIdentifier()		

Sample_Employee FirstName	Sample_Employee LastName	Sample_Employee BirthDate	Sample_Employee Email	Sample_Employee Phone	Sample_Employee Bio	Sample_Employee HiringDate	Sample_Employee Filename	Sample_Employee JobPosition
Britney	Smith	1991-10-05	britney.smith@example.com	1-555-895-9275		1900-01-01	Britney_Smith.jpg	Sales Manager East
Haley	Scott	1981-03-15	heley.scott@example.com	1-555-672-9322		1900-01-01	Haley_Scott.jpg	Human Resources C
Ashton	James	1978-08-18	ashton.james@example.com	1-555-720-9353		1900-01-01	Ashton_James.jpg	Front-End Develope
Harrison	Fitzgerald	1984-09-25	harrison.fitzgerald@example.com	1-555-398-7940		1900-01-01	Harrison_Fitzgerald.jpg	Full-Stack Developer
Charles	Ament	1986-02-26	charles.ament@example.com	1-555-762-8771		1900-01-01	Charles_Ament.jpg	Back-End Developer

Figure 6.6: Example of the existing textual language overloading - The entity "*Sample_Employee*" is represented seven times.

Therefore, the way entities and joins were listed in the sources tab of the existing interface was reconsidered, since it was intended to create a simpler, intuitive, and compact viewing area.

The idea of a tree view which displays all entities and joins used in the query has emerged to tackle the problem mentioned. Through that approach, it would be possible to reduce the entity repetition and to explicitly perceive which entities have a certain entity been joined.

Query formulation improvements:

Regarding query formulation, there was several users who are not accustomed to the query builder, meaning that they appeared to have difficulty to discover some functionalities of the system when they were testing the existing interface. Thereby, learnability was also considering during the design phase of the first prototype.

As mentioned in section 5.1.1, the options to add a new calculated attribute or to apply a group by or aggregation functions were hidden. These functionalities could be frequently used, thus it was concluded that these options should be visible in the sources area. By doing this, not only the options were still visible to the novice users but users could have a faster alternative to insert these query components.

Lastly, the distribution of the formulation options (i.e., the buttons or other interface elements that allow users to insert new elements in the query) in the screen was a relevant aspect considered.

The main concern was to put these controls near to the area where the content will be displayed. As an example, if there is an area where entities, joins, and attributes are placed together, the related interaction should be accessible near them. In that way, users do not need to find options in other areas of the interface, keeping the user task on track, avoiding them to lose reasoning context.

That design principle was considered advantageous to reduce the user's working memory overload since they could focus on each part of the query without distractions. Moreover, if the controls are near, the time the mouse need to move to them, is also reduced, accelerating the query formulation process.

6.2.2 Implementation

After defining priorities for the current design iteration, the paper prototype implementation has launched. In this phase, the pieces of the prototype were built progressively considering the key points presented before and refining some details whenever necessary.

General layout:

As the reasoning applied in the design phase, the general layout was the first part implemented. The main concern at this phase was the definition of the dimensions for each sub-editor. In order to ensure that the design will be properly integrated into the Service Studio, it was created a sketched background image of the IDE using a function of Balsamiq [31] to convert the Service Studio screenshot to a black and white line drawing version [81]. As a result, it was possible to perceive if the design done would combine with the system where it is integrated without increasing the fidelity level of the prototype.

After that, the main areas of the interface were elaborated until reaching the general layout that can be observed in Figure 6.7.

Query Result Table:

The next step was the design of the query result tables which shows the result preview of the query built. An output result table was designed in order to provide, in all user testing scenarios, a real context to users when they tested the prototype.

In order to accelerate the process of the table design in paper, it was used the same Balsamiq functionality used for background, to transform the table of the existing interface into a low-fidelity design.

However, the headers of the table were redesigned manually to improve the readability of the table to optimize the understanding of where data of an entity starts from data of another entity. In that way, the entity reference in the table header were merged to give

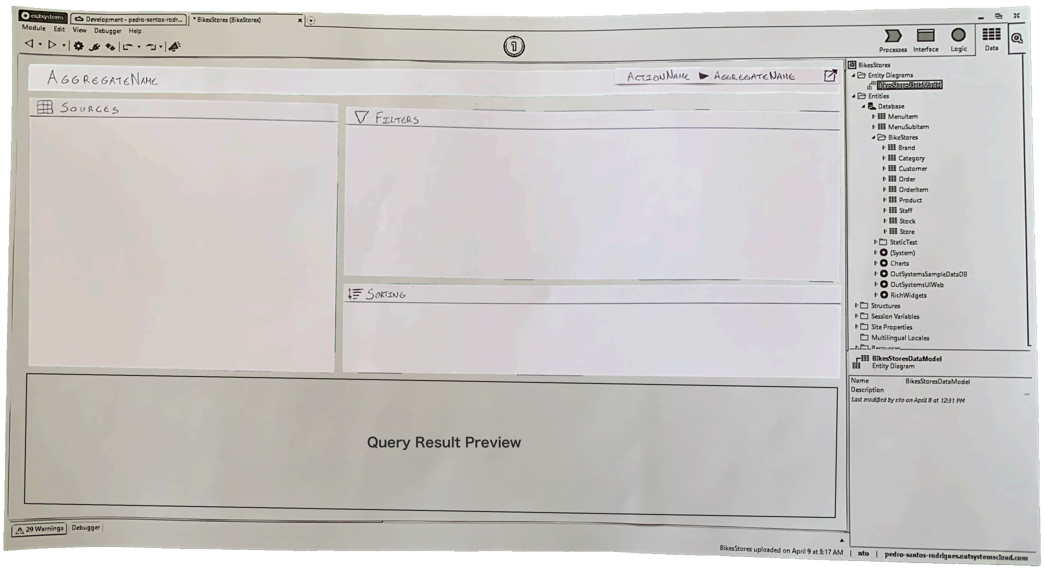
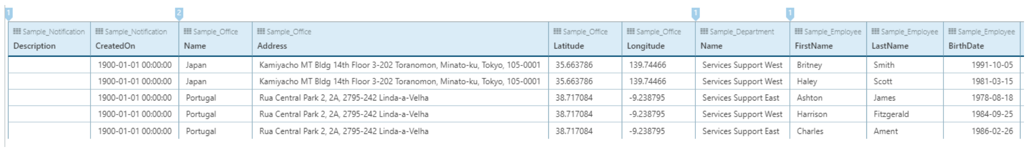
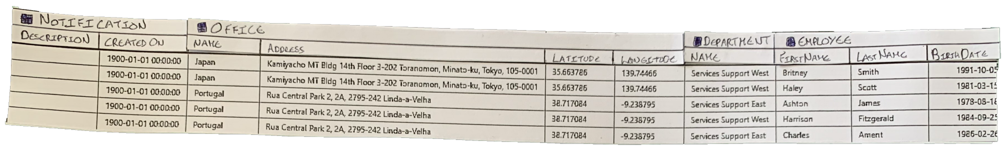


Figure 6.7: Paper Prototype general layout.



Sample_Notification Description	Sample_Notification CreatedOn	Sample_Office Name	Sample_Office Address	Sample_Office Latitude	Sample_Office Longitude	Sample_Department Name	Sample_Employee FirstName	Sample_Employee LastName	Sample_Employee BirthDate
	1900-01-01 00:00:00	Japan	Kamiyacho MT Bldg 14th Floor 3-202 Toranomon, Minato-ku, Tokyo, 105-0001	35.663786	139.74466	Services Support West	Britney	Smith	1991-10-05
	1900-01-01 00:00:00	Japan	Kamiyacho MT Bldg 14th Floor 3-202 Toranomon, Minato-ku, Tokyo, 105-0001	35.663786	139.74466	Services Support West	Haley	Scott	1981-03-15
	1900-01-01 00:00:00	Portugal	Rua Central Park 2, 2A, 2795-242 Linda-a-Velha	38.717084	-9.238795	Services Support East	Ashton	James	1978-08-18
	1900-01-01 00:00:00	Portugal	Rua Central Park 2, 2A, 2795-242 Linda-a-Velha	38.717084	-9.238795	Services Support West	Harrison	Fitzgerald	1984-09-25
	1900-01-01 00:00:00	Portugal	Rua Central Park 2, 2A, 2795-242 Linda-a-Velha	38.717084	-9.238795	Services Support East	Charles	Ament	1986-02-26

(a) Existing Interface



NOTIFICATION		OFFICE				DEPARTMENT	EMPLOYEE		
DESCRIPTION	CREATED ON	NAME	ADDRESS	LATITUDE	LONGITUDE	NAME	FIRSTNAME	LASTNAME	BIRTHDATE
	1900-01-01 00:00:00	Japan	Kamiyacho MT Bldg 14th Floor 3-202 Toranomon, Minato-ku, Tokyo, 105-0001	35.663786	139.74466	Services Support West	Britney	Smith	1991-10-0
	1900-01-01 00:00:00	Japan	Kamiyacho MT Bldg 14th Floor 3-202 Toranomon, Minato-ku, Tokyo, 105-0001	35.663786	139.74466	Services Support West	Haley	Scott	1981-03-1
	1900-01-01 00:00:00	Portugal	Rua Central Park 2, 2A, 2795-242 Linda-a-Velha	38.717084	-9.238795	Services Support East	Ashton	James	1978-08-1
	1900-01-01 00:00:00	Portugal	Rua Central Park 2, 2A, 2795-242 Linda-a-Velha	38.717084	-9.238795	Services Support West	Harrison	Fitzgerald	1984-09-2
	1900-01-01 00:00:00	Portugal	Rua Central Park 2, 2A, 2795-242 Linda-a-Velha	38.717084	-9.238795	Services Support East	Charles	Ament	1986-02-2

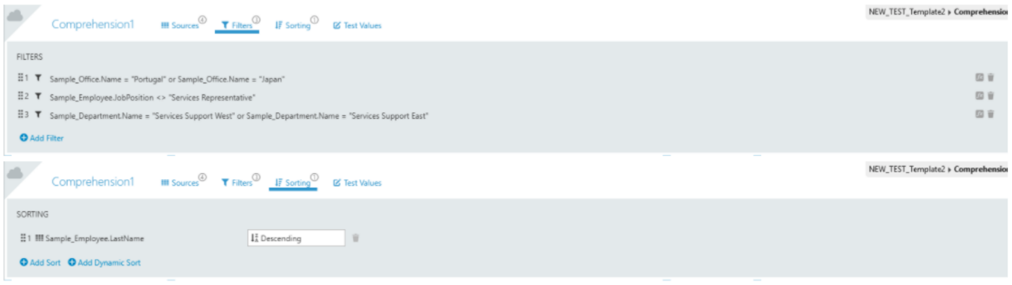
(b) Paper Prototype

Figure 6.8: Comparison between the existing query result table and the designed for the Paper Prototype.

to the user an easier perception that all the attributes below belong to the entity above. Figure 6.8 compares the existing one to an example of one of the tables built.

Filters and Sorting:

Regarding filters and sorting, it was decided to not apply major changes since there were other problems considered as more relevant. Even though the filters edition could be faster using accelerators such as copy and paste or the possibility to edit each filter in the line, instead of opening the expression editor, these problems, as well as other similar problems, mentioned in the Appendix A - Taxonomy of Problems - Existing Interface, were not tackled. The reason to disconsider these improvements in the prototyping phase was that there are straightforward improvements that have a low-risk of affecting negatively the user's experience. In that way, it the focus was to the design parts that should be rigorously tested by users to ensure that riskier changes are improving the


Comprehension1 Sources⁴ Filters¹ Sorting¹ Test Values NEW_TEST_Template2 ▸ Comprehensio

FILTERS

1 Sample_Office.Name = "Portugal" or Sample_Office.Name = "Japan"

2 Sample_Employee.JobPosition <> "Services Representative"

3 Sample_Department.Name = "Services Support West" or Sample_Department.Name = "Services Support East"

Add Filter

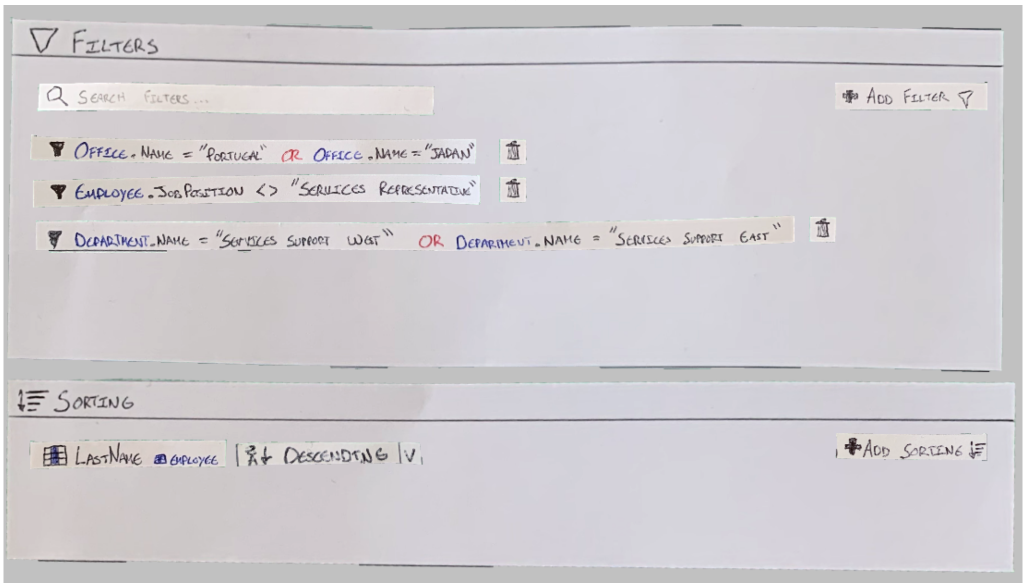
Comprehension1 Sources⁴ Filters¹ Sorting¹ Test Values NEW_TEST_Template2 ▸ Comprehensio

SORTING

1 Sample_Employee.LastName Descending

Add Sort Add Dynamic Sort

(a) Existing Interface


FILTERS
SEARCH FILTERS ...
ADD FILTER
OFFICE.NAME = "PORTUGAL" OR OFFICE.NAME = "JAPAN"
EMPLOYEE.JOBPOSITION <> "SERVICES REPRESENTATIVE"
DEPARTMENT_NAME = "SERVICES SUPPORT WEST" OR DEPARTMENT.NAME = "SERVICES SUPPORT EAST"
SORTING
LASTNAME EMPLOYEE
DESCENDING
ADD SORTING

(b) Paper Prototype

Figure 6.9: Comparison between the existing filters and sorting sub-editors and the ones designed for the Paper Prototype.

usability of the interface.

Accordingly, concerning filters, it was only added some syntax highlighting to each one of the conditions presented in order to reduce the time required to understand the conditions. Figure 6.9 presents an example of Filters and Sorting design in the paper prototype.

Sources View:

As referred before, the design of the sources view was considered one of the most impactful aspects to improve the usability of the query builder. The restructuration design of this element was reasoned as a crucial section of the interface that could leverage the usability of the system.

A new approach to represent sources and joins was progressively built. The first aspect approached was the creation of a hierarchical view similar to a tree view which gives to the user the perception of what entities are related with each other. Figure 6.10 shown an example of a tree built at that stage.

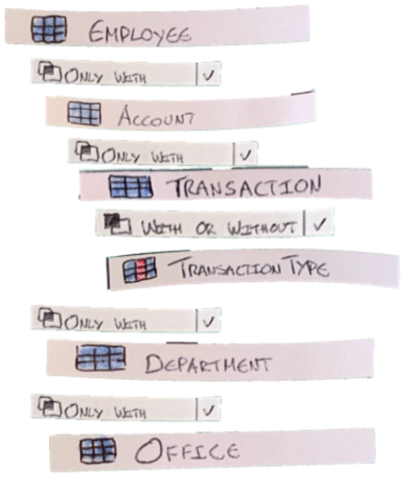


Figure 6.10: First stage of the design of a new representation approach to display sources and joins.

Notwithstanding the visual simplicity achieved was insufficient since it does not represent the join conditions. However, if join conditions were just included near to the join type, the interface would be overloaded again and it would still poor readability.

Furthermore, the usability tests of the existing interface showed that users tended to misunderstand the queries that display joins with specific characteristics. In particular, in cases where queries had joins with conditions that contained logical operators or when the join between the two tables could be done through different attributes.

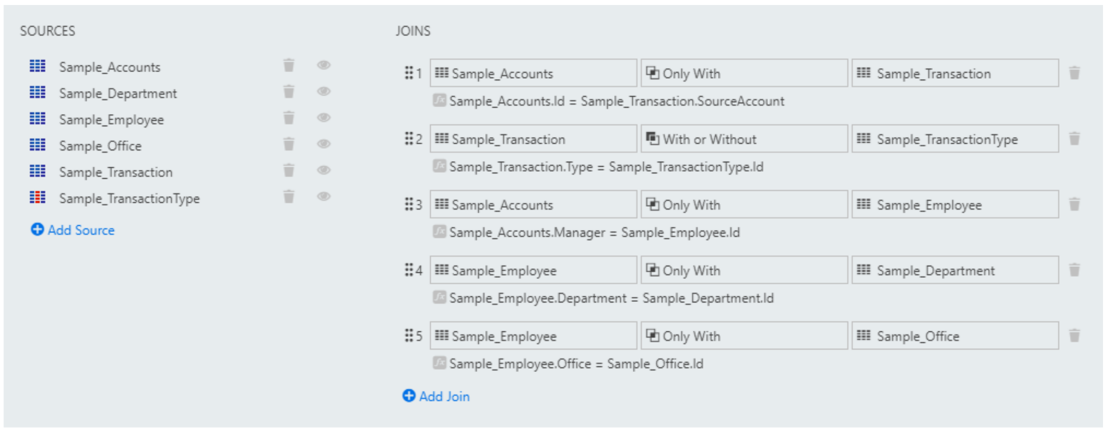
Through practical examples it is easier to understand the dimension of the problem. For example, considering the data model presented in Figure 4.1, if a user wants to query the employees and their departments, he will add the two entities and the visual query builder will build automatically the join condition:

```
"Sample_Employee.Office = Sample_Office.Id"
```

This automatism is useful and turns the query formulation process more simple and efficient. However, there are cases where the join needs to be specified. For instance, considering the same data model, if a user wants to query the employees who are owners of accounts, the join condition has to merge the two entities *"Sample_Employee"* and *"Sample_Account"* using the foreign key *"Owner"* and not the two other available alternatives: *"CreatedBy"* and *"Manager"*. In this case the condition required to build this query is:

```
"Sample_Employee.Id = Sample_Account.Owner"
```

In these cases, the join condition, which in many cases is not so important as just covers the general case and it was generated automatically by the system, represents an important aspect of the query. However, as illustrated in the example of Figure 6.11, the existing interface represents all joins equally.


SOURCES

Sample_Accounts
Sample_Department
Sample_Employee
Sample_Office
Sample_Transaction
Sample_TransactionType
Add Source

JOINS

1 Sample_Accounts | Only With | Sample_Transaction
Sample_Accounts.Id = Sample_Transaction.SourceAccount

2 Sample_Transaction | With or Without | Sample_TransactionType
Sample_Transaction.Type = Sample_TransactionType.Id

3 Sample_Accounts | Only With | Sample_Employee
Sample_Accounts.Manager = Sample_Employee.Id

4 Sample_Employee | Only With | Sample_Department
Sample_Employee.Department = Sample_Department.Id

5 Sample_Employee | Only With | Sample_Office
Sample_Employee.Office = Sample_Office.Id

Add Join

Figure 6.11: Example of joins representation in the existing interface in a query where the foreign key used to join is an important detail.

When users tested the existing interface with cases similar to the one presented, two relevant usability problems were detected. On the one hand, when users tried to comprehend the query, most of them did not pay attention to this detail, not mentioning the foreign keys used. On the other hand, when they need to formulate queries that contemplates these cases, several users, even the most experienced using this query builder, did not select the intended foreign key. This behavior was considered normal since the system just assumes one of the foreign keys and generate the join, without asking user what attribute he would want to use to join the two entities.

Therefore, the exploration of new sources and joins representation was an excellent opportunity to tackle this problem. In that way, the existing tree view of sources was designed in order to integrate also the foreign key used to build each of the presented joins. Moreover, when two entities are added and the join between them can be done through different keys, the system starts asking the user which key he wants to use. Figure 6.12 illustrates how this idea was transposed to the paper prototype.

Nevertheless, the foreign key is not the only aspect that could be specified in the join condition. As in SQL, the existing query builder allows user to edit manually the join condition, for example to add more restrictions using logical operators.

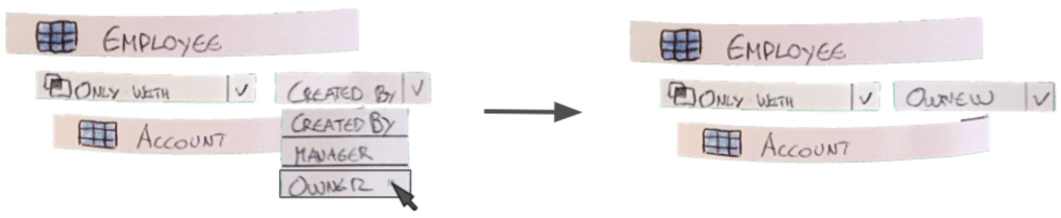


Figure 6.12: Example of the new foreign key selection and representation applied in the Paper Prototype.

In these cases, not only is it important to continue to allow the edition of the conditions in the new interface but the conditions already edited should also be highlighted in such a way that users who open the query for the first time perceive that the condition of these joins is different.

The strategy applied in the paper prototype to maintain the interface simple, clear, and intuitive while highlighting the relevant aspects was to put only the function icon in the simple join conditions, and the other ones has their difference explicitly represented after. Figure 6.13 represents the sources of a query with a join condition edited. In this case the join condition between *"Sample_Employee"* and *"Sample_Notification"* was changed since the goal is to fetch the employees who have never created notifications. Accordingly, a simplification took place to represent only the differentiating factor instead of the full condition which is:

```
"Sample_Employee.Id = Sample_Notification.CreatedBy and
        Sample_Notification.Id = NullIdentifier()"
```

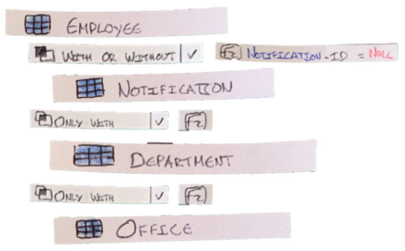


Figure 6.13: Example of the sources of a query that has a join with a condition edited.

Through that approach, users will be able to identify entities and joins of the query without information overloading, allowing them to perceive easily and faster the purpose of the query. Even if the join condition is hidden, the user could click on the function icon and the expression editor would appear making possible to see and edit the join condition selected.

New alternatives to add and search for attributes:

The last improved aspect during the development of this paper prototype was regarding the addition of new attributes. In accordance with the mentioned in the last section, there was a demand to provide other alternatives to add calculated attributes as well as group bys or aggregation functions.

Considering the concern to insert strategically these alternatives, in easily and fast accessible places, it was necessary to improve the visibility of these options to improve the learnability of novice users. The following options were applied:

- **Click and right-click in the attributes exhibited in the Sources sub-editor:** The list of the attributes of each entity was added into each entity represented in the Sources of the query. On the one hand, if users click in an attribute, they would see the data related into the query result preview (according to the sketch already presented in Figure 6.3). On the other hand, if they right-click in an attribute, they would see the same context menu that they already could see when they right-click on the result table headers. Therefore they could apply operations using that alternative, as exemplified in Figure 6.14.

- **Two new visible buttons to add attributes:** In the footer of the Sources View, two buttons were added:

  - **Aggregation / Group By:** Clicking in this button the user could choose an attribute and apply the following operations to group the data of the attribute: Group by, Sum, Average, Max, Min, or Count. Figure 6.15 shows this new formulation area.
  - **Calculated Attribute:** When the user selects this option the interface will show an expression edition area similar to the existing one (as exemplified in Figure 2.9) to indicate the formula of the new attribute. Figure 6.16 shows this alternative to add a new calculated attribute.

In order to improve the query readability and the consistency of the interface, these attributes, added to the query, were put into the sources view as exemplified in Figure 6.17.

Digital Paper Prototype:

Combining all the components elaborated in paper, the result is illustrated in Figure 6.18. In order to make it remotely testable, each one of the interface components were scanned and composed again using the InVision App [48].

Therefore, for each scenario, it was required to build multiple images since each one represents a state among the interface manipulation. In that way, interactions were added to the components to configure the images sequence, as can be observed in Figure 6.19.

In total, 291 images were created and, multiple interactions between them were configured, in order to reach a prototype that could be tested in all scenarios.

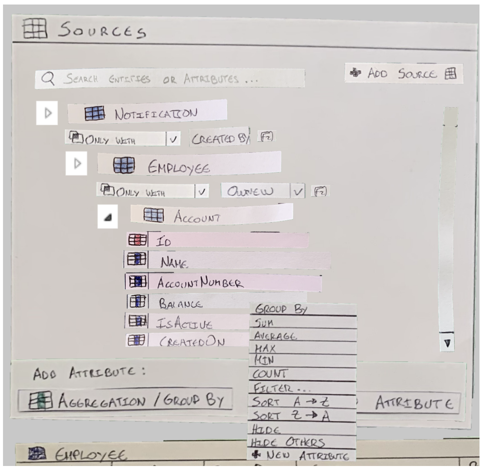


Figure 6.14: Alternative provided to open the attribute context menu.

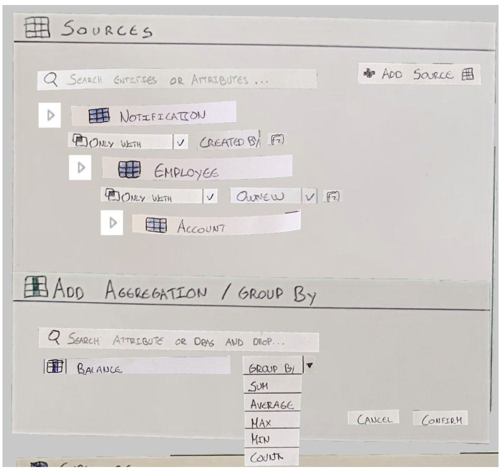


Figure 6.15: New sub-editor into the sources view to add aggregation functions and group bys.

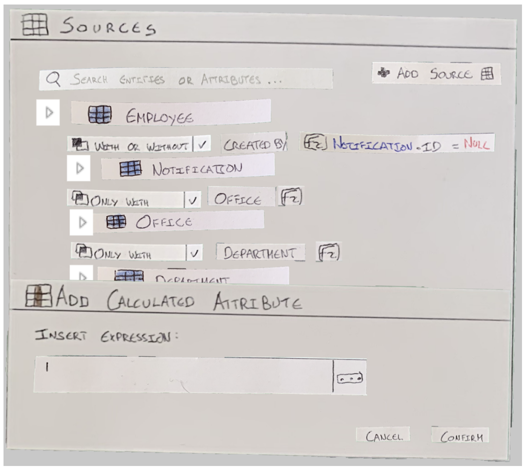


Figure 6.16: New sub-editor into the sources view to add calculated attributes.

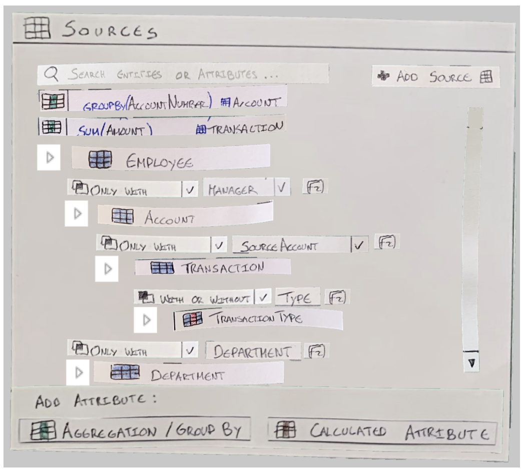
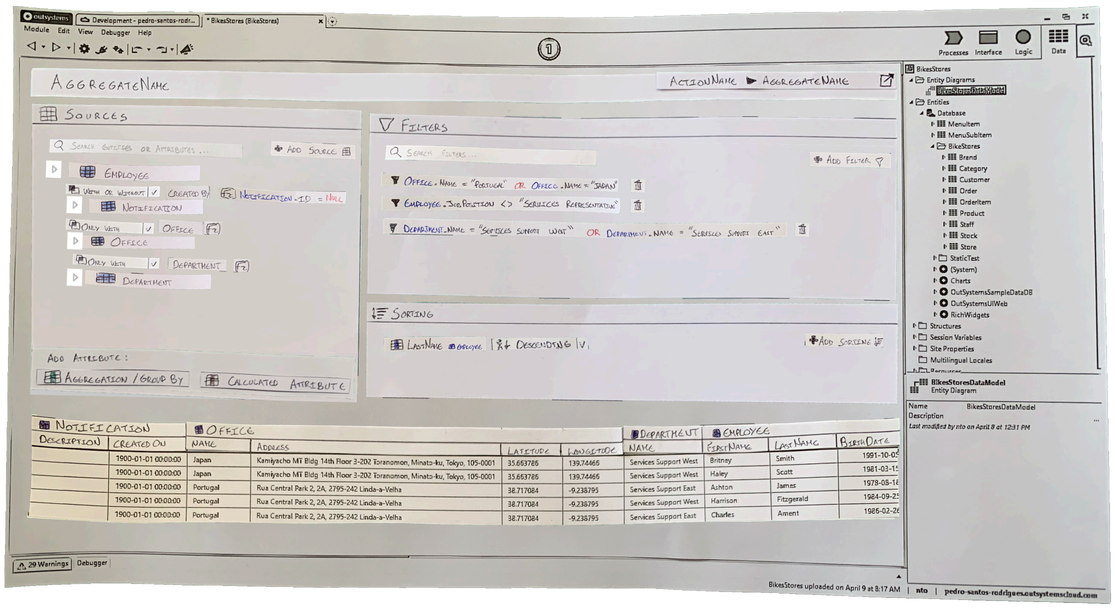


Figure 6.17: Sources of a query where a group by and a SUM aggregation function were applied.



Notification		Office				Department	Employee		
Description	Created On	Name	Address	Latitude	Longitude	Name	FirstName	LastName	BirthDate
	1900-01-01 00:00:00	Japan	Kamiyacho MT Bldg 14th Floor 3-202 Toranomon, Minato-ku, Tokyo, 105-0001	35.663786	139.74466	Services Support West	Britney	Smith	1991-10-05
	1900-01-01 00:00:00	Japan	Kamiyacho MT Bldg 14th Floor 3-202 Toranomon, Minato-ku, Tokyo, 105-0001	35.663786	139.74466	Services Support West	Haley	Scott	1981-03-15
	1900-01-01 00:00:00	Portugal	Rua Central Park 2, 2A, 2795-242 Linda-a-Velha	38.717084	-9.238795	Services Support East	Ashton	James	1978-08-18
	1900-01-01 00:00:00	Portugal	Rua Central Park 2, 2A, 2795-242 Linda-a-Velha	38.717084	-9.238795	Services Support West	Harrison	Fitzgerald	1984-09-25
	1900-01-01 00:00:00	Portugal	Rua Central Park 2, 2A, 2795-242 Linda-a-Velha	38.717084	-9.238795	Services Support East	Charles	Ament	1986-02-26

Figure 6.18: Final Paper Prototype before scanning it.

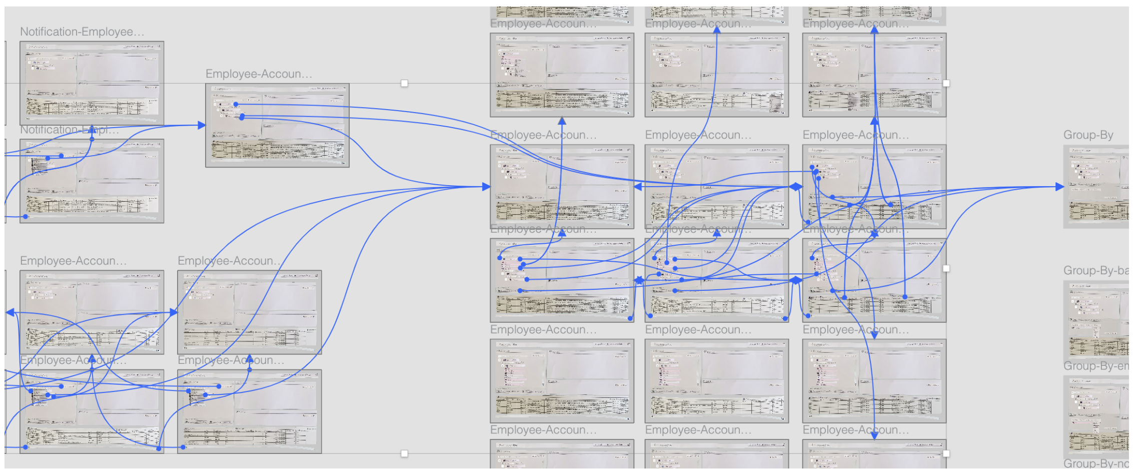


Figure 6.19: Brief example of a small part of the interaction configuration in InVision in order to allow to test the Paper Prototype in a fluid way.

6.2.3 Evaluation

After completing the implementation of the Paper Prototype, the user testing phase has started. This prototype was tested with 5 users of each user group to perform a qualitative analysis regarding the changes applied in the interface. The testing methodology used was the same used to test the existing interface evaluation, completely described in section 5.1.5. Accordingly, the Zoom [82] meeting solution was used to share and record the screen, showing the Paper Prototype built in InVision [48], thus users managed to interact with the prototype using the Zoom remote control feature. In that way, the aim, at this stage, was to map the improved aspects and details which require a redesign in the following phase.

Understanding the user's profile:

The profile of each user was identified through a survey where users answered some questions concerning their background namely regarding OutSystems, SQL, and data tools. This information was useful to attribute each user to his respective user group according to his knowledge in relational databases and OutSystems development.[1]

The detailed information about the usability test participants' profile (15 participants) are presented in Table B.2 of Appendix B - Usability Tests Results.

Preparation of the test:

The testing process started with a brief explanation of the data model used across the tests. The communication used to explain the data model was adapted according to the relational database knowledge each user has. When users confirmed they comprehended the data model and the aim of the tasks they started using the interface according to the testing scenarios. More details about the evaluation method adopted and the testing scenarios planned were already described in sections 4.5 and 4.4 respectively.

[1]A complete definition of each user group is provided in section 5.2.

Table 6.1: Average of the success rate of all scenarios by user group (Paper Prototype usability tests - 15 users).

User Group	Not Achieved	Partially Achieved	Achieved
OutSystems Developer	11.43%	2.86%	85.71%
Software Developer	14.29%	8.57%	77.14%
Citizen Developer	28.00%	4.00%	68.00%
Total	**17.90%**	**5.14%**	**76.95%**

Observation during the test:

During the test, it was performed a direct observation of users interacting with the system to accomplish the tasks proposed in the scenarios planned. Thereby, a spreadsheet was used to manually register all aspects evaluated in each scenario that was previously enumerated in Table 4.4.The results considered relevant to the usability improvement will be presented below.

In summary, the achievement of the goals of each scenario was registered (effectiveness metrics) as well as the time users spent in each scenario (efficiency metrics). Moreover, as there was scenarios built to test specific problems and functionalities of the interface, specific notes and metrics regarding these functionalities studied were also registered.

Results:

Regarding the achievement of the tasks proposed, Table 6.1 summarizes the effectiveness results of users by each user group (evaluation point S1 of Table 4.4). As can be observed, there were positive results concerning the achievements of the tasks' goals since 76.95% of users managed to reach the intended goals. Full results are detailed in table B.5 of Appendix B - Usability Tests Results.

By observing users using the interface, the new sub-editors layout applied was accepted positively by the users, even the ones who were more accustomed to the previous tab layout. Although, at a first glance, some users claimed feeling overloaded with information, they mentioned, that seeing all query structure in a short visualization area above was useful. Other feedback pointed out was that it could be useful to resize the dimensions of each sub-editor area.

Regarding the query result preview, a set of users did not manage to comprehend which attributes belong to the query output against the ones that were only a preview to understand the aggregations or group bys applied [2]. Even though it was difficult to distinguish the two elements of the query result, since it was used a low expressiveness in terms of color, this constraint will be taken into account in the next design iteration since more styling and color resources will be available.

[2]As mentioned in section 2.2.2.2 and exemplified in Figure 2.10, the query output preview not only presents the attributes that belong to the query output but also a preview of the data related to the aggregated attributes.

Table 6.2: **Evaluation point S8:** Readability of the foreign keys used to join entities that could be joined using different attributes. (Paper Prototype usability tests - 15 users)

Foreign Key Identification	Identified	Did not Identify	Did not see the Join
OutSystems Developer	86.67%	13.33%	0.00%
Software Developer	66.67%	33.33%	0.00%
Citizen Developer	80.00%	20.00%	0.00%
Total	77.78%	22.22%	0.00%

As predicted in the design and implementation phases of this prototype, the completely redesigned sources editor triggered different points of view and interesting feedback for the next iteration design phase.

Therefore, the following points describe the information collected with respect to the users' comprehension of this new representation:

- Some users misinterpreted the joins of some queries due to the indentation used to represent them. For instance, in the example shown in Figure 6.18, some users indicated that "Office"was joined with "Department", when "Office"was joined with "Employee"(evaluation point S5 of Table 4.4);
- Some users did not manage to understand what the foreign key label was, until they need to select one of them in the formulation scenario. That way, the representation of this element should be reevaluated to make it more clear. Nevertheless, the results concerning the identification of the foreign key, presented in Table 6.2, are positive since considering all user groups 77.78% of users have identified correctly the foreign key in the causes that entities could be joined using different attributes (evaluation point S8 of Table 4.4). Therefore, although the positive results, the lack of intuitiveness regarding the foreign key presentation perceived by users when they observed the interface at the first sight was considered as an aspect to improve in the following design iteration;
- Regarding the join conditions simplification, it was identified an increased awareness, when compared to the existing interface, to perceive that there were two representations of join conditions, as shown in Figure 6.13. The majority of users did not manage to understand the meaning of these different visual representations, which intended to differentiate joins automatically generated from the ones manually edited, as Table 6.3 points out since only 26.67% of users have identified clearly the join purpose (evaluation point S7 of Table 4.4).

Furthermore, the following aspects were captured from observing users trying to formulate queries:

- As expected, the foreign key selection method presented in Figure 6.12 has optimized the effectiveness of the formulation scenario (evaluation point S22 of Table

Table 6.3: **Evaluation point S7:** Readability rate of the join which represents the case where two entities were merged using a left join and the conditions specified the that the primary key of the right enitity must be null. (Paper Prototype usability tests - 15 users)

Left Join with Null Identifier Comprehension	Identified	Did not Identify	Did not see the Join
OutSystems Developer	40.00%	60.00%	0.00%
Software Developer	20.00%	80.00%	0.00%
Citizen Developer	20.00%	80.00%	0.00%
Total	26.67%	73.33%	0.00%

4.4). Since users were asked to choose between the existing foreign key, they were able to select the correct one. The results presented in Table 6.4 illustrates the effect of the improvement made in the interface since all users have selected the correct foreign key to indicate the properly join the formulation case that was necessary to add a join to merge two entities that could be joined using three different foreign keys;

- Concerning the two new buttons to add aggregation functions, group bys, and calculated attributes, it was concluded that, the results have enhanced, but some drawback aspects were identified (evaluation points S14 and S24 of Table 4.4). Users who have never used the system, found these features easy since the two buttons became visible in the interface. However, they continue to make misunderstand when they needed to use a calculated attribute or the aggregation function:
  - Insert Calculated Attribute: The users' acceptance to the new options provided to insert calculated attributes was positive. Not only users who not have already used the query builder have less difficulty to create new attributes, since only 20% of Software Developers and 40% of Citizen Developers had difficulty to find the options and the remaining ones managed to added it intuitively, as detailed in Table 6.5, but also the new options provided were used by OutSystems Developers who are accustomed to use the previous query builder interface. As Table 6.6 demonstrates, the new button added on the bottom of the sources view was used by 86.67% of users. Besides, even 80% of the OutSystems Developers, who knew the existence of a button on the final of the query result table, preferred to use the new option to add the attribute, so that the acceptance of that changes was positive not only to mitigate learnability problems of users who had not found the option before in the existing interface, but also for expert users that preferred this option more accessible.

Finally, some users referred that they felt a reduced necessity to confer the Data Model while they were building queries, since the new sources representation shown how entities were related with each other

Table 6.4: **Evaluation point S22:** Selection of the foreign key, in the formulation scenario, used to join entities that could be joined using different attributes. (Paper Prototype usability tests - 15 users)

Foreign Key Selection	Selected	Did not Select	Did not Add the Join
OutSystems Developer	100.00%	0.00%	0.00%
Software Developer	100.00%	0.00%	0.00%
Citizen Developer	100.00%	0.00%	0.00%
Total	100.00%	0.00%	0.00%

Table 6.5: **Evaluation point S14:** Success rate of users when they needed to add a calculated attribute. (Paper Prototype usability tests - 15 users)

Insert Calculated Attribute	Added Successfully	Difficulty to Add	Could not Add
OutSystems Developer	100.00%	0.00%	0.00%
Software Developer	80.00%	20.00%	0.00%
Citizen Developer	60.00%	40.00%	0.00%
Total	80.00%	20.00%	0.00%

Table 6.6: **Evaluation point S16:** Options used by users to insert the calculated attribute in the context of the M1 scenario. (Paper Prototype usability tests - 15 users)

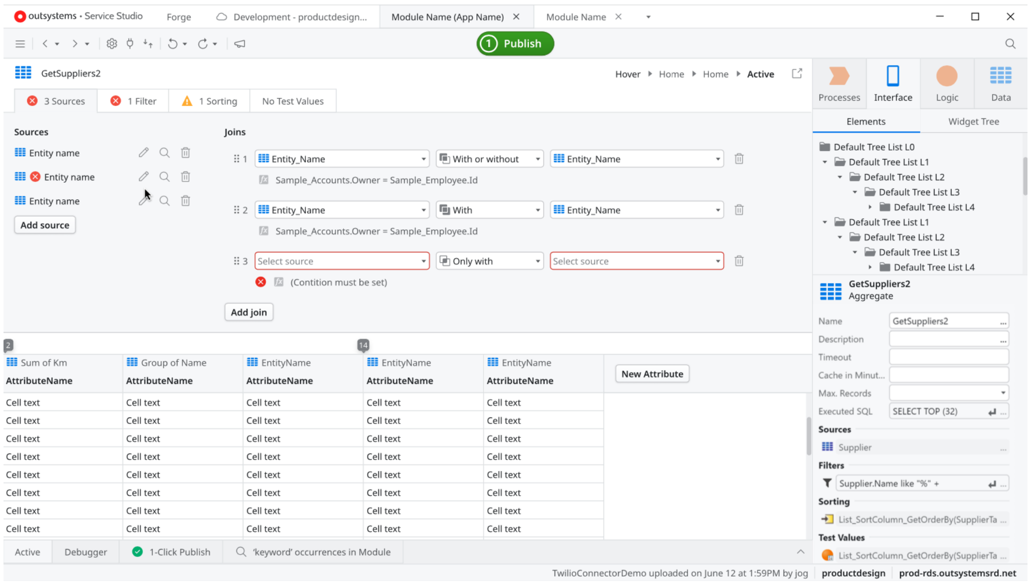
Option used to insert the calculated attribute	Add button on the bottom of the Sources View	Right click in the attributes tree (Sources View)	Add button on the result table (after all attributes)
OutSystems Developer	80.00%	0.00%	20.00%
Software Developer	100.00%	0.00%	0.00%
Citizen Developer	80.00%	20.00%	0.00%
Total	86.67%	6.67%	6.67%

In conclusion, the general success results of the 15 users performing the proposed scenarios were positive although the improvement topics already stated. Therefore, the design implemented in the paper prototype was considered favourable to optimize the usability of the interface for the different user groups analyzed. In that way, the following prototyping iteration will take into account the feedback users provided in this evaluation stage as well as the results obtained.

6.3 Service Studio Implementation

As mentioned in section 2.2.1, Service Studio is the IDE integrated in the OutSystems Platform that allows users to build full applications using the low-code programming paradigm. In that way, the last iteration of the design process, and consequently, the final prototype elaborated in this dissertation, was a functional prototype of the query builder completely integrated into the code of the Service Studio.

Nevertheless, Service Studio was getting a major facelift to achieve a new revamped and refreshed design. Even though the new Service Studio design has not been released yet, the development of the final prototype was decided to be done using the new Service Studio look. In that way, the interface of the query builder developed could be easily integrated later on the OutSystems Platform. Figure 6.20 illustrates the design used as the starting point for the final prototype development. As can be observed, is a redesigned version of the previous Service Studio design illustrated before in Figures 2.3 and 2.5



1 Entity_Name — With or without — Entity_Name
Sample_Accounts.Owner = Sample_Employee.Id

2 Entity_Name — With — Entity_Name
Sample_Accounts.Owner = Sample_Employee.Id

3 Select source — Only with — Select source
(Contition must be set)

Sum of Km	Group of Name	EntityName	EntityName	EntityName
AttributeName	AttributeName	AttributeName	AttributeName	AttributeName
Cell text	Cell text	Cell text	Cell text	Cell text
Cell text	Cell text	Cell text	Cell text	Cell text
Cell text	Cell text	Cell text	Cell text	Cell text
Cell text	Cell text	Cell text	Cell text	Cell text
Cell text	Cell text	Cell text	Cell text	Cell text
Cell text	Cell text	Cell text	Cell text	Cell text
Cell text	Cell text	Cell text	Cell text	Cell text
Cell text	Cell text	Cell text	Cell text	Cell text

Default Tree List L0
Default Tree List L1
Default Tree List L2
Default Tree List L3
Default Tree List L4
Default Tree List L1
Default Tree List L2
Default Tree List L3
Default Tree List L4

GetSuppliers2
Aggregate

Name	GetSuppliers2
Description	
Timeout	
Cache in Minut...	
Max. Records	
Executed SQL	SELECT TOP (32)

Sources
Supplier
Filters
Supplier.Name like "%" +
Sorting
List_SortColumn_GetOrderBy(SupplierTa ...
Test Values
List_SortColumn_GetOrderBy(SupplierTa ...

TwilioConnectorDemo uploaded on June 12 at 1:59PM by jog | productdesign | prod-rds.outsystemsrd.net

Figure 6.20: The new design of the Service Studio and its Visual Query Builder.

Despite the new design of all interface elements and components, the core behaviors and interactions of the interface remained, so the same design and implementation intentions to be applied in this new phase of development were maintained.

6.3.1 Design

Following the same methodology used in the Paper Prototype, the building process of this solution started with the design phase. In this phase, the paper prototype evaluation outcomes were deliberated in order to define and prioritize the aspects to be tackled in the final prototype.

First of all, the source code of the Service Studio was explored in order to plan the development that could be done in the available time. As explained in the presentation of the paper prototype, the development aspects were planned according to their impact on the final interface. Ideas that fundamentally alter the experience of using the interface will be considered a priority, since, is primarily important to perceive how users react to those changes when testing the interface. Minor features and enhancements could also be considered as future work if the new interface solution evaluation show positive results.

Taking into account the evaluation of the Paper Prototype, the sources editor was considered the main focus of the design phase of the Final Prototype. As referred, during the usability tests of the paper prototype, the sources view has revealed positive results, but it was concluded that it should be refined regarding some aspects, mainly the joins representation which were not completely clear for a set of users.

Nevertheless, in order to implement the new interface representation of entities and joins, a considerable implementation effort would be required. The visualization area requires a personalized hierarchical view that should be built from scratch, since there is no similar element in the rest of the Service Studio.

Accordingly, before implementing any idea, a robust design should be established in order to avoid wasting implementation time on doing and redoing ideas. Therefore, more designs were performed, regarding the sources editor of the interface, using the product design system components. Basically, it was used the Figma [46], which is a design tool where all the new Service Studio components were designed, to explore concrete options to represent sources and solve the problems found when users tested the Paper Prototype. By this means, it was possible to obtain a high-fidelity design faster and perceive what solution could most fit the existing requirements.

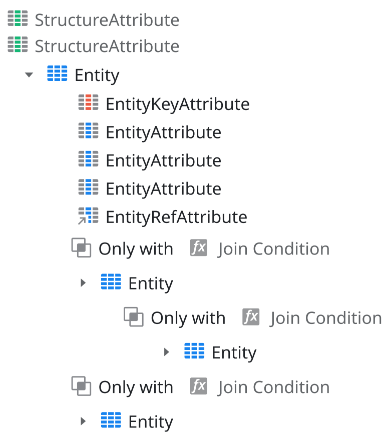
After exploring multiple alternatives, the options illustrated in Figure 6.21 were considered the most valuable. The main difference between them is the way joins are represented. In Option A, the join is represented in a single line placed between the two entities involved. In Option B, the join condition is represented after the second entity involved in the join. Lastly, in Option C, the second entity is presented on the right of the join kind.

In order to decide an option, the following topics were taken into account:

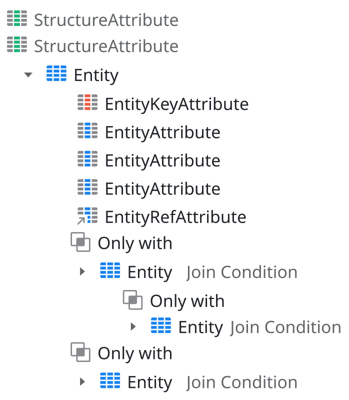
- **Reading flow:** The flow of the entities and joins was considered important, especially due to the representation of the join kind. As this was presented in natural language, users should manage to read sequentially, first the entity, than the join, and after the second entity, all in a fluid way. Due to the arrangement adopted, the Option C was considered the most clear regarding this aspect since the second entity is placed in the right side of the join kind. In that way, the users reading affordance will unconsciously help them to understand the reading flow, since becomes more natural due to the similarities with natural languages;

- **Width and height required:** The space that each option would occupy was important to analyze for the accommodation in the interface. Comparing the options presented, it can be concluded that the Option A and Option B would require more height, and the Option C more width;

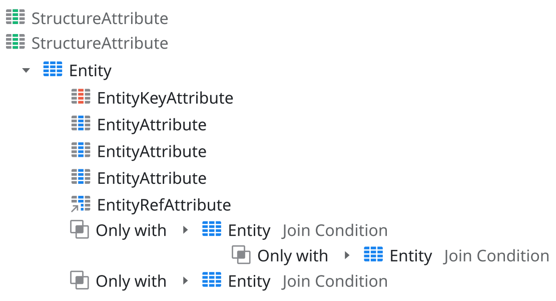
- **Emphasis:** The used representation may end up giving more prominence to certain elements even though they were not built for that purpose. Accordingly, it was verified the aspects of the interface which would stand out at a first glance. In Option A and Option B, joins are more highlighted since there are specific lines to



(a) Option A



(b) Option B



(c) Option C

Figure 6.21: High-fidelity design ideas built in Figma to present the sources of the query.

represent them. Conversely, in the Option C the join is presented in the same line of an entity, which makes entities stand out, in general. Therefore, the Option C was considered the most appropriated to highlight entities at a first sight.

Reflecting on the topics presented, the evaluation of the advantages and disadvantages of each option was conducted, the Option C was elected, as preferred, since it provides a clear readability and highlights the entities used in the query. Despite it requires more width space in the interface, mainly if there are multiple joins nested, these disadvantages could be mitigated in the future if the join conditions would be simplified or represented in a different way.

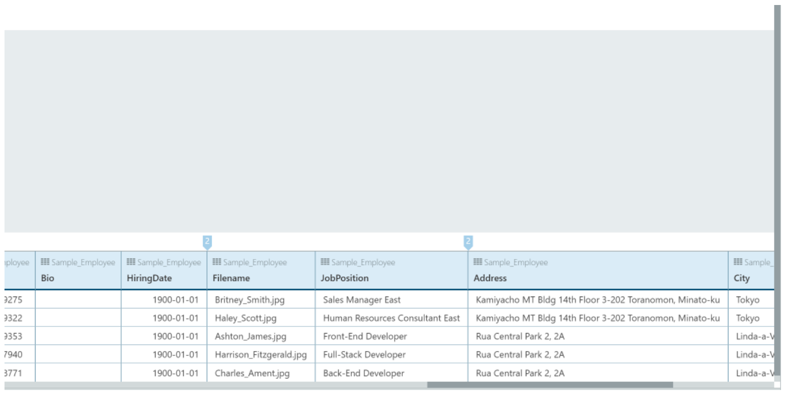
Regarding the remaining elements of the interface, the design approaches applied in the Paper Prototype improved the usability of the interface. As a result, the design concerning filters, sorting and the query result table have followed the ideas already detailed in section 6.2.

6.3.2 Implementation

Bearing in mind the ideas explored in the paper prototype, and taking in account the evaluation performed with users, as well as, the high-fidelity design ideas built in Figma, the system started to be implemented in React [69], Typescript [55] and C#.

General layout:

Following the methodology used in the previous prototype, the first stage of the development process was to switch the tabs by an equivalent edition area with the three components visible at once (sources, filters, and sorting), as can be observed in Figure 6.23. Also, the existing scroll problem identified, illustrated in Figure 6.22, (i.e., when users wanted to scroll the table horizontally to see the columns in overflow, the edition area above disappears) was solved. As a result, they are able to see all the components of the edition area and the information in the query result table at the same time. This change was important to reduce the users' working memory overload since they have the information they need visible.



ployee	Sample_Employee Bio	Sample_Employee HiringDate	Sample_Employee Filename	Sample_Employee JobPosition	Sample_Employee Address	Sample_ City
9275		1900-01-01	Britney_Smith.jpg	Sales Manager East	Kamiyacho MT Bldg 14th Floor 3-202 Toranomon, Minato-ku	Tokyo
9322		1900-01-01	Haley_Scott.jpg	Human Resources Consultant East	Kamiyacho MT Bldg 14th Floor 3-202 Toranomon, Minato-ku	Tokyo
9353		1900-01-01	Ashton_James.jpg	Front-End Developer	Rua Central Park 2, 2A	Linda-a-V
7940		1900-01-01	Harrison_Fitzgerald.jpg	Full-Stack Developer	Rua Central Park 2, 2A	Linda-a-V
3771		1900-01-01	Charles_Ament.jpg	Back-End Developer	Rua Central Park 2, 2A	Linda-a-V

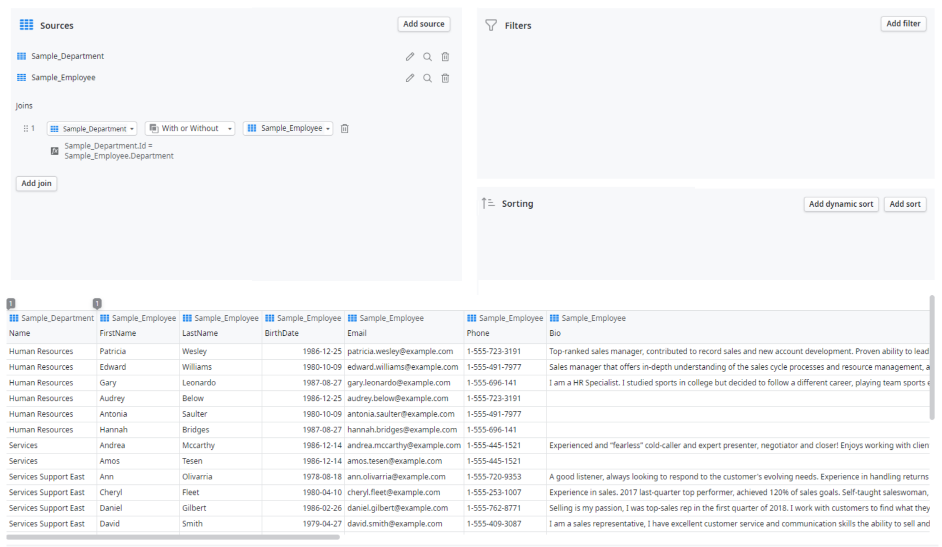
Figure 6.22: Horizontal scroll problem in the existing interface: when users use scroll the see the columns at the right side, the editors are not fixed above.

Sources View:

As approached in the section above, the sources and joins editor has been completely redesigned. Since the design elaborated for this interface area presents multiple components using some indentation and nesting, the React components were structured before the implementation, they were organized, as shown in Figure 6.24, in order to guarantee the feasibility of the arrangement created.

The *"SourcesAndJoinsEditor"* is the parent component that contains all query sources. This component includes two areas: a list of *"DataSetAttribute"*, which are the attributes of the query result added throughout the query formulation, such as aggregated attributes, group bys, or calculated attributes, and a component *"Sources"* that represents all entities and joins of the query.

As the presentation of the entities and joins depends on the joins and the entities involved in the query, due to the indentation and nesting applied whenever an entity is


Sources

Add source

Sample_Department

Sample_Employee

Joins

1 Sample_Department With or Without Sample_Employee

Sample_Department.Id =
Sample_Employee.Department

Add join

Filters

Add filter

Sorting

Add dynamic sort

Add sort

Sample_Department	Sample_Employee	Sample_Employee	Sample_Employee	Sample_Employee	Sample_Employee	Sample_Employee
Name	FirstName	LastName	BirthDate	Email	Phone	Bio
Human Resources	Patricia	Wesley	1986-12-25	patricia.wesley@example.com	1-555-723-3191	Top-ranked sales manager, contributed to record sales and new account development. Proven ability to lead
Human Resources	Edward	Williams	1980-10-09	edward.williams@example.com	1-555-491-7977	Sales manager that offers in-depth understanding of the sales cycle processes and resource management, a
Human Resources	Gary	Leonardo	1987-08-27	gary.leonardo@example.com	1-555-696-141	I am a HR Specialist. I studied sports in college but decided to follow a different career, playing team sports e
Human Resources	Audrey	Below	1986-12-25	audrey.below@example.com	1-555-723-3191	
Human Resources	Antonia	Saulter	1980-10-09	antonia.saulter@example.com	1-555-491-7977	
Human Resources	Hannah	Bridges	1987-08-27	hannah.bridges@example.com	1-555-696-141	
Services	Andrea	Mccarthy	1986-12-14	andrea.mccarthy@example.com	1-555-445-1521	Experienced and “fearless” cold-caller and expert presenter, negotiator and closer! Enjoys working with clien
Services	Amos	Tesen	1986-12-14	amos.tesen@example.com	1-555-445-1521	
Services Support East	Ann	Olivarria	1978-08-18	ann.olivarria@example.com	1-555-720-9353	A good listener, always looking to respond to the customer's evolving needs. Experience in handling returns
Services Support East	Cheryl	Fleet	1980-04-10	cheryl.fleet@example.com	1-555-253-1007	Experience in sales. 2017 last-quarter top performer, achieved 120% of sales goals. Self-taught saleswoman,
Services Support East	Daniel	Gilbert	1986-02-26	daniel.gilbert@example.com	1-555-762-8771	Selling is my passion, I was top-sales rep in the first quarter of 2018. I work with customers to find what they
Services Support East	David	Smith	1979-04-27	david.smith@example.com	1-555-409-3087	I am a sales representative, I have excellent customer service and communication skills the ability to sell and

Figure 6.23: Interface after removed the tabs and expanded them in three areas visible at the same time.

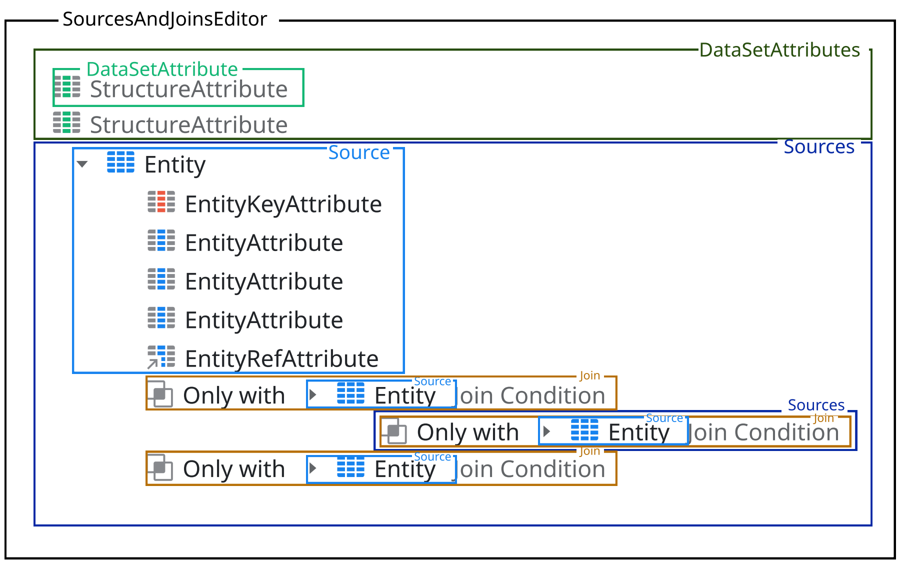
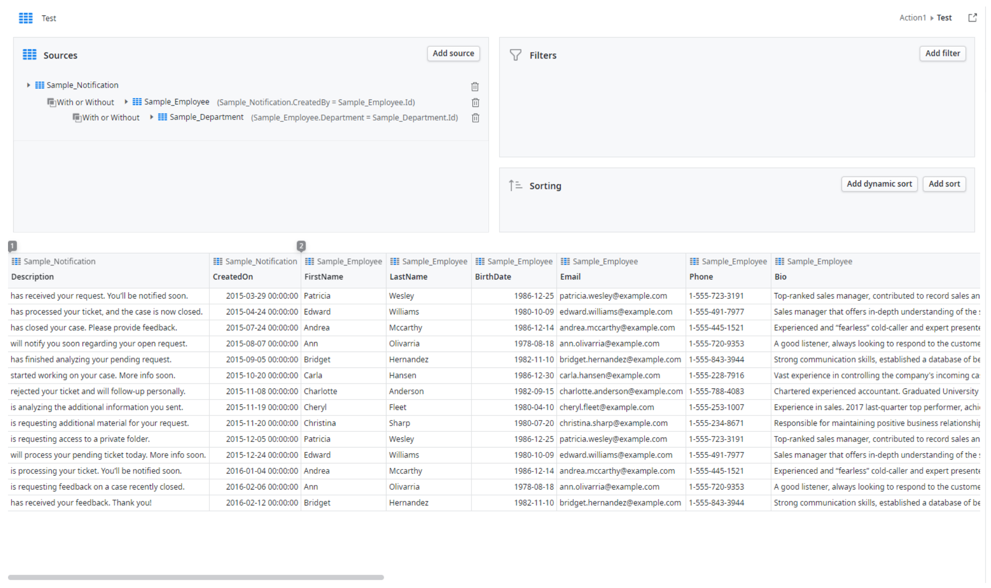


Figure 6.24: Structure of the React components created to dispose the sources.

joined with another already presented above, it was necessary to create some logic in the *"Sources"* component to present the entities with the design intended.

Therefore, the *"Sources"* renders the first entity added to the query, and after that, it iterates along the joins of the query to represent the ones that are related to the entity already rendered. For each join related to an entity, it is presented the join specification (i.e. join kind and join condition), however when the join is being generated, it is verifies if the entity joined to the previous one is related to more entities. In those cases, other *"Sources"* is rendered to represent the remaining sources. Figure 6.25 exemplifies the result of the interface implemented at this stage.



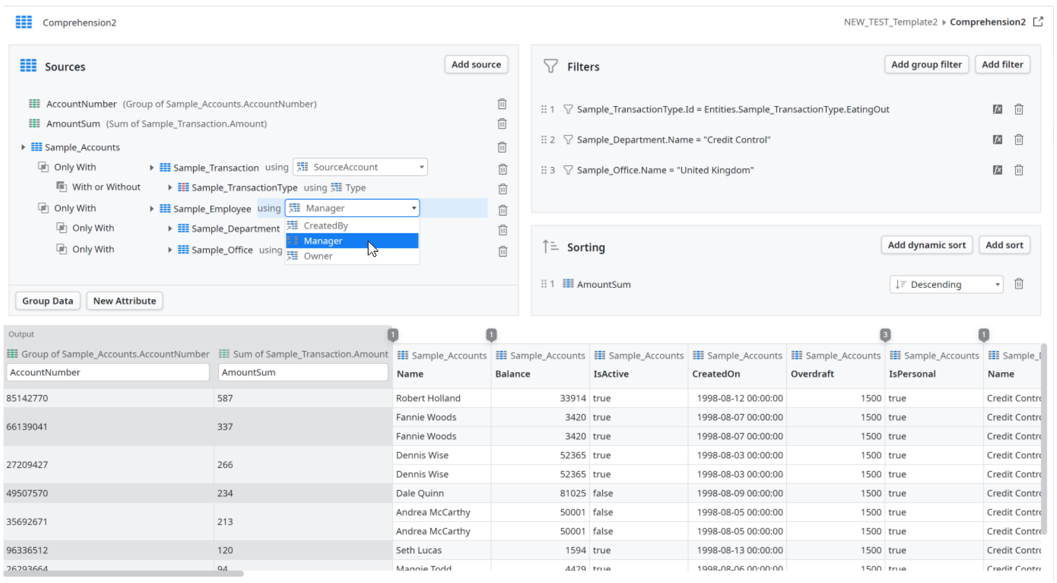
Sample_Notification
With or Without ▸ Sample_Employee (Sample_Notification.CreatedBy = Sample_Employee.Id)
With or Without ▸ Sample_Department (Sample_Employee.Department = Sample_Department.Id)

Sample_Notification	Sample_Notification	Sample_Employee	Sample_Employee	Sample_Employee	Sample_Employee	Sample_Employee	Sample_Employee
Description	CreatedOn	FirstName	LastName	BirthDate	Email	Phone	Bio
has received your request. You'll be notified soon.	2015-03-29 00:00:00	Patricia	Wesley	1986-12-25	patricia.wesley@example.com	1-555-723-3191	Top-ranked sales manager, contributed to record sales an
has processed your ticket, and the case is now closed.	2015-04-24 00:00:00	Edward	Williams	1980-10-09	edward.williams@example.com	1-555-491-7977	Sales manager that offers in-depth understanding of the
has closed your case. Please provide feedback.	2015-07-24 00:00:00	Andrea	Mccarthy	1986-12-14	andrea.mccarthy@example.com	1-555-445-1521	Experienced and "fearless" cold-caller and expert presente
will notify you soon regarding your open request.	2015-08-07 00:00:00	Ann	Olivarria	1978-08-18	ann.olivarria@example.com	1-555-720-9353	A good listener, always looking to respond to the custome
has finished analyzing your pending request.	2015-09-05 00:00:00	Bridget	Hernandez	1982-11-10	bridget.hernandez@example.com	1-555-843-3944	Strong communication skills, established a database of be
started working on your case. More info soon.	2015-10-20 00:00:00	Carla	Hansen	1986-12-30	carla.hansen@example.com	1-555-228-7916	Vast experience in controlling the company's incoming ca
rejected your ticket and will follow-up personally.	2015-11-08 00:00:00	Charlotte	Anderson	1982-09-15	charlotte.anderson@example.com	1-555-788-4083	Chartered experienced accountant. Graduated University
is analyzing the additional information you sent.	2015-11-19 00:00:00	Cheryl	Fleet	1980-04-10	cheryl.fleet@example.com	1-555-253-1007	Experience in sales. 2017 last-quarter top performer, achi
is requesting additional material for your request.	2015-11-20 00:00:00	Christina	Sharp	1980-07-20	christina.sharp@example.com	1-555-234-8671	Responsible for maintaining positive business relationshi
is requesting access to a private folder.	2015-12-05 00:00:00	Patricia	Wesley	1986-12-25	patricia.wesley@example.com	1-555-723-3191	Top-ranked sales manager, contributed to record sales an
will process your pending ticket today. More info soon.	2015-12-24 00:00:00	Edward	Williams	1980-10-09	edward.williams@example.com	1-555-491-7977	Sales manager that offers in-depth understanding of the
is processing your ticket. You'll be notified soon.	2016-01-04 00:00:00	Andrea	Mccarthy	1986-12-14	andrea.mccarthy@example.com	1-555-445-1521	Experienced and "fearless" cold-caller and expert presente
is requesting feedback on a case recently closed.	2016-02-06 00:00:00	Ann	Olivarria	1978-08-18	ann.olivarria@example.com	1-555-720-9353	A good listener, always looking to respond to the custome
has received your feedback. Thank you!	2016-02-12 00:00:00	Bridget	Hernandez	1982-11-10	bridget.hernandez@example.com	1-555-843-3944	Strong communication skills, established a database of be

Figure 6.25: Example of the sources view implemented at this stage.

Nevertheless, as referred in section 6.2.2, the join conditions which were generated automatically by the system were simplified. In those cases, the join conditions, on the existing interface, presented the primary key of an entity equal to a foreign key of the other entity, however since the characteristics of this formulation shown drawbacks for the users fully comprehension, it was designed a simplified version of the join condition. In this simplified version, only the foreign key is presented. If the join between the two entities could be made through different foreign keys, the foreign key used is presented in a dropdown, which allows users to change to another key if they intend. An example of a query with these dropdowns is presented in Figure 6.26. Through that approach, users not only can change that query aspect faster but also the data model could be perceived without consulting it directly, since the dropdown is an indicator that there are multiple attributes that could be used to join the two entities.

Furthermore, for users who open a query already created, it is important to perceive if any join condition was edited manually before in order to mitigate miscomprehensions of the query purpose. In that way, it was decided to represent distinctively, the conditions for these joins, whose condition was edited manually, in order to differentiate them from



Sources:

AccountNumber (Group of Sample_Accounts.AccountNumber)
AmountSum (Sum of Sample_Transaction.Amount)
Sample_Accounts
- Only With — Sample_Transaction using SourceAccount
  - With or Without — Sample_TransactionType using Type
- Only With — Sample_Employee using Manager
  - CreatedBy
  - Manager
  - Owner
  - Only With — Sample_Department
  - Only With — Sample_Office using

Filters:

1 Sample_TransactionType.Id = Entities.Sample_TransactionType.EatingOut
2 Sample_Department.Name = "Credit Control"
3 Sample_Office.Name = "United Kingdom"

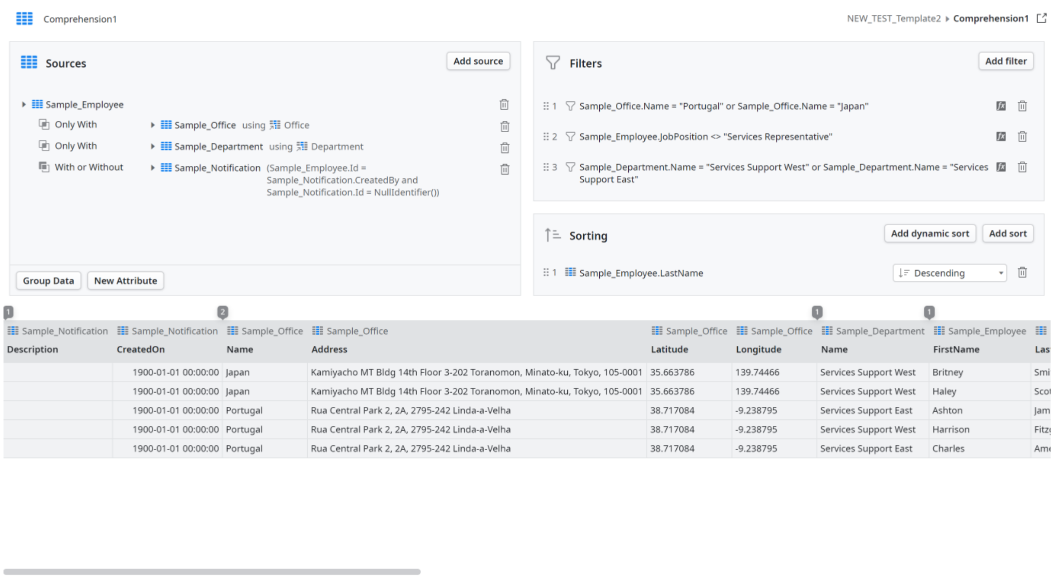
Sorting:

1 AmountSum — Descending

Group of Sample_Accounts.AccountNumber	Sum of Sample_Transaction.Amount	Sample_Accounts	Sample_Accounts	Sample_Accounts	Sample_Accounts	Sample_Accounts	Sample_Accounts	Sample_
AccountNumber	AmountSum	Name	Balance	IsActive	CreatedOn	Overdraft	IsPersonal	Name
85142770	587	Robert Holland	33914	true	1998-08-12 00:00:00	1500	true	Credit Contr
66139041	337	Fannie Woods	3420	true	1998-08-07 00:00:00	1500	true	Credit Contr
		Fannie Woods	3420	true	1998-08-07 00:00:00	1500	true	Credit Contr
27209427	266	Dennis Wise	52365	true	1998-08-03 00:00:00	1500	true	Credit Contr
		Dennis Wise	52365	true	1998-08-03 00:00:00	1500	true	Credit Contr
49507570	234	Dale Quinn	81025	false	1998-08-09 00:00:00	1500	true	Credit Contr
35692671	213	Andrea McCarthy	50001	false	1998-08-05 00:00:00	1500	true	Credit Contr
		Andrea McCarthy	50001	false	1998-08-05 00:00:00	1500	true	Credit Contr
96336512	120	Seth Lucas	1594	true	1998-08-13 00:00:00	1500	true	Credit Contr
26293664	94	Maggie Todd	4429	true	1998-08-06 00:00:00	1500	true	Credit Contr

Figure 6.26: Example of a query were there are joins that could be applied using different foreign keys to merge the two entities, so the user can use the dropdown to choose the intended one.

the ones generated automatically. Figure 6.27 shows an example of a query where a join condition was edited manually.



Sources:

Sample_Employee
- Only With — Sample_Office using Office
- Only With — Sample_Department using Department
- With or Without — Sample_Notification (Sample_Employee.Id = Sample_Notification.CreatedBy and Sample_Notification.Id = NullIdentifier())

Filters:

1 Sample_Office.Name = "Portugal" or Sample_Office.Name = "Japan"
2 Sample_Employee.JobPosition <> "Services Representative"
3 Sample_Department.Name = "Services Support West" or Sample_Department.Name = "Services Support East"

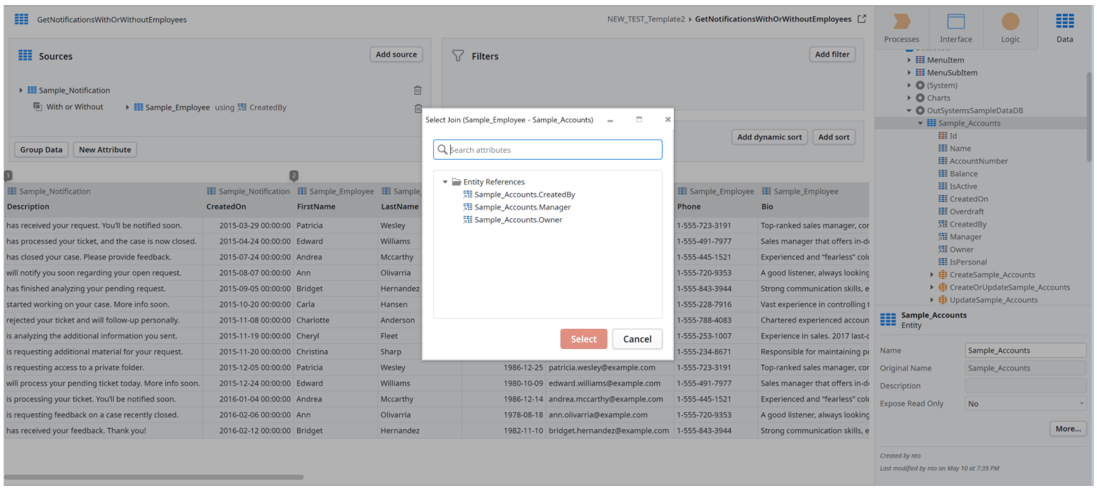
Sorting:

1 Sample_Employee.LastName — Descending

Sample_Notification	Sample_Notification	Sample_Office	Sample_Office	Sample_Office	Sample_Office	Sample_Department	Sample_Employee	
Description	CreatedOn	Name	Address	Latitude	Longitude	Name	FirstName	Las
	1900-01-01 00:00:00	Japan	Kamiyacho MT Bldg 14th Floor 3-202 Toranomon, Minato-ku, Tokyo, 105-0001	35.663786	139.74466	Services Support West	Britney	Smi
	1900-01-01 00:00:00	Japan	Kamiyacho MT Bldg 14th Floor 3-202 Toranomon, Minato-ku, Tokyo, 105-0001	35.663786	139.74466	Services Support West	Haley	Scot
	1900-01-01 00:00:00	Portugal	Rua Central Park 2, 2A, 2795-242 Linda-a-Velha	38.717084	-9.238795	Services Support East	Ashton	Jam
	1900-01-01 00:00:00	Portugal	Rua Central Park 2, 2A, 2795-242 Linda-a-Velha	38.717084	-9.238795	Services Support West	Harrison	Fitz
	1900-01-01 00:00:00	Portugal	Rua Central Park 2, 2A, 2795-242 Linda-a-Velha	38.717084	-9.238795	Services Support East	Charles	Ame

Figure 6.27: Example of a query that contains a join condition edited manually: The join condition between *"Sample_Employee"* and *"Sample_Notification"* was edited so this condition was not simplified as the other ones.

Finally on this point concerning the foreign keys of the joins, there was a problem in the existing interface which could lead users to make semantic errors while they are formulating queries that uses entities which could be joined using different foreign keys. As explained in section 6.2.2, the existing interface did not ask users which foreign key he wants to use. Consequently, the idea adopted in the paper prototype (Figure 6.12), which had leveraged the results for the user testing in a positive way, continued to this phase, where a modal appears in cases where there are multiple foreign keys, asking user which attribute he would want to use to merge the entities, as demonstrated in Figure 6.28.


GetNotificationsWithOrWithoutEmployees

NEW_TEST_Template2 ▸ GetNotificationsWithOrWithoutEmployees

Sources — Add source

- Sample_Notification
  - With or Without ▸ Sample_Employee using CreatedBy

Group Data | New Attribute

Filters — Add filter

Add dynamic sort | Add sort

Select Join (Sample_Employee - Sample_Accounts)

Search attributes

- Entity References
  - Sample_Accounts.CreatedBy
  - Sample_Accounts.Manager
  - Sample_Accounts.Owner

Select | Cancel

Sample_Notification	Sample_Notification	Sample_Employee	Sample_			Sample_Employee	Sample_Employee
Description	CreatedOn	FirstName	LastName			Phone	Bio
has received your request. You'll be notified soon.	2015-03-29 00:00:00	Patricia	Wesley			1-555-723-3191	Top-ranked sales manager, cor
has processed your ticket, and the case is now closed.	2015-04-24 00:00:00	Edward	Williams			1-555-491-7977	Sales manager that offers in-d
has closed your case. Please provide feedback.	2015-07-24 00:00:00	Andrea	Mccarthy			1-555-445-1521	Experienced and "fearless" col
will notify you soon regarding your open request.	2015-08-07 00:00:00	Ann	Olivarria			1-555-720-9353	A good listener, always looking
has finished analyzing your pending request.	2015-09-05 00:00:00	Bridget	Hernandez			1-555-843-3944	Strong communication skills, e
started working on your case. More info soon.	2015-10-20 00:00:00	Carla	Hansen			1-555-228-7916	Vast experience in controlling t
rejected your ticket and will follow-up personally.	2015-11-08 00:00:00	Charlotte	Anderson			1-555-788-4083	Chartered experienced accoun
is analyzing the additional information you sent.	2015-11-19 00:00:00	Cheryl	Fleet			1-555-253-1007	Experience in sales. 2017 last-c
is requesting additional material for your request.	2015-11-20 00:00:00	Christina	Sharp			1-555-234-8671	Responsible for maintaining p
is requesting access to a private folder.	2015-12-05 00:00:00	Patricia	Wesley	1986-12-25	patricia.wesley@example.com	1-555-723-3191	Top-ranked sales manager, cor
will process your pending ticket today. More info soon.	2015-12-24 00:00:00	Edward	Williams	1980-10-09	edward.williams@example.com	1-555-491-7977	Sales manager that offers in-d
is processing your ticket. You'll be notified soon.	2016-01-04 00:00:00	Andrea	Mccarthy	1986-12-14	andrea.mccarthy@example.com	1-555-445-1521	Experienced and "fearless" col
is requesting feedback on a case recently closed.	2016-02-06 00:00:00	Ann	Olivarria	1978-08-18	ann.olivarria@example.com	1-555-720-9353	A good listener, always looking
has received your feedback. Thank you!	2016-02-12 00:00:00	Bridget	Hernandez	1982-11-10	bridget.hernandez@example.com	1-555-843-3944	Strong communication skills, e

- MenuItem
- MenuSubItem
- (System)
- Charts
- OutSystemsSampleDataDB
  - Sample_Accounts
    - Id
    - Name
    - AccountNumber
    - Balance
    - IsActive
    - CreatedOn
    - Overdraft
    - CreatedBy
    - Manager
    - Owner
    - IsPersonal
    - CreateSample_Accounts
    - CreateOrUpdateSample_Accounts
    - UpdateSample_Accounts

Sample_Accounts
Entity

Name	Sample_Accounts
Original Name	Sample_Accounts
Description	
Expose Read Only	No

More...

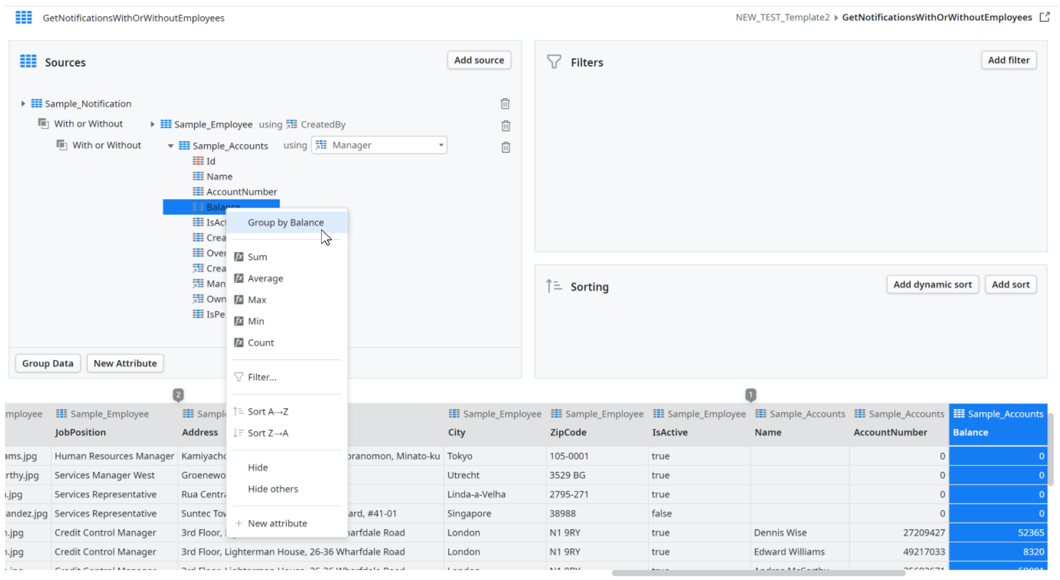
Created by nto
Last modified by nto on May 10 at 7:39 PM

Figure 6.28: Modal that asks user which foreign key attribute he wants to use to merge the two entities. In the example illustrated, the user has already added *"Sample_Notification"* and *"Sample_Employee"*, and when he added *"Sample$_A$ccount"* the interface showed the modal presented since there are three references of employees on the account entity: *"CreatedBy"*, *"Manager"*, and *"Owner"*.

Moreover, the *"Source"* component was changed to present every entity as a tree node which contains the name and icon of each attribute as a child. Besides, not only a context menu, available when users right-click in each entity attribute, was added, as Figure 6.29 exemplifies, but also an option to click on the attribute to highlight it on the query result preview. In that way, there is a possibility to find and add query options easier, since the table auto-direct the user to the intended attribute, promoting a more quick right-click on the attribute to apply aggregation functions, group bys, filters, sorting, or even to hide it.

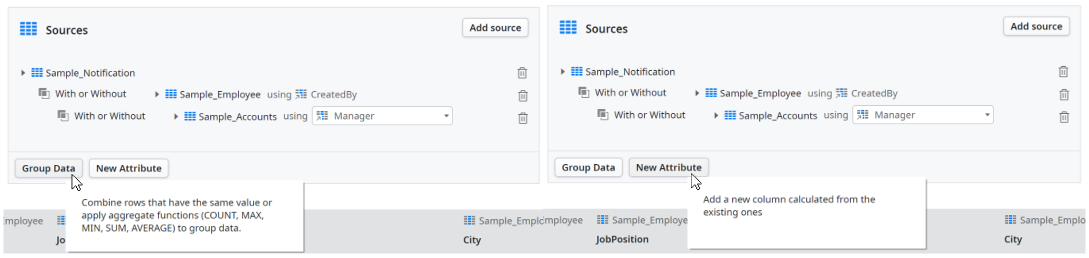
New alternatives to add and search for attributes:

Nevertheless, these options not only are interesting alternatives to access these functionalities of the query builder, but also add a visible way to access these options, which will improve the adoption of new users when they try to use the system. As stated before, these options were added in the Paper Prototype, as illustrated in 6.15 and 6.16. However, as mentioned in section 6.2.3, some users did not understand if they would need to add a calculated attribute or an aggregated attribute. Consequently, the button labels were changed to provide a more imperative and informative message related to the action accessible through each button. In that way, "Add Aggregation / Group By"was changed to "Group Data"and "Add Calculated Attribute"was changed to "New Attribute", to reinforce that using the "Group Data"button users can combine data of an attribute



Sample_Employee	Sample_Employee	Sample_Employee	Sample_Employee	Sample_Employee	Sample_Employee	Sample_Accounts	Sample_Accounts	Sample_Accounts
	JobPosition	Address	City	ZipCode	IsActive	Name	AccountNumber	Balance
ams.jpg	Human Resources Manager	Kamiyachc oranomon, Minato-ku	Tokyo	105-0001	true		0	0
rthy.jpg	Services Manager West	Groenewo	Utrecht	3529 BG	true		0	0
.jpg	Services Representative	Rua Centra	Linda-a-Velha	2795-271	true		0	0
andez.jpg	Services Representative	Suntec Tov ard, #41-01	Singapore	38988	false		0	0
.jpg	Credit Control Manager	3rd Floor, arfdale Road	London	N1 9RY	true	Dennis Wise	27209427	52365
.jpg	Credit Control Manager	3rd Floor, Lighterman House, 26-36 Wharfdale Road	London	N1 9RY	true	Edward Williams	49217033	8320

Figure 6.29: Attributes of each entity and interactions accessible using the right click to trigger the context menu.



Combine rows that have the same value or apply aggregate functions (COUNT, MAX, MIN, SUM, AVERAGE) to group data.

Add a new column calculated from the existing ones

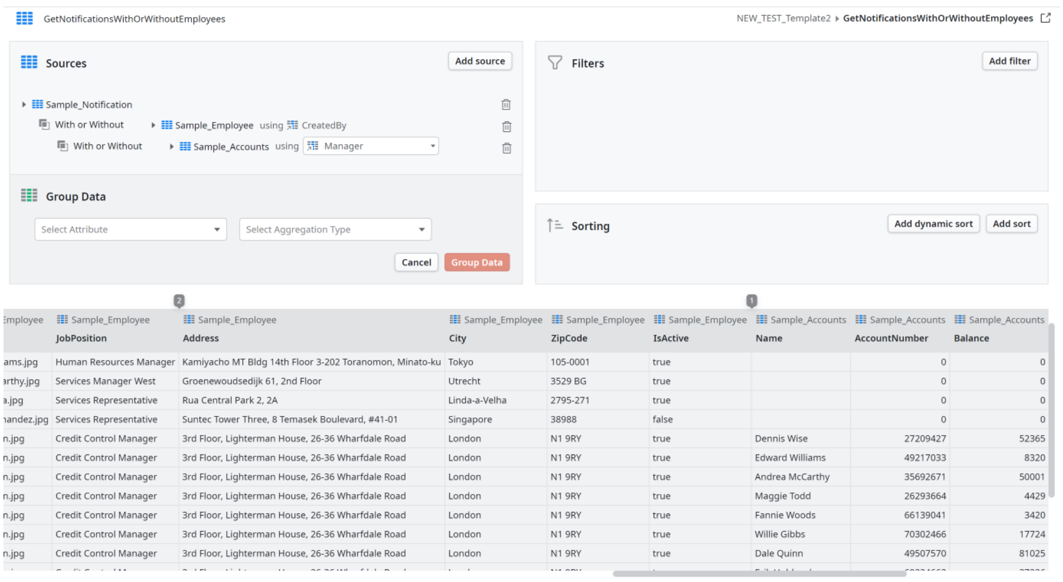
(a) New Attribute (b) Group Data

Figure 6.30: Tooltips of the two new buttons added.

merging its rows, and in "New Attribute", they can add a new attribute calculated from the existing ones to the query result. Besides, tooltips were also added to each button in order to clarify the users who were confused about what function they intend. Figure 6.30 illustrates the two buttons and the respective tooltips added.

When a user click on the "Group Data"button the area, the interface shows a new area where the user can select the attribute and the functions he could apply to the attribute according to the attribute data type. Figure 6.31 demonstrates this new designed functionality of the system. This option is advantageous not only for users who are not familiarized with the system, who can see a visible option to group data, but also is it provides a faster way to add multiple group bys or aggregation functions since in a short area of the interface users can group data of several attributes without the need to search horizontally through the query result.

The "New Attribute"button allows users to add calculated attributes in the same way they could add it before using the button available at the right side of all columns (in most times hidden due to the overflow). However, this button not only is more visible


GetNotificationsWithOrWithoutEmployees — NEW_TEST_Template2 ▸ GetNotificationsWithOrWithoutEmployees

Sources — Add source
- Sample_Notification
  - With or Without ▸ Sample_Employee using CreatedBy
    - With or Without ▸ Sample_Accounts using Manager

Group Data
- Select Attribute
- Select Aggregation Type
- Cancel | Group Data

Filters — Add filter

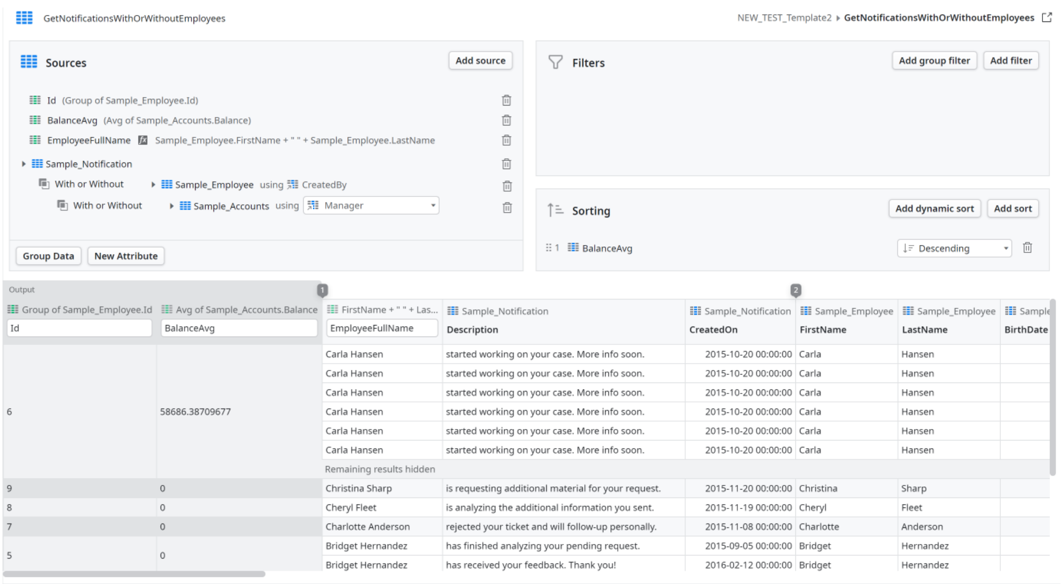
Sorting — Add dynamic sort | Add sort

mployee	Sample_Employee JobPosition	Sample_Employee Address	Sample_Employee City	Sample_Employee ZipCode	Sample_Employee IsActive	Sample_Accounts Name	Sample_Accounts AccountNumber	Sample_Accounts Balance
ams.jpg	Human Resources Manager	Kamiyacho MT Bldg 14th Floor 3-202 Toranomon, Minato-ku	Tokyo	105-0001	true		0	0
arthy.jpg	Services Manager West	Groenewoudsedijk 61, 2nd Floor	Utrecht	3529 BG	true		0	0
a.jpg	Services Representative	Rua Central Park 2, 2A	Linda-a-Velha	2795-271	true		0	0
andez.jpg	Services Representative	Suntec Tower Three, 8 Temasek Boulevard, #41-01	Singapore	38988	false		0	0
n.jpg	Credit Control Manager	3rd Floor, Lighterman House, 26-36 Wharfdale Road	London	N1 9RY	true	Dennis Wise	27209427	52365
n.jpg	Credit Control Manager	3rd Floor, Lighterman House, 26-36 Wharfdale Road	London	N1 9RY	true	Edward Williams	49217033	8320
n.jpg	Credit Control Manager	3rd Floor, Lighterman House, 26-36 Wharfdale Road	London	N1 9RY	true	Andrea McCarthy	35692671	50001
n.jpg	Credit Control Manager	3rd Floor, Lighterman House, 26-36 Wharfdale Road	London	N1 9RY	true	Maggie Todd	26293664	4429
n.jpg	Credit Control Manager	3rd Floor, Lighterman House, 26-36 Wharfdale Road	London	N1 9RY	true	Fannie Woods	66139041	3420
n.jpg	Credit Control Manager	3rd Floor, Lighterman House, 26-36 Wharfdale Road	London	N1 9RY	true	Willie Gibbs	70302466	17724
n.jpg	Credit Control Manager	3rd Floor, Lighterman House, 26-36 Wharfdale Road	London	N1 9RY	true	Dale Quinn	49507570	81025

Figure 6.31: New alternative to group data.

but also is more accessible.

The attributes added through these two buttons over the query formulation process are presented in the "Sources"tab, as illustrated in Figure 6.32. This representation is advantageous since users can read more information about how those attributes were added because there are visible information about the attributes and the formulas used to insert them.


GetNotificationsWithOrWithoutEmployees — NEW_TEST_Template2 ▸ GetNotificationsWithOrWithoutEmployees

Sources — Add source
- Id (Group of Sample_Employee.Id)
- BalanceAvg (Avg of Sample_Accounts.Balance)
- EmployeeFullName Sample_Employee.FirstName + " " + Sample_Employee.LastName
- Sample_Notification
  - With or Without ▸ Sample_Employee using CreatedBy
    - With or Without ▸ Sample_Accounts using Manager

Group Data | New Attribute

Filters — Add group filter | Add filter

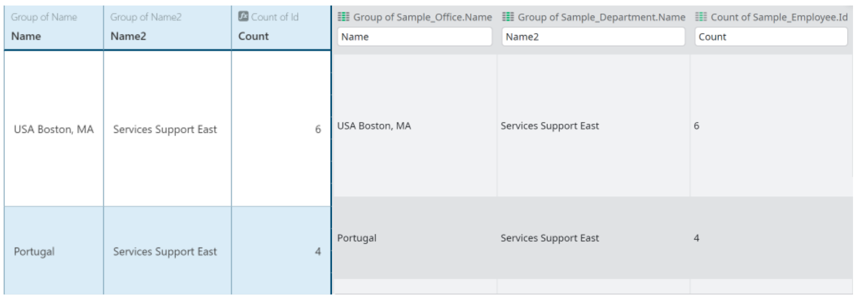
Sorting — Add dynamic sort | Add sort
- 1 BalanceAvg — Descending

Output

Group of Sample_Employee.Id Id	Avg of Sample_Accounts.Balance BalanceAvg	FirstName + " " + Las... EmployeeFullName	Sample_Notification Description	Sample_Notification CreatedOn	Sample_Employee FirstName	Sample_Employee LastName	Sample BirthDate
6	58686.38709677	Carla Hansen	started working on your case. More info soon.	2015-10-20 00:00:00	Carla	Hansen	
		Carla Hansen	started working on your case. More info soon.	2015-10-20 00:00:00	Carla	Hansen	
		Carla Hansen	started working on your case. More info soon.	2015-10-20 00:00:00	Carla	Hansen	
		Carla Hansen	started working on your case. More info soon.	2015-10-20 00:00:00	Carla	Hansen	
		Carla Hansen	started working on your case. More info soon.	2015-10-20 00:00:00	Carla	Hansen	
		Carla Hansen	started working on your case. More info soon.	2015-10-20 00:00:00	Carla	Hansen	
		Remaining results hidden					
9	0	Christina Sharp	is requesting additional material for your request.	2015-11-20 00:00:00	Christina	Sharp	
8	0	Cheryl Fleet	is analyzing the additional information you sent.	2015-11-19 00:00:00	Cheryl	Fleet	
7	0	Charlotte Anderson	rejected your ticket and will follow-up personally.	2015-11-08 00:00:00	Charlotte	Anderson	
5	0	Bridget Hernandez	has finished analyzing your pending request.	2015-09-05 00:00:00	Bridget	Hernandez	
		Bridget Hernandez	has received your feedback. Thank you!	2016-02-12 00:00:00	Bridget	Hernandez	

Figure 6.32: Example of a query that contains a group by (*"Id"*), an aggregation function (*"BalanceAvg"*) and a calculated attribute (*"EmployeeFullName"*).

In the same way, the query result table headers of these attributes were changed not only to improve their readability but also to maintain the consistency with the new



Group of Name Name	Group of Name2 Name2	Count of Id Count
USA Boston, MA	Services Support East	6
Portugal	Services Support East	4

Group of Sample_Office.Name Name	Group of Sample_Department.Name Name2	Count of Sample_Employee.Id Count
USA Boston, MA	Services Support East	6
Portugal	Services Support East	4

(a) Existing Interface (b) Service Studio Prototype

Figure 6.33: Comparison between the table headers of the existing interface and the new ones (two group bys and one aggregated attribute, from the left to the right).

area visible in sources. Figure 6.33 compares the header representation of an aggregated attribute, a group by, and a calculated attribute in the existing interface to the new one. As can be observed, the reference of the source attribute used to group data was added to the group bys and aggregation function, since before it was difficult to perceive which attributes were grouped in the query. Moreover, the function symbol of aggregation functions was removed since no formula is inserted in these attributes and several users tried to click on the function symbol to open some formula.

6.3.3 Evaluation

The final evaluation phase elaborated in the context of this dissertation was the usability tests of the final prototype, using the same methodology and testing scenarios used to test the existing interface and the paper prototype. Accordingly, the Final Prototype was tested with 30 users (10 users of each user group) under a remote environment supported by the Zoom [82] meeting solution and its feature of sharing and recording the presenter's screen.

Understanding the user's profile:

The profile of each user was identified through a survey where users answered some questions concerning their background namely regarding OutSystems, SQL, and data tools. This information was useful to attribute each user to his respective user group according to his knowledge in relational databases and OutSystems development.[3]

The detailed information about the usability test participants' profile are presented in Table B.3 of Appendix B - Usability Tests Results.

Preparation of the test:

The testing process started with a brief explanation of the data model used across the tests. The communication used to explain the data model was adapted according to the relational database knowledge each user has. When users confirmed they comprehended

[3]A complete definition of each user group is provided in section 5.2.

Table 6.7: **Evaluation point S1:** Average of the success rate of all scenarios by user group (Final Prototype usability tests - 30 users).

User Group	Not Achieved	Partially Achieved	Achieved
OutSystems Developer	12.86%	5.71%	81.43%
Software Developer	21.43%	10.00%	68.57%
Citizen Developer	42.00%	8.00%	50.00%
Total	**25.43%**	**7.90%**	**66.67%**

the data model and the aim of the tasks they started using the interface according to the testing scenarios. More details about the evaluation method adopted and the testing scenarios planned were already described in sections 4.5 and 4.4 respectively.

Observation during the test:

During the test, it was performed a direct observation of users interacting with the system to accomplish the tasks proposed in the scenarios planned. Thereby, a spreadsheet was used to manually register all aspects evaluated in each scenario that was previously enumerated in Table 4.4.The results considered relevant to the usability improvement will be presented below.

In summary, the achievement of the goals of each scenario was registered (effectiveness metrics) as well as the time users spent in each scenario (efficiency metrics). Moreover, as there was scenarios built to test specific problems and functionalities of the interface, specific notes and metrics regarding these functionalities studied were also registered.

Results:

In that way, the users tried to accomplish the tasks integrated in the testing scenarios, and all this interaction process was analyzed in order to perceive the difficulties they had as well as to register usability metrics that could allow to characterize the success of the prototype built and compare it with the previous visual querying solution.

Regarding the achievement of the tasks proposed (evaluation point S1 of Table 4.4), Table 6.7 summarizes the effectiveness results of users by each user group. The positive results about the tasks achievement were positive since 66.67% of users managed to reach the intended goals and 7.90% partially achieved the goal. The complete list of results are presented in table B.6 of Appendix B - Usability Tests Results.

The users who use to the previous interface, have reacted positively to the new interface layout without tabs and the three areas always visible. They reinforced that in the previous interface it was very cumbersome to interpret or formulate queries using the tabs since they did not have a query overview anytime.

In general, users who have some relational database foundations managed to comprehend effortlessly the new sources view, which presents all entities in a tree and promotes a more simple and compact view of the entities and joins included in the query, and pointed out that the arrangement used allow to perceive faster the query purpose since it is possible to visualize which entities are related with each other.

The changes applied to the foreign key used in the joins generated automatically have impacted positively the users' success rate performing scenarios that contain these cases. For example, as Table 6.9 details, the 58.89% of users have identified the foreign keys used to merge tables that could be linked using different attributes (evaluation point S8 of Table 4.4). On the other hand, the modal added to ask user which foreign key he intend to use when there are multiple available options to create the automatic join, have improved the effectiveness of the join insertion for these cases (evaluation point S22 of Table 4.4). For instance, Table 6.10 demonstrates the effectiveness of the join specification of the F1 scenario where is necessary to join two entities using a specific foreign key and these two entities have three relationships linking each other. As it can be observed, the results are very positive since 96.67% of users have joined the two entities used the required foreign key. In that way, this change of the interface have leveraged the guidance that the interface could give to users reaching their goals, mainly in these cases where a distraction could lead to query the wrong data, turning the interface less error-prone.

Nevertheless, the approach used to distinguish the joins generated automatically to the ones that were edited manually and have a more complex condition have not revealed positive results (evaluation point S7 of Table 4.4). As Table 6.11 presents, in the use case used to present a join edited manually, only 16.67% of users have successfully interpreter the join purpose.

Regarding the operations which were identified as difficult to find by users who have never used the query builder neither visualize any tutorial video concerning it (add group by, aggregation function or calculated attributes), the majority of users managed to find intuitively the options to access this functions in the scenarios they needed it.

As presented in table 6.8, 83.33% of users added calculated attributes when necessary without difficulties and no user did not manage to add it (evaluation points S14 of Table 4.4). At the same way, users have fewer difficulties to apply aggregation functions and group bys, since the two buttons were visible in the interface and they found them easily. In that way, the main cause for users did not manage to group data was the lack of relational database knowledge.

The new functionality added that allow users to search for a specific attribute and apply operations using the attributes tree was not found intuitively. Several users thought that the "triangle"buttons to expand the attributes of an entity were to add information regarding the join of the entity and the nested one.

However, after ending the test, this functionality was demonstrated to the users and they revealed that is very useful although they did not discover the feature by themselves. In that way, it is necessary to add some introduction to indicate that this functionality is available or study other representations that could be more intuitive for users.

In conclusion, Figure 6.34 summarize the average of the success of each user group in each scenario as well as the total average (evaluation point S1 of Table 4.4). OutSystems

Table 6.8: **Evaluation point S14:** Success rate of users when they needed to add a calculated attribute. (Final Prototype usability tests - 30 users)

Insert Calculated Attribute	Added Successfully	Difficulty to Add	Could not Add
OutSystems Developer	100.00%	0.00%	0.00%
Software Developer	70.00%	30.00%	0.00%
Citizen Developer	80.00%	20.00%	0.00%
Total	83.33%	16.67%	0.00%

Table 6.9: **Evaluation point S8:** Readability of the foreign keys used to join entities that could be joined using different attributes. (Final Prototype usability tests - 30 users)

Foreign Key Identification	Identified	Did not Identify	Did not See the Join
OutSystems Developer	86.67%	13.33%	0.00%
Software Developer	50.00%	50.00%	0.00%
Citizen Developer	40.00%	40.00%	20.00%
Total	58.89%	34.44%	6.67%

Table 6.10: **Evaluation point S22:** Selection of the foreign key, in the formulation scenario, used to join entities that could be joined using different attributes. (Final Prototype usability tests - 30 users)

Foreign Key Selection	Selected	Did not Select	Did not Add the Join
OutSystems Developer	90.00%	0.00%	10.00%
Software Developer	100.00%	0.00%	0.00%
Citizen Developer	100.00%	0.00%	0.00%
Total	96.67%	0.00%	3.33%

Developers achieved 81.43% of the scenarios, Software Developers 68.57%, and Citizen Developers 50%. Considering all user groups, 66.67% of the scenarios were completely achieved and 7.90% have not completely achieved due to the lack of some peculiarities. The success was classified according to the Table 4.5 already presented and the description of user testing scenarios and user groups which are detailed in section 4.4 and in section 5.2 respectively.

Table 6.11: **Evaluation point S7:** Readability rate of the join which represents the case where two entities were merged using a left join and the conditions specified the that the primary key of the right enitity must be null. (Final Prototype usability tests - 30 users)

Left Join with Null Identifier Comprehension	Identified	Did not Identiy	Did not seen the Join
OutSystems Developer	30.00%	70.00%	0.00%
Software Developer	20.00%	70.00%	10.00%
Citizen Developer	0.00%	90.00%	10.00%
Total	16.67%	76.67%	6.67%

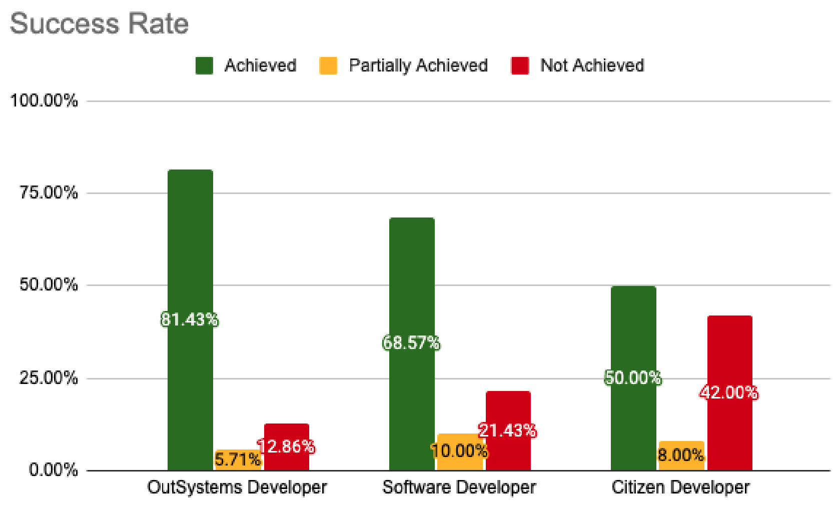


Figure 6.34: **Evaluation point S1:** Average of the success rate of all scenarios by each user group (Final Prototype usability tests - 30 users).

6.4 Results Analysis

Since the goal of this dissertation is the usability improvement of the visual query building interface approached in this study, the results of the usability tests were analyzed regarding three attributes of usability: effectiveness, efficiency, and satisfaction.

In that way, the purpose of this section is to compare the usability of the final prototype with the usability of the existing interface, measured before starting the solution building. Accordingly, the results obtained in usability tests of the existing interface, presented in section 5.1.5, were compared against the results extracted from usability tests of the final Service Studio prototype, presented in section 6.3.3. The test environment was the same for both tests where 30 users (10 of each user group) tested the existing interface and different 30 users (also 10 of each user group) tested the final prototype. Accordingly, this section will present a comparison of the gathered metrics in the evaluation of the two interfaces.

6.4.1 Effectiveness

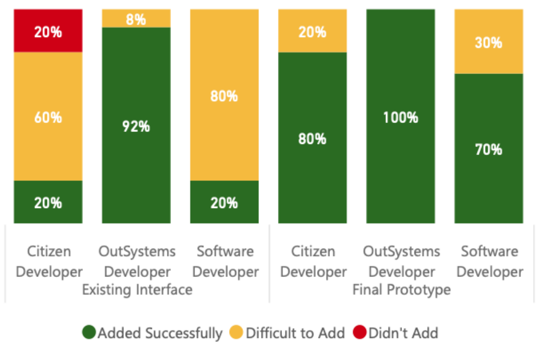
Regarding the problems of the interface, identified throughout the evaluation phases of the iterative design, it was identified that users have improved their success perceiving the details considered critical in the first evaluation. Figure 6.35 illustrates the comparisons of the aspects described below between the existing interface and the final prototype:

- **Insert calculated attribute (Figure 6.35a):** When users performed the scenario which required a calculated attribute insertion, in the existing interface, the users who knew how to do it in the existing interface (OutSystems Developers) managed to insert it, but the majority of other users did not manage to add the calculated attribute or had difficulties discovering how to add it. For instance, in the existing interface 20% of the citizen developers, 92% of the OutSystems Developers and 20% of the software developers added successfully the calculated attribute without difficulty. The results were more positive when users used the final prototype since 80% of citizen developers, 100% of OutSystems developers, and 70% of software developers managed to add successfully the intended calculated attribute without difficulty. In conclusion, the new, and more visible option, mitigated the learnability problem identified in the existing interface, since in the final prototype not only the majority of users could successfully add the calculated attribute but also used the new button to add calculated attributes, as you can confirm in table 6.12;

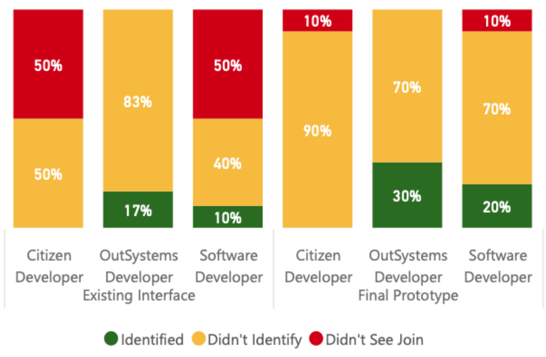
- **Left join with null identifier readability (Figure 6.35b):** The strategy adopted to distinguish the joins generated automatically to the ones edited manually did not fulfill the expected effectiveness. The join presented in scenario C1, to select only the employees who have never created notifications (one entity left joined, with another entity and the condition defining that the primary key of the second entity must be null), was not comprehended by most of users. In spite of the number of users who successfully interpreted this joins have quite duplicated for OutSystems developers (from 17% to 30%) and Software Developers (from 10% to 20%), the values continue to be low. Consequently, the way these joins are presented is an aspect that should be further improved;

- **Foreign key selection (Figure 6.35c):** The modal implemented to force users to choose the foreign key, to join two entities, everytime the foreign key could not be foreseen due to the multiple hypotheses available, have revealed outstanding positive results in the correct join formulation. For the scenario F1, the success rate on adding correctly one of these joins that could be formulated using different foreign keys was 22%, 50%, and 40% (Citizen Developers, OutSystems Developers, and Software Developers, respectively) in the existing interface. The values registered by observing the number of users interacting with the final prototype were much more favorable since 100% of Citizen Developers, 90% of OutSystems Developers, and 100% of Software Developers added these joins correctly;

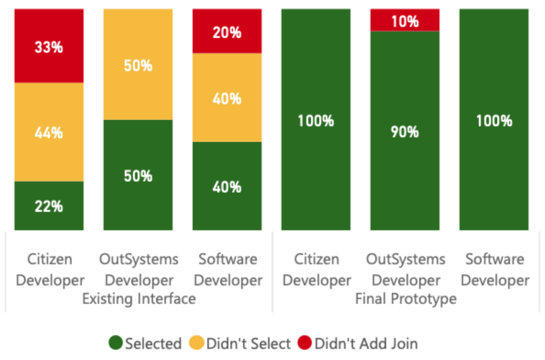
- **Foreign key readability (Figure 6.35d):** The new representation of joins, which highlights the foreign key used, have leveraged the number of users who have correctly interpreted the joins of the query. Concerning the comprehension scenarios C2 and C4 that contained queries that included joins that could have been built by different foreign keys, it is important to highlight the foreign key used to link the entities in order to completely comprehend the query. With the highlight gave to the foreign keys in these cases, through the usage of a dropdown list, the foreign key identification success rate have more than duplicated, passing from 0%, 28% and 20% of Citizen Developers, OutSystems Developers, and Software Developers,



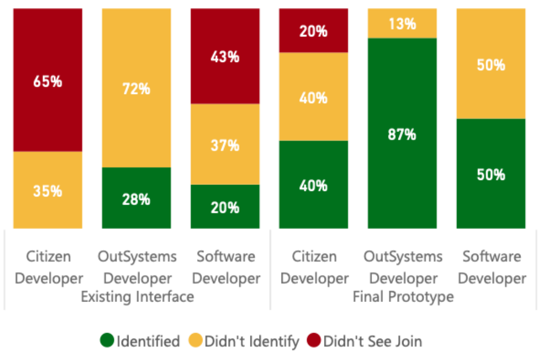
(a) Insert calculated attribute (evaluation point S14)



(b) Left join with null readability (evaluation point S7)



(c) Foreign key selection (evaluation point S22)



(d) Foreign key readability (evaluation point S8)

Figure 6.35: Comparison of the effectiveness between the Existing Interface (on the left side of each chart) and the Final Prototype (on the right side of each chart), regarding specific use cases.

Table 6.12: **Evaluation point S16:** Options used by users to insert the calculated attribute in the context of the M1 scenario. (Final Prototype usability tests - 30 users)

Option used to insert the calculated attribute	Add button on the bottom of the Sources View	Right click in the attributes tree (Sources View)	Add button on the result table (after all attributes)
OutSystems Developer	50.00%	20.00%	30.00%
Software Developer	90.00%	10.00%	0.00%
Citizen Developer	100.00%	0.00%	0.00%
Total	80.00%	10.00%	10.00%

respectively, in the existing interface to a success rate of the correct comprehension of 40%, 87%, and 50%.

The results of all scenarios and all users were combined, in order to evaluate the differences of the success rate that users achieved with the existing interface and with the final prototype (evaluation point S1 of Table 4.4). Thereby, Figure 6.36 presents the average of the success rate, considering all user groups (defined in section 5.2.2), all scenarios, described in section 4.4, and the success rate labelling, presented in table 4.5.

The changes applied in the interface have leveraged the values of the success rate. In the existing interface only 36% of the scenarios were completely achieved and in the final prototype the percentage of the scenarios completely achieved was 68%.

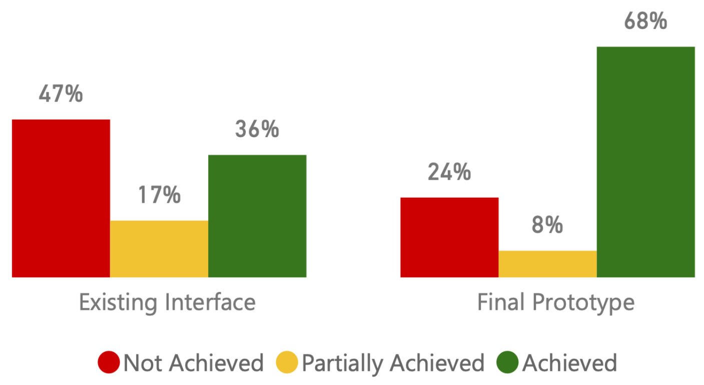


Figure 6.36: **Evaluation point S1:** Effectiveness Comparison between the existing interface and the final prototype. (Comparing the 30 users who tested the existing interface with the other 30 users who tested the final prototype)

Analysis of Variance (ANOVA)

Furthermore, two one-way analysis of variance (ANOVA) were performed to determine if there are statistically significant differences between the mean users' success rate and the mean users' fail rate, using the existing interface and the final prototype.

Achieved Scenarios: Accordingly, the first ANOVA was computed to determine if the users' success rate (based on the percentage of achieved scenarios - stats presented in table 6.13) means are statistically different between the existing interface and the final prototype.

H0 - "There is no significant difference between the mean of achieved scenarios rate, in the existing interface and in the final prototype"

A one-way ANOVA, between subjects, was conducted (table 6.14 to compare if the mean between the samples are equal. In a 95% confidence interval, the null hypothesis is rejected, meaning that the two means are statistically different between the two tests ($p-value = 1.24E-06 < 0,05$). Therefore, it can be concluded that there is a statistically significant difference in terms of the achieved scenarios between the existing interface and the final prototype tests.

Not achieved scenarios: Since the method used to classify the success rate of the users while they performed the usability tests was not only yes or no answers but three types (achieved, partially achieved, and not achieved), the opposite of the achieved analysis is not the not achieved. In that way, another ANOVA analysis was performed concerning the not achieved scenarios in order to measure if users could formulate and comprehend

Table 6.13: Summary of the statistically data regarding the **achieved scenarios** in the existing interface and in the final prototype.

Groups	**Count**	**Sum**	**Average**	**Variance**
Achieved scenarios rate on the existing interface	30	10.4857	0.3495	0.0519
Achieved scenarios rate on the final prototype	30	20.0000	0.6667	0.0511

Table 6.14: Analysis of Variance (ANOVA) of the **scenarios achieved** in the existing interface and in the final prototype.

Source of Variation	**SS**	**df**	**MS**	**F**	**P-value**	**F crit**
Between Groups	1.5087	1	1.5087	29.2874	1.24E-06	4.0069
Within Groups	2.9878	58	0.0515			
Total	4.4965	59				

queries in a most effective way using the final prototype than with the existing interface way even if they made some mistakes. [4]

Therefore, the second ANOVA was used to determine the differences between the not achieved scenarios (stats presented in table 6.15) in the existing interface and the final prototype.

H0 - "There is no significant difference between the mean of not achieved scenarios rate, in the existing interface and in the final prototype"

A one-way ANOVA, between subjects, was conducted (table 6.16 to compare if the mean between the samples are equal. In a 95% confidence interval, the null hypothesis is rejected, meaning that the two means are statistically different between the two tests ($p-value = 5.08E-05 < 0,05$). Therefore, it can be concluded that there is a statistically significant difference in terms of the not achieved scenarios between the existing interface and the final prototype tests.

It was also verified that in the existing interface the average of the achieved scenarios (≈ 0.3495) was lower than the average of the not achieved scenarios (≈ 0.4905), but in the final prototype the average of the achieve scenarios (≈ 0.6667) was superior of the average of the not achieved scenarios (≈ 0.2543). By this means, users managed to most successfully comprehend and formulate queries using the final prototype than using the existing interface.

[4] If the mean of the not achieved scenarios is statistically different and lower, the users could achieved and partially achieved scenarios with a higher probability in the final prototype than using the existing interface.

Table 6.15: Summary of the statistically data regarding the **not achieved scenarios** in the existing interface and in the final prototype.

Groups	**Count**	**Sum**	**Average**	**Variance**
Not Achieved scenarios rate on the existing interface	30	14.7143	0.4905	0.0516
Not Achieved scenarios rate on the final prototype	30	7.6286	0.2543	0.0358

Table 6.16: Analysis of Variance (ANOVA) of the **scenarios not achieved** in the existing interface and in the final prototype.

Source of Variation	**SS**	**df**	**MS**	**F**	**P-value**	**F crit**
Between Groups	0.8837	1	0.8368	19.1610	5.08E-05	4.0069
Within Groups	2.5330	58	0.0437			
Total	3.3697	59				

6.4.2 Efficiency

The time users spent in each scenario when they tested the existing interface and the final prototype was also evaluated in order to compare the usability efficiency attribute of the final prototype against the one of the existing interface (evaluation point S2 of Table 4.4). Figure 6.37 and Table 6.17 presents the statistical data extracted to analyze the distribution of the time elapsed in each scenario and their skewness along the different quartiles.

The interquartile range (IQR) (i.e., the difference between the first and the third quartiles) was used to measure the variability. Moreover, there were values considered as outliers (isolated points in Figure 6.37)) since they were located 1.5(IQR) or more below the first quartile (Q1) or they were located 1.5(IQR) or more above the third quartile (Q3) [14].

As detailed in Table 6.18, the comparison of the results regarding comprehension scenarios shown that the median has decreased at least 34% with the exception of the scenario C4, where the median has increased 20 seconds from the existing interface to the final prototype tests. The difference between the first and the third quartiles has also become shorter in the comprehension scenarios excluding the case of the scenario C3 where this difference has slightly increased (2.67%).

Also noteworthy was the results about the scenario C1. As it was the first scenario tested in the usability tests, the time users needed to perform the intended task reflected also the time needed to explore the system for the first time to understand the way it works. In this scenario, the median has decreased 33.98% and the difference between the first and the third quartiles 52.50%, which means that users required less time to understand the interface. In that way, not only users who explored the interface for the first time were faster in the new prototype, but also the users, who were already accustomed, to use the existing interface were faster reaching the task's goal in the new interface.

Furthermore, the median time spent in the last comprehension scenario performed by users in the usability tests has decreased 42.71% and the difference between the first and the third quartiles 133%, which indicates that after exploring the interface through some scenarios, users needed less time to comprehend the query. Thereby, the interface revealed promising results in terms of productivity, essentially after passing the initial learning phase.

From the edition and formulation point of view, the results revealed that in the first modification scenario the median has decreased 8.76% and the difference between the first and the third quartiles 26.30%. This result demonstrated the success of improving the visibility option to add a new calculated attribute, which made new users to spend less time searching for this option, promoting a decrease on variance.

The formulation scenario did not presented a huge difference in terms of efficiency in opposite of the values presented before, regarding effectiveness. The median has decreased 4.0% and the difference between first and the third quartiles 0.67%. However, as stated before, there were positive results in terms of effectiveness which confirms that the changes applied in the interface, have helped users to reach the formulation goals. Nevertheless, there is a lack of some training and some accelerators to improve this values. Since the usability tests were made without previous training, users have used the final prototype for the first time, which led to a negative impact in terms of efficiency metrics because they needed time to explore the interface.

Even without implementing some accelerators, as shortcuts or search engines, the efficiency results of the final prototype were not worse than the ones of the existing interface. Moreover, in the existing interface, some users have already used the interface before. In opposite, all users who tested the final prototype, tested the interface for the first. In that way, they needed time to explore the interface. Considering these factors, the efficiency metrics remained stable, which is a positive indicator, since users needed the same time using the interface although they were less experienced. Also, these results could be optimized if users gained experience using the interface, mainly if the missing accelerators were further implemented.

The time needed to complete the last modification scenario (M4) has decreased also in the final prototype. The median has decreased 29.35% and the difference between the first and the third quartiles 42.40%.

In a nutshell, this analysis on efficiency optimization pointed out multiple indicators that revealed improvements, even without the presence of some accelerators as well as experience using the interface which could leverage the efficiency metrics to a superior level.

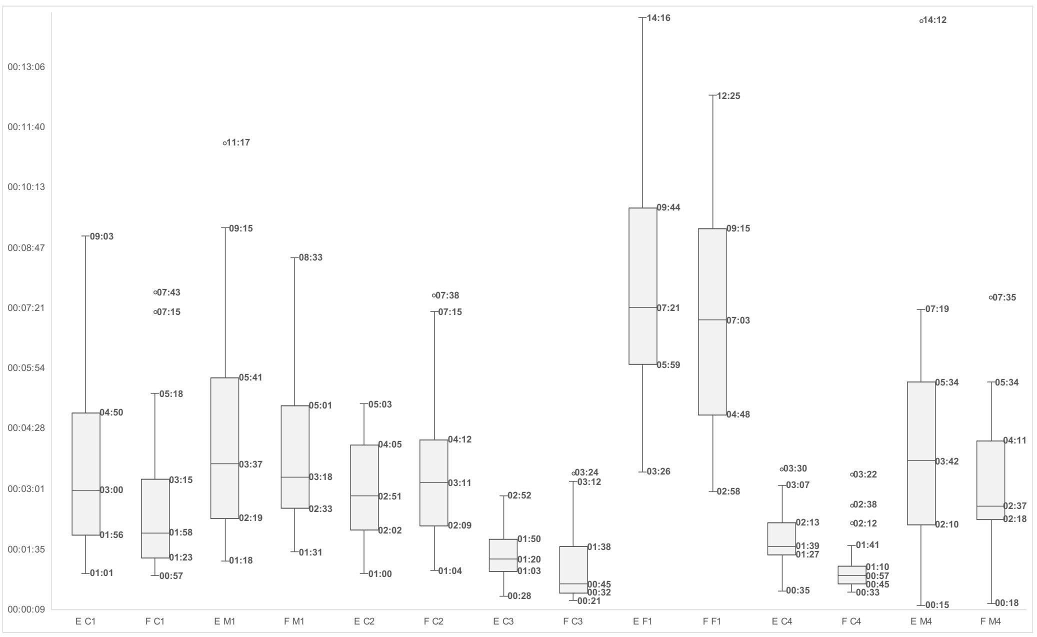


Figure 6.37: Box plots of the time used by users in each scenario in the existing interface (E) and in the final prototype (F).

6.4.3 Satisfaction

After completing all the testings, users filled out a System Usability Survey (SUS) [73], where they attributed a number from 1 to 5 to each one of the sentences presented in Table B.9 and Figure B.1.

The users' satisfaction increased in all questions. The users reacted positively to the changes implemented into the prototype, revealing that the system was more pleasant and simple to use as well as more consistent than the previous interface. Namely, the questions below were the ones with the highest average increase:

- "I thought there was too much inconsistency in this system"(improved 44.58%)
- I found the system very cumbersome to use"(43.88%)
- "I found the various functions in this system were well integrated"(40.0%)
- "I found the system unnecessarily complex"(33.80%)
- "I think that I would like to use this system frequently"(27.35%)

Furthermore, the average SUS score [73] of all participants in the existing prototype was around 58.58% and the score of the final prototype was around 79.08%, which transposed the improvement regarding users' satisfaction.

Table 6.17: Statistical information about the time users elapsed in each scenario in the existing interface and in the final prototype.

Scenario	Interface	Min	Q1	Median	Q3	Max
C1	Existing Interface	0:01:01	0:01:56	0:03:00	0:04:50	0:09:03
	Final Prototype	0:00:57	0:01:23	0:01:59	0:03:15	0:07:43
M1	Existing Interface	0:01:18	0:02:20	0:03:37	0:05:41	0:11:17
	Final Prototype	0:01:31	0:02:33	0:03:18	0:05:01	0:08:33
C2	Existing Interface	0:01:00	0:02:02	0:02:51	0:04:05	0:05:03
	Final Prototype	0:01:04	0:02:09	0:03:11	0:04:12	0:07:38
C3	Existing Interface	0:00:28	0:01:03	0:01:21	0:01:50	0:02:52
	Final Prototype	0:00:21	0:00:32	0:00:45	0:01:39	0:03:24
F1	Existing Interface	0:03:26	0:05:59	0:07:21	0:09:44	0:14:16
	Final Prototype	0:02:58	0:04:48	0:07:03	0:09:15	0:12:25
C4	Existing Interface	0:00:35	0:01:27	0:01:40	0:02:13	0:03:31
	Final Prototype	0:00:33	0:00:45	0:00:57	0:01:10	0:03:22
M4	Existing Interface	0:00:15	0:02:10	0:03:42	0:05:34	0:14:12
	Final Prototype	0:00:18	0:02:18	0:02:37	0:04:11	0:07:35

Table 6.18: Comparison of the time elapsed in each scenario between the tests of the existing interface and the tests of the final prototype.

Scenario	Median Difference	Percentage	Q3-Q1 Difference	Percentage
C1	-0:01:01	-33.98%	-0:01:32	-52.50%
M1	-0:00:19	-8.76%	-0:00:53	-26.30%
C2	0:00:20	11.40%	0:00:03	2.65%
C3	-0:00:36	-44.10%	-0:00:04	-8.65%
F1	-0:00:18	-4.08%	-0:00:02	-0.67%
C4	-0:00:43	-42.71%	-0:01:01	-133.33%
M4	-0:01:05	-29.35%	-0:01:27	-42.40%

However, although the questions regarding learnability have improved in comparison to the existing interface, they do not show as relevant as the remaining questions.

Since the opinion about how the system is easy to use could be highly affected by the users' background, the information retrieved from the users' profile survey was analyzed in order to perceive if there is some degree of correlation between the technical knowledge and the satisfaction using the interface.

Therefore, a study was performed in order to evaluate if the relational databases knowledge could have a significant impact into the users' opinion about how visual query builder is easy to use.

Table 6.19: Kendall's Tau-a significance test.

Kendall's Tau-a	**Existing Interface**	**Final Prototype**
α	0.05	0.05
Sample size	30	30
Combinatorial	435	435
Number of discordant pairs	104	52
Number of concordant pairs	331	383
Tau (τ)	0.5218	0.7609
Tau-Critical	0.218	0.218
Significant? (Tau >Tau-Critical)	**Yes**	**Yes**

In that way, a Kendall's Tau-a correlation test was performed to determine the correlation between the relational database knowledge and the perception about how easy is to use the system. The sample considered the 30 users who tested the existing interface and the 30 users who tested the final prototype.

H0 = "There is no correlation between the relational databases knowledge and the interface ease of use"

Table 6.19 presents the values necessary to calculate this correlation value.

In a 95% confidence interval, the null hypothesis was rejected for both cases, the existing interface and the final prototype, which means that users with more relational databases knowledge were more likely to find the system easy to use.

In conclusion, all the analysis and comparisons made between the existing interface and the final prototype built in this dissertation shown that, there was a considering improvement regarding the task completion rate and in the time users needed to comprehend the queries, illustrated in the interface. However, there are an opportunities to improve in terms of efficiency, mainly for formulation tasks and also regarding learnability, since the relational databases foundations are important to find the system easy to use.

Chapter 7

Conclusions and Future Work

This chapter presents a wide perspective on all of the points approached in this dissertation, as well as, the results achieved, reinforcing also what could be improved in the future.

7.1 Conclusion

The low-code development paradigm has accelerated software development and extended this industry to people with different backgrounds, due to the usage of visual programming languages. Sharing the same vision, VQIs arose in order to facilitate the data querying process, turning it easier to learn, more efficient, and less error-prone.

In this dissertation, the VQI integrated into the OutSystems low-code development platform was studied due to the low level of the users' acceptance and satisfaction on this platform component. Notwithstanding the lack of some SQL features in the visual query building solution, the large set of usability problems identified were the main topics addressed in this dissertation.

In a nutshell, the interface had different usability problems that were preventing users to exploit the main advantages of a visual query language: easiness to learn, acceleration of the querying process, and reduction of the errors which could occur during this process.

Hence, all problems were detailed and categorized as well as the target users of the system who were divided into groups, according to their requirements and expectations. By this means, the solution could be built, by having users in mind (following a user-centered design approach), in order to have a positive effect on all users. Moreover, it was prepared a design and evaluation method that would make it possible to build the most appropriate solution to cover the problems identified and to improve the experience of the target users.

Firstly, a paper prototype was built to explore the first ideas and test with users, in order to validate if the design choices made were forwarding the solution in the right direction to solve the problems that users felt the most in the existing interface.

Secondly, the information obtained from the evaluation of the paper prototype was used to refine details and implement a new prototype, integrated into the OutSystems Platform, with the aim to mitigate some usability problems in the interface and reducing the errors that users mainly made while they were building their queries.

The existing interface and the final prototype were both tested with 30 users, using the same method and testing scenarios in order to compare the success of the changes in design, applied in the interface. With the changes applied to the new interface, users have increased around 84% the task completion rate, which means the new interface has significant improvements regarding effectiveness.

Furthermore, users spent less time, on average, to complete each one of the proposed testing scenarios. Also, they have reported through a System Usability Scale [73] that the system was more consistent and more pleasant to use.

Regarding the research questions pointed out in the beginning of this dissertations, it was concluded that OutSystems developers can easily do complex database queries without using SQL in the final prototype, taking into account the effectiveness results achieved, excluding the complex queries which contain operations not supported by the existing interface. Nevertheless, the formulation of other complex database queries which can include several entities, joins, attributes, and conditions revealed more easily using the final prototype built in this dissertation.

On the other hand, it was concluded that the existing interface had usability problems for less experienced users and the UX of the system could be improved without reducing the system's learnability and satisfaction for new users given that the results of the final prototype have revealed more positive results for less experienced users, regarding these aspects, against the existing interface.

The main goal stated for this dissertation was the mitigation of the current problems in the visual query interface, which has been blocking users to completely take advantage of the VQIs value proposition, leading to a low users' acceptance of this visual query builder.

Since the potential of these interfaces establishes a reduced previous knowledge, the possibility to accelerate the query building process, and the reduction of errors users could make, due to the lack of technical knowledge or because they do not remember important details in some queries, the results showed there was a significant improvement in all these three components.

7.2 Future Work

Although the prototype build has leveraged the potential of the visual query interface in all dimensions (i.e., learnability, effectiveness, and efficiency), some aspects should be further improved in order to optimize even more the UX success metrics of this interface:

1. Learnability: despite the evident improvement noticed, regarding learnability in the new interface, since new users had less difficulties to find how to apply the query operations they needed. However, there is still a correlation between the relational databases knowledge and how users prospect the easiness of use of the system. The join operations used in the visual interface, to combine columns from one or more tables, are the same used in relational databases, which come from relational algebra. Accordingly, the users who have never used a relational database have more difficulty to understand how these merging functions work. In that way, this dependency should be further mitigated trying a new design approach that could be intuitive for users without these relational algebra foundations, maintaining the productivity level provided for technical users. Moreover, users still do not understand the joins with more difficult conditions, thus the join condition should be improved, to not only show a textual expression, but resorting on more visual elements that could turn the readability simpler and accessible for everyone, independently of the users' background;

2. Efficiency/Productivity: the new designed interface has improved the efficiency of use due to the new rearrangement of the different areas and components in the interface. However, as mentioned before, the accelerators and other productivity features, as search engines and keyboard shortcuts, were not implemented as the tests performed could not test this type of features, since users had no previous training or adaptation period before testing the interface. Regardless, it was proven that the most restructuring changes made have increased the efficiency levels even without these boosters. In that way, if these accelerators were also implemented, there was a significant prospection to increase the productivity metrics;

3. Efficiency comparison with SQL: the time users need to formulate queries is a relevant topic. If a user is faster formulating a query in SQL than using the visual query interface, he might prefer to use SQL even if the visual query builder helps him making fewer mistakes. Although the efficiency of the interface built is superior to the one of the previous query builder, a different case study should be further approached in order to evaluate if users with similar database knowledge are faster using SQL or the visual query interface. In order to make this comparison, as believable as possible, the accelerators mentioned should be implemented and users should test the interface after a learning and adaptation period, and not as users who tried the interface for the first time during the test, as occurred in the usability tests performed in this dissertation.

The idea and prototype proposed in this dissertation triggered the start of a new project at OutSystems that aims to explore the solution presented and explore how to improve also, the aspects mentions that should be further improved. The goal is to clearly define a roadmap to invest in the development of these improvements and release of a new visual query builder to the low-code platform.

Bibliography

[1] A. Ahadi, J. Prior, V. Behbood, and R. Lister. "Students' Semantic Mistakes in Writing Seven Different Types of SQL Queries." In: *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*. ITiCSE '16. Arequipa, Peru: Association for Computing Machinery, 2016, 272–277. isbn: 9781450342315. doi: 10.1145/2899415.2899464. url: https://doi.org/10.1145/2899415.2899464.

[2] A. Alshamrani and A. Bahattab. "A comparison between three SDLC models waterfall model, spiral model, and Incremental/Iterative model." In: *International Journal of Computer Science Issues (IJCSI)* 12.1 (2015), p. 106.

[3] T. CATARCI, M. F. COSTABILE, S. LEVIALDI, and C. BATINI. "Visual Query Systems for Databases." In: *J. Vis. Lang. Comput.* 8.2 (Apr. 1997), 215–260. issn: 1045-926X. doi: 10.1006/jvlc.1997.0037. url: https://doi.org/10.1006/jvlc.1997.0037.

[4] T. Catarci and G. Santucci. "Diagrammatic Vs Textual Query Languages: A Comparative Experiment." In: *Visual Database Systems 3: Visual information management*. Ed. by S. Spaccapietra and R. Jain. Boston, MA: Springer US, 1995, pp. 69–83. isbn: 978-0-387-34905-3. doi: 10.1007/978-0-387-34905-3_5. url: https://doi.org/10.1007/978-0-387-34905-3_5.

[5] D. D. Chamberlin and R. F. Boyce. "SEQUEL: A Structured English Query Language." In: *Proceedings of the 1974 ACM SIGFIDET (Now SIGMOD) Workshop on Data Description, Access and Control*. SIGFIDET '74. Ann Arbor, Michigan: Association for Computing Machinery, 1974, 249–264. isbn: 9781450374156. doi: 10.1145/800296.811515. url: https://doi.org/10.1145/800296.811515.

[6] H. Chan, H. Teo, and X. Zeng. "An evaluation of novice end-user computing performance: Data modeling, query writing, and comprehension." In: *Journal of the American Society for Information Science and Technology* 56.8 (2005), pp. 843–853. doi: 10.1002/asi.20178. eprint: https://asistdl.onlinelibrary.wiley.com/doi/pdf/10.1002/asi.20178. url: https://asistdl.onlinelibrary.wiley.com/doi/abs/10.1002/asi.20178.

[7] H. Desurvire, J. Kondziela, and M. E. Atwood. "What is Gained and Lost When Using Methods Other than Empirical Testing." In: *Posters and Short Talks of the 1992 SIGCHI Conference on Human Factors in Computing Systems*. CHI '92. Monterey, California: Association for Computing Machinery, 1992, 125–126. ISBN: 9781450378048. DOI: 10.1145/1125021.1125115. URL: https://doi.org/10.1145/1125021.1125115.

[8] A. Dillon and C. Watson. *User analysis in HCI: the historical lesson from individual differences research*. 1996. URL: http://hdl.handle.net/10150/105824.

[9] A. Dix, A. J. Dix, J. Finlay, G. D. Abowd, and R. Beale. *Human-computer interaction*. Pearson Education, 2003. ISBN: 978-0-13-046109-4.

[10] J. Gehrke and R. Ramakrishnan. *Database management systems*. McGraw-Hill, 2003.

[11] R. Geng, M. Chen, and J. Tian. "In-process Usability Problem Classification, Analysis and Improvement." In: *2014 14th International Conference on Quality Software*. 2014, pp. 240–245.

[12] H. Henriques, H. Lourenço, V. Amaral, and M. Goulão. "Improving the Developer Experience with a Low-Code Process Modelling Language." In: *Proceedings of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*. MODELS '18. Copenhagen, Denmark: Association for Computing Machinery, 2018, 200–210. ISBN: 9781450349499. DOI: 10.1145/3239372.3239387. URL: https://doi.org/10.1145/3239372.3239387.

[13] P. Jennifer, R. Yvonne, and S. Helen. "Interaction design: beyond human-computer interaction." In: *NY: Wiley* (2002).

[14] D. Larose. *Discovering Knowledge in Data: An Introduction to Data Mining*. Wiley, 2005. ISBN: 9780471687535. URL: https://books.google.pt/books?id=JbPMdPWQIOwC.

[15] H. Lu, H. C. Chan, and K. K. Wei. "A Survey on Usage of SQL." In: *SIGMOD Rec.* 22.4 (Dec. 1993), 60–65. ISSN: 0163-5808. DOI: 10.1145/166635.166656. URL: https://doi.org/10.1145/166635.166656.

[16] J. V. M.A. "I. On the diagrammatic and mechanical representation of propositions and reasonings." In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 10.59 (1880), pp. 1–18. DOI: 10.1080/14786448008626877. eprint: https://doi.org/10.1080/14786448008626877. URL: https://doi.org/10.1080/14786448008626877.

[17] J. Nielsen. "Iterative user-interface design." In: *Computer* 26.11 (1993), pp. 32–41. ISSN: 1558-0814. DOI: 10.1109/2.241424.

[18] J. Nielsen. *Usability Engineering*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993. ISBN: 0-12-518406-9.

[19] J. Nielsen and R. Molich. “Heuristic Evaluation of User Interfaces.” In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’90. Seattle, Washington, USA: Association for Computing Machinery, 1990, 249–256. ISBN: 0201509326. DOI: 10.1145/97243.97281. URL: https://doi.org/10.1145/97243.97281.

[20] W. C. Ogden. “IMPLICATIONS OF A COGNITIVE MODEL OF DATABASE QUERY: COMPARISON OF A NATURAL LANGUAGE, FORMAL LANGUAGE AND DIRECT MANIPULATION INTERFACE.” In: *SIGCHI Bull.* 18.2 (Oct. 1986), 51–54. ISSN: 0736-6906. DOI: 10.1145/15683.1044078. URL: https://doi.org/10.1145/15683.1044078.

[21] OutSystems. *OutByNumbers - Benchmark Overview Repor*. Tech. rep. 2013.

[22] P. G. Polson, C. Lewis, J. Rieman, and C. Wharton. “Cognitive walkthroughs: a method for theory-based evaluation of user interfaces.” In: *International Journal of Man-Machine Studies* 36.5 (1992), pp. 741 –773. ISSN: 0020-7373. DOI: https://doi.org/10.1016/0020-7373(92)90039-N. URL: http://www.sciencedirect.com/science/article/pii/002073739290039N.

[23] P. Reisner. “Human Factors Studies of Database Query Languages: A Survey and Assessment.” In: *ACM Comput. Surv.* 13.1 (Mar. 1981), 13–31. ISSN: 0360-0300. DOI: 10.1145/356835.356837. URL: https://doi.org/10.1145/356835.356837.

[24] D. A. Robb, P. L. Bowen, A. F. Borthick, and F. H. Rohde. “Improving New Users’ Query Performance: Deterring Premature Stopping of Query Revision with Information for Forming Ex Ante Expectations.” In: *J. Data and Information Quality* 3.4 (Sept. 2012). ISSN: 1936-1955. DOI: 10.1145/2348828.2348829. URL: https://doi.org/10.1145/2348828.2348829.

[25] K. L. Siau, Hock Chuan Chan, and Kwok Kee Wei. “Effects of query complexity and learning on novice user query performance with conceptual and logical database interfaces.” In: *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 34.2 (2004), pp. 276–281. ISSN: 1558-2426. DOI: 10.1109/TSMCA.2003.820581.

[26] J. B. Smelcer. “User errors in database query composition.” In: *International Journal of Human-Computer Studies* 42.4 (1995), pp. 353 –381. ISSN: 1071-5819. DOI: https://doi.org/10.1006/ijhc.1995.1017. URL: http://www.sciencedirect.com/science/article/pii/S1071581985710178.

[27] C. Stephanidis. “User interfaces for all: New perspectives into human-computer interaction.” In: *User Interfaces for All-Concepts, Methods, and Tools* 1 (2001), pp. 3–17.

[28] T. Taipalus, M. Siponen, and T. Vartiainen. “Errors and Complications in SQL Query Formulation.” In: *ACM Trans. Comput. Educ.* 18.3 (Aug. 2018). DOI: 10.1145/3231712. URL: https://doi.org/10.1145/3231712.

[29] M. Unsöld. "Measuring Learnability in Human-Computer Interaction." Master's thesis. 2018.

Webography

[30] C. Alchin, J. Martin, J. Allenby, and T. Prowse. *2019: Week 9 Solution*. URL: https://preppindata.blogspot.com/2019/04/2019-week-9-solution.html (visited on 02/19/2020).

[31] Balsamiq. *Balsamiq*. URL: https://balsamiq.com/ (visited on 01/29/2020).

[32] Chartio. *Advanced Sorting*. URL: https://chartio.com/help/data-pipeline/advanced-sorting/ (visited on 02/10/2020).

[33] Chartio. *Chartio*. URL: https://chartio.com/ (visited on 02/07/2020).

[34] Chartio. *Chartio Data Explorer | Documentation*. URL: https://chartio.com/docs/data-explorer/ (visited on 02/07/2020).

[35] Chartio. *Chartio FAQs: Joining Data Across Databases*. URL: https://chartio.com/docs/visual-sql/actions/ (visited on 02/10/2020).

[36] Chartio. *Chartio Visual SQL (beta) | Documentation*. URL: https://chartio.com/docs/visual-sql/ (visited on 02/07/2020).

[37] Chartio. *Data Pipeline Steps*. URL: https://chartio.com/docs/data-pipeline/basic/steps/ (visited on 02/10/2020).

[38] Chartio. *Visual SQL Actions | Chartio Documentation*. URL: https://chartio.com/docs/visual-sql/actions/ (visited on 02/10/2020).

[39] Devart. *Building WHERE or HAVING Clause*. URL: https://docs.devart.com/querybuilder-for-sql-server/building-queries-with-query-builder/building-where-and-having-clause.html (visited on 02/10/2020).

[40] Devart. *Grouping Data In Grid*. URL: https://docs.devart.com/querybuilder-for-sql-server/working-with-data-in-data-editor/grouping-data-in-grid.html (visited on 02/10/2020).

[41] Devart. *Making Joins Between Tables*. URL: https://docs.devart.com/querybuilder-for-sql-server/building-queries-with-query-builder/making-joins-between-tables.html (visited on 02/10/2020).

[42] Devart. *Query Builder Tool in dbForge Studio for SQL Server*. URL: https://www.devart.com/dbforge/sql/studio/query-builder.html (visited on 02/07/2020).

[43] Devart. *Sorting Data*. URL: https://docs.devart.com/querybuilder-for-sql-server/working-with-data-in-data-editor/sorting-data-in-grid.html (visited on 02/10/2020).

[44] Devart. *Subqueries in From Clauses*. URL: https://docs.devart.com/querybuilder-for-sql-server/building-queries-with-query-builder/subqueries-other-clauses.html (visited on 02/10/2020).

[45] Devart. *Subqueries Overview*. URL: https://docs.devart.com/querybuilder-for-sql-server/building-queries-with-query-builder/subqueries-overview.html (visited on 02/10/2020).

[46] *Figma: the collaborative interface design tool*. URL: https://www.figma.com/ (visited on 10/25/2020).

[47] Google. *Google Sheets*. URL: https://www.google.com/sheets/about/ (visited on 02/05/2020).

[48] *InVision | Digital product design, workflow and colaboration*. URL: https://www.invisionapp.com/ (visited on 10/21/2020).

[49] ISO. *ISO 9241-11:2018(en) Ergonomics of human-system interaction — Part 11: Usability: Definitions and concepts*. 2018. URL: https://www.iso.org/obp/ui/#iso:std:iso:9241:-11:ed-2:v1:en (visited on 01/27/2020).

[50] JavaScriptor. *js-SQL-parser*. URL: https://github.com/JavaScriptor/js-SQL-parser (visited on 02/10/2020).

[51] Microsoft. *Microsoft Excel*. URL: https://products.office.com/en/excel (visited on 02/05/2020).

[52] Microsoft. *Microsoft Power BI*. URL: https://powerbi.microsoft.com/en-us/ (visited on 02/07/2020).

[53] Microsoft. *Perform common query tasks in Power BI Desktop*. URL: https://docs.microsoft.com/en-us/power-bi/desktop-common-query-tasks#group-rows (visited on 02/10/2020).

[54] Microsoft. *Tutorial: Shape and combine data in Power BI Desktop*. URL: https://docs.microsoft.com/en-us/power-bi/desktop-shape-and-combine-data (visited on 02/07/2020).

[55] Microsoft. *TypeScript - JavaScript that scales*. URL: https://www.typescriptlang.org/ (visited on 02/20/2020).

[56] Mockingbird. *Mockingbird*. URL: https://gomockingbird.com/home (visited on 01/29/2020).

[57] J. Nielsen. *10 Usability Heuristics for User Interface Design*. 1994. URL: https://www.nngroup.com/articles/ten-usability-heuristics/ (visited on 05/05/2020).

[58] J. Nielsen. *Why You Only Need to Test with 5 Users.* 2000. URL: https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/ (visited on 10/14/2020).

[59] J. Nielsen. *How Many Test Users in a Usability Study?* 2012. URL: https://www.nngroup.com/articles/how-many-test-users/ (visited on 10/14/2020).

[60] OutSystems. *Advanced Aggregates Course - Training - OutSystems.* URL: https://www.outsystems.com/learn/courses/132/advanced-aggregates/?LearningPathId=18 (visited on 10/30/2019).

[61] OutSystems. *Aggregates 101 Course - Training - OutSystems.* URL: https://www.outsystems.com/learn/courses/126/aggregates-101/?LearningPathId=18 (visited on 10/28/2019).

[62] OutSystems. *Evaluation Guide (Developing with OutSystems.* URL: https://www.outsystems.com/evaluation-guide/developing-with-outsystems/ (visited on 02/01/2020).

[63] OutSystems. *Low-Code Development Platform for Enterprise Applications.* URL: https://www.outsystems.com/platform/ (visited on 01/31/2020).

[64] OutSystems. *OutSystems Community.* URL: https://www.outsystems.com/community/ (visited on 05/06/2020).

[65] OutSystems. *Service Studio Overview - OutSystems 11 Documentation.* URL: https://success.outsystems.com/Documentation/11/Getting_Started/Service_Studio_Overview (visited on 02/02/2020).

[66] OutSystems. *What's new in OutSystems Hub Edition 2.0.* 2003. URL: https://drive.google.com/file/d/1wkKESumKhJbwK6aoeW2DGU4-TMq-sn0p/view (visited on 02/03/2020).

[67] OutSystems. *What's new in OutSystems Hub Edition 2.2.* 2004. URL: https://drive.google.com/file/d/0B7C37RyL27_oNHF4UmFRX3pFMlpMTVlCWnV5dzJCcmhRS2tj/view (visited on 02/03/2020).

[68] OutSystems. *Agile Platform What's New in Version 5.0.* 2009. URL: https://www.outsystems.com/home/document-download/542/31/0/0 (visited on 02/03/2020).

[69] React. *React - A JavaScript library for building user interfaces.* URL: https://reactjs.org/ (visited on 02/20/2020).

[70] M. Revell. *What Is Low-Code?* 2020. URL: https://www.outsystems.com/blog/what-is-low-code.html (visited on 01/24/2020).

[71] T. Simões. *What's (Not) New in OutSystems: A Product Timeline.* 2018. URL: https://www.outsystems.com/blog/posts/not-new-product-timeline/ (visited on 02/03/2020).

[72] C. Souther. *Low-Code vs. No-Code: What's the Real Difference.* 2019. URL: https://www.outsystems.com/blog/posts/low-code-vs-no-code/ (visited on 01/25/2020).

[73] *System Usability Scale (SUS).* URL: https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html (visited on 10/20/2020).

[74] Tableau. *Add More Data in the Input Step - Tableau.* URL: https://help.tableau.com/current/prep/en-us/prep_add_input_data.htm (visited on 02/10/2020).

[75] Tableau. *Aggregate, Join, or Union Data - Tableau.* URL: https://help.tableau.com/current/prep/en-us/prep_combine.htm (visited on 02/10/2020).

[76] Tableau. *Create a Simple Calculated Field.* URL: https://help.tableau.com/current/pro/desktop/en-us/calculations_calculatedfields_formulas.htm (visited on 02/10/2020).

[77] Tableau. *Filter Your Data - Tableau.* URL: https://help.tableau.com/current/prep/en-us/prep_filter.htm (visited on 02/10/2020).

[78] Tableau. *Sorting Data.* URL: https://help.tableau.com/current/reader/desktop/en-us/reader_sort.htm (visited on 02/10/2020).

[79] Tableau. *Tableau Prep.* URL: https://www.tableau.com/products/prep (visited on 02/07/2020).

[80] Tableau. *What's New in Tableau Prep Builder.* URL: https://help.tableau.com/current/prep/en-us/prep_whatsnew.htm (visited on 02/07/2020).

[81] *Using Images and Assets - Balsamiq for Desktop Documentation | Balsamiq.* URL: https://balsamiq.com/wireframes/desktop/docs/images/ (visited on 10/23/2020).

[82] *Video Conferencing, Web Conferencing, Webinars, Screen Sharing - Zoom.* URL: https://zoom.us/ (visited on 11/12/2020).

Appendix A

Taxonomy of Problems - Existing Interface

The visual interface to formulate queries that was implemented before this dissertation had several usability problems which led users to prefer to use other DQLs such as SQL. In order to comprehend the existing problems of the visual interface or the characteristics that hamper users to extract advantages of that visual approach to query databases, there were performed the following studies:

- **Study and Analysis:** This was the first approach used to explore the existing problems of the interface. The process started with the visualization of two OutSystems tutorials [60, 61] about visual data querying. This was the first experience using the interface and there were pointed out some problems. As this was the first experience, it was mainly found issues regarding hidden operations and behaviors that were not clear for users who did not use the platform. After that tutorials, there has been made some more explorations using practical examples where there were found other problems related to efficiency and effectiveness of use;

- **User Interviews:** There were performed some dialogues with a reduced set of novice and expert users to comprehend what are the main issues pointed out. In the case of expert users, the causes to use SQL instead of the Visual Interface were registered as well as a set of other UX issues. Regarding users who did not have relevant experience either in SQL or OutSystems, it was registered the functionalities more difficult to learn or understand;

- **Community Ideas:** Since OutSystems contains a wide and worldwide Community where users could contribute with suggestions or express their problems or difficulties, all the ideas of the category "Aggregates and Queries"were explored to understand the problems presented. Moreover, the number of likes of each post was registered in order to know what are the problems which had more user reactions;

- **User Testing:** During the user tests of the existing implementation, there were pointed out some issues by users. Also, other problems were found through the interpretation of user-interface interaction.

Furthermore, each issue registered was characterized according to the Nielsen Heuristics [57] and the artifact and task attributes of a Framework adapted from Usability-ODC Framework [11]. Besides, for each issue it was checked if the action most hampered is the query formulation or the query comprehension as well as what are the interface components affected between the ones presented below:

- Actions and Nodes (Method/Function where the Visual Query was added)
- General Layout (The arrangement of the main interface components on the main window)
- Sources
- Joins
- Filters
- Sorting
- Aggregation Functions
- Calculated Attributes
- Static Entities
- Query Result Table
- Test Values

In that way, it is simple to categorize and prioritize the problems detected. Besides, as mentioned above, if the problem were referred in the OutSystems Community, their number of likes and the respecting posts were included. Table A.1 represents the result of all problems detected and categorized under the parameters mentioned:

Table A.1: Taxonomy of Aggregates' Problems - Existing Interface (Last community posts update: May 06, 2020)

Taxonomy of Aggregates' Problems		
ID	**Issue**	**Description**
1	**Entity misadded (automatic joins deletion)**	**When the user adds an entity that trigger automatically a new join, if this entity is removed, the entity could also be removed.**
Figured Out	Community Ideas	
Interface Components Related	Sources and Joins	
Main Action Hampered	Query Formulation	
Usability-ODC Framework Attributes		
Artifact Category	Representation	
Artifact Subcategory	Presentation of Information/Results	
Task Category	Task-facilitation	
Task Subcategory	User Error Tolerance	
Nielsen Heuristics	User Control and Freedom	
Community Posts		
Post		Likes
Changes to behaviour of adding Entities to an Aggregate[1]		25
2	**Select multiple sources at once**	**There are some use cases where this is already possible. However, it should be possible in any use case where the user can add multiple entities.**
Figured Out	Study and Analysis, User Interviews, Community Ideas, and User Testing	
Interface Components Related	Sources	
Main Action Hampered	Query Formulation	
Usability-ODC Framework Attributes		
Artifact Category	Manipulation	
Artifact Subcategory	Direct Manipulation	

Continued on next page

[1]https://www.outsystems.com/ideas/2890/changes-to-behaviour-of-adding-entities-to-an-aggregate

Continuation of Table A.1		
ID	**Issue**	**Description**
Task Category	Task-facilitation	
Task Subcategory	Alternatives	
Nielsen Heuristics	Flexibility and efficiency of use, and Consistency and standards	
Community Posts		
Post		Likes
Changes to behaviour of adding Entities to an Aggregate[2]		25
Allow multiple entity selection on first screen of new aggregate[3]		6
Replace data with multiple entities[4]		1
3	**Add the same entity more than once**	**Allow drag/dropping the same entity more than once (It's possible using Select Source pop-up but it's not possible using Drag and Drop.**
Figured Out	Study and Analysis, and Community Ideas	
Interface Components Related	Sources	
Main Action Hampered	Query Formulation	
Usability-ODC Framework Attributes		
Artifact Category	Manipulation	
Artifact Subcategory	Direct Manipulation	
Task Category	Task-facilitation	
Task Subcategory	Alternatives	
Nielsen Heuristics	Flexibility and efficiency of use, and User control and freedom	
Community Posts		
Post		Likes
Changes to behaviour of adding Entities to an Aggregate[5]		25
Continued on next page		

[2]https://www.outsystems.com/ideas/2890/changes-to-behaviour-of-adding-entities-to-an-aggregate
[3]https://www.outsystems.com/ideas/5417/allow-multiple-entity-selection-on-first-screen-of-new-aggregate
[4]https://www.outsystems.com/ideas/7568/replace-data-with-multiple-entities
[5]https://www.outsystems.com/ideas/2890/changes-to-behaviour-of-adding-entities-to-an-aggregate

Continuation of Table A.1		
ID	**Issue**	**Description**
4	**Add automatically a static entity even if it is not mandatory**	**"It's very annoying that every time I drag to an aggregate an entity with a foreign key to a static entity (SE) - even if it's not mandatory - that static entity is dragged along with it."**
Figured Out	Community Ideas	
Interface Components Related	Sources, and Static Entities	
Main Action Hampered	Query Formulation	
Usability-ODC Framework Attributes		
Artifact Category	Manipulation	
Artifact Subcategory	Direct Manipulation	
Task Category	Task-facilitation	
Task Subcategory	Task/function automation	
Nielsen Heuristics	Consistency and standards	
Community Posts		
Changes to behaviour of adding Entities to an Aggregate[6]		25
5	**Join tables with more than one identifier of the same table (other table)**	**When an entity has several identifiers of same other entity, do the join like currently but with a warning or ask which field to do the join.**
Figured Out	Study and Analysis, Community Ideas, and User Testing	
Interface Components Related	Joins	
Main Action Hampered	Query Formulation	
Usability-ODC Framework Attributes		
Artifact Category	Representation	
Artifact Subcategory	No-message feedback	
Continued on next page		

[6] https://www.outsystems.com/ideas/2890/changes-to-behaviour-of-adding-entities-to-an-aggregate

Continuation of Table A.1		
ID	**Issue**	**Description**
Task Category	Task-facilitation	
Task Subcategory	Keeping the user task on track	
Nielsen Heuristics	User control and freedom, and Visibility of System Status	
Community Posts		
Post		Likes
Changes to behaviour of adding Entities to an Aggregate[7]		25
Join tables with more than one identifier of same other table[8]		5
6	**Move multiple aggregate columns at the same time**	**Actually it is possible to reorder one column. Extend this behavior.**
Figured Out	Community Ideas, and User Testing	
Interface Components Related	Query Result Table	
Main Action Hampered	Query Formulation and Query Comprehension	
Usability-ODC Framework Attributes		
Artifact Category	Manipulation	
Artifact Subcategory	Direct Manipulation	
Task Category	Task-mapping	
Task Subcategory	Functionality	
Nielsen Heuristics	Flexibility and efficiency of use	
Community Posts		
Post		Likes
Move multiple aggregate columns at the same time[9]		5
7	**Search and focus for particular columns of the query result**	**There should be an intuitive solution to navigate through the query result columns. That solution should include search fields and vertical scroll lists to improve the user navigation and control.**

Continued on next page

[7] https://www.outsystems.com/ideas/2890/changes-to-behaviour-of-adding-entities-to-an-aggregate
[8] https://www.outsystems.com/ideas/2885/aggregate-join-tables-with-more-than-one-identifier-of-same-othe
[9] https://www.outsystems.com/ideas/6590/move-multiple-aggregate-columns-at-the-same-time

Continuation of Table A.1		
ID	**Issue**	**Description**
Figured Out	Study Analysis, User Interviews, Community Ideas, and User Testing	
Interface Components Related	Sources and Query Result Table	
Main Action Hampered	Query Formulation and Query Comprehension	
Usability-ODC Framework Attributes		
Artifact Category	Manipulation	
Artifact Subcategory	Direct Manipulation	
Task Category	Task-mapping	
Task Subcategory	Navigation	
Nielsen Heuristics	Flexibility and efficiency of use, and Visibility of System Status	
Community Posts		
Post		Likes
Improve Aggregates to allow entity/attribute filtering[10]		18
Aggregates improvement - Visual Filters[11]		32
Ability to search attributes in aggregates while grouping[12]		5
Search columns in Aggregate[13]		0
Focus Aggregate Column[14]		2
Aggregate Group By Tab[15]		19
8	**Expand hidden columns.**	
Figured Out	Analysis and Study, User Interviews, Community Ideas, and User Testing	
Interface Components Related	Query Result Table	
Main Action Hampered	Query Formulation and Query Comprehension	
Usability-ODC Framework Attributes		
Artifact Category	Manipulation	
Artifact Subcategory	Mouse click	
Task Category	Task-mapping	
Continued on next page		

[10] https://www.outsystems.com/ideas/5720/service-studio-improve-aggregates-to-allow-entity-attribute-filte
[11] https://www.outsystems.com/ideas/5955/aggregates-improvement-visual-filters
[12] https://www.outsystems.com/ideas/5904/ability-to-search-attributes-in-aggregates-while-grouping
[13] https://www.outsystems.com/ideas/6826/search-columns-in-aggregate
[14] https://www.outsystems.com/ideas/6537/focus-aggregate-column
[15] https://www.outsystems.com/ideas/7322/aggregate-group-by-tab

Continuation of Table A.1		
ID	**Issue**	**Description**
Task Subcategory	Functionality	
Nielsen Heuristics	Flexibility and efficiency of use, and Consistency and standards	
Community Posts		
Post		Likes
Expand hidden columns[16]		8
9	**Hide multiple columns with right-click**	**Actually users can use table of query results to hidden some columns. In addition, they can right-click in an attribute column and select hide. However, if they select multiple columns and right-click in one of them, only one will be selected.**
Figured Out	Analysis and Study, Community Ideas, and User Testing	
Interface Components Related	Query Result Table	
Main Action Hampered	Query Comprehension	
Usability-ODC Framework Attributes		
Artifact Category	Manipulation	
Artifact Subcategory	Mouse click	
Task Category	Task-mapping	
Task Subcategory	Interaction	
Nielsen Heuristics	Consistency and standards	
Community Posts		
Post		Likes
Ability to select multiple columns in an aggregate for hiding them[17]		1
10	**If an attribute is lengthy the user cannot see the entry.**	
Figured Out	Community Ideas	

Continued on next page

[16]https://www.outsystems.com/ideas/5054/aggregate-expand-hidden-columns
[17]https://www.outsystems.com/ideas/7102/ability-to-select-multiple-columns-in-an-aggregate-for-hiding-th

Continuation of Table A.1		
ID	**Issue**	**Description**
Interface Components Related	Query Result Table	
Main Action Hampered	Query Comprehension	
Usability-ODC Framework Attributes		
Artifact Category	Representation	
Artifact Subcategory	Presentation of information/results	
Task Category	Task-mapping	
Task Subcategory	Interaction	
Nielsen Heuristics	User control and freedom, and Visibility of System Status	
Community Posts		
Post		Likes
Adjust the columnsize in aggregates query data result[18]		2
Improve cell options for aggregate tables[19]		5
11	**Search and navigate between values**	**Actually users can only select columns. They cannot select rows or cells.**
Figured Out	Community Ideas	
Interface Components Related	Query Result Table	
Main Action Hampered	Query Comprehension	
Usability-ODC Framework Attributes		
Artifact Category	Manipulation	
Artifact Subcategory	Direct Manipulation	
Task Category	Task-mapping	
Task Subcategory	Functionality	
Nielsen Heuristics	Flexibility and efficiency of use	
Community Posts		
Post		Likes
Position cursor on desired field within an aggregate[20]		1
Improve cell options for aggregate tables[21]		5
Continued on next page		

[18]https://www.outsystems.com/ideas/7027/adjust-the-columnsize-in-aggregates-query-data-result
[19]https://www.outsystems.com/ideas/6009/service-studio-improve-cell-options-for-aggregate-tables
[20]https://www.outsystems.com/ideas/7290/position-cursor-on-desired-field-within-an-aggregate
[21]https://www.outsystems.com/ideas/6009/service-studio-improve-cell-options-for-aggregate-tables

Continuation of Table A.1		
ID	**Issue**	**Description**
Copy a value from the Aggregate preview data[22]		27
In an aggregate you can't copy (Ctrl-C) values[23]		3
12	**It's not possible to see all the results (disable remaining results).**	
Figured Out	Community Ideas	
Interface Components Related	Query Result Table	
Main Action Hampered	Query Comprehension	
Usability-ODC Framework Attributes		
Artifact Category	Representation	
Artifact Subcategory	Presentation of information/results	
Task Category	Task-mapping	
Task Subcategory	Functionality	
Nielsen Heuristics	Visibility of System Status, and Match between system and the real world	
Community Posts		
Post		Likes
In aggregates I should have an option to see all results in DEV mode[24]		18
Don't always truncate aggregate results[25]		1
Ability to switch off 'remaining results truncated'[26]		2
13	**Provide more shortcuts.**	
Figured Out	Community Ideas	
Interface Components Related	General Layout, Sources, Joins, Filters, Sorting, Aggregation Functions, Calculated Attributes, Query Result Table, and Test Values	
Main Action Hampered	Query Formulation and Query Comprehension	
Usability-ODC Framework Attributes		
Artifact Category	Manipulation	
Artifact Subcategory	Keyboard press	
Task Category	Task-facilitation	
Task Subcategory	Alternatives	

Continued on next page

[22]https://www.outsystems.com/ideas/2828/copy-a-value-from-the-aggregate-preview-data
[23]https://www.outsystems.com/ideas/7015/in-an-aggregate-you-cant-copy-ctrl-c-values
[24]https://www.outsystems.com/ideas/3052/in-aggregates-i-should-have-an-option-to-see-all-results-in-dev-
[25]https://www.outsystems.com/ideas/8097/dont-always-truncate-aggregate-results
[26]https://www.outsystems.com/ideas/6824/ability-to-switch-off-remaining-results-truncated

Continuation of Table A.1		
ID	**Issue**	**Description**
Nielsen Heuristics	Flexibility and efficiency of use	
Community Posts		
Post		Likes
Aggregate Editor - Shortcut Keys - Move Column Left and Right[27]		5
14	**Present number of results (rows) of the query result.**	
Figured Out	User Interviews and Community Tests	
Interface Components Related	Query Result Tables	
Main Action Hampered	Query Comprehension	
Usability-ODC Framework Attributes		
Artifact Category	Representation	
Artifact Subcategory	Presentation of information/results	
Task Category	Task-mapping	
Task Subcategory	Functionality	
Nielsen Heuristics	Visibility of System Status	
Community Posts		
Post		Likes
Record count in Aggregates[28]		7
Display number of rows returned for an aggregate[29]		2
15	**Invert members of join operation**	**As we only have left join (with or without) it would be interesting one button that changes the entity in the left side of the join with the one that is in the right side.**
Figured Out	Community Ideas and User Testing	
Interface Components Related	Joins	
Main Action Hampered	Query Formulation	
Usability-ODC Framework Attributes		
Artifact Category	Manipulation	
Continued on next page		

[27] https://www.outsystems.com/ideas/3322/aggregate-editor-shortcut-keys-move-column-left-and-right
[28] https://www.outsystems.com/ideas/1780/record-count-in-aggregates
[29] https://www.outsystems.com/ideas/6825/display-number-of-rows-returned-for-an-aggregate

Continuation of Table A.1

ID	Issue	Description
Artifact Subcategory	Mouse click	
Task Category	Task-facilitation	
Task Subcategory	Alternatives	
Nielsen Heuristics	Flexibility and efficiency of use	
Community Posts		
Post		Likes
Aggregate: swap join entities[30]		9
16	**Improve joins readability**	**Change the visual representation of joins, because could be difficult to users to understand what are the tables joined in the query, principally if there is some nesting.**
Figured Out	Community Ideas and User Testing	
Interface Components Related	Joins	
Main Action Hampered	Query Comprehension	
Usability-ODC Framework Attributes		
Artifact Category	Representation	
Artifact Subcategory	Object appearance	
Task Category	Task-mapping	
Task Subcategory	Interaction	
Nielsen Heuristics	Visibility of System Status	
Community Posts		
Post		Likes
Aggregates: make them smarter (nested joins)[31]		14

Continued on next page

[30] https://www.outsystems.com/ideas/1870/aggregate-swap-join-entities
[31] https://www.outsystems.com/ideas/1846/aggregates-make-them-smarter-nested-joins

Continuation of Table A.1		
ID	**Issue**	**Description**
17	**Delete last joins automatically added if a user want to remove last entity added**	**When an entity is added, the system analyzes the relationships with other tables. In some cases the system put automatically other entities related with the added. But if the user removes the entity that he had added, the joins automatically added, remain in the aggregate.**
Figured Out	Community Ideas	
Interface Components Related	Sources and Joins	
Main Action Hampered	Query Formulation	
Usability-ODC Framework Attributes		
Artifact Category	Representation	
Artifact Subcategory	Object appearance	
Task Category	Task-facilitation	
Task Subcategory	Keeping the user task on track	
Nielsen Heuristics	Consistency and standards, and Match between system and the real world	
Community Posts		
Post		Likes
Delete joins automatically when we delete an entity[32]		22
18	**Copy and past filters**	**Could be more efficient to user, to generate a similar filter.**
Figured Out	Community Ideas and User Testing	
Interface Components Related	Filters	
Main Action Hampered	Query Formulation	
Usability-ODC Framework Attributes		
Artifact Category	Manipulation	

Continued on next page

[32]https://www.outsystems.com/ideas/8104/aggregates-delete-joins-automatically-when-we-delete-an-entity

Continuation of Table A.1

ID	Issue	Description
Artifact Subcategory	Keyboard press	
Task Category	Task-facilitation	
Task Subcategory	Alternatives	
Nielsen Heuristics	Flexibility and efficiency of use, and Consistency and standards	
Community Posts		
Post		Likes
Ability to copy and paste Filters in Aggregates (including in bulk)[33]		2
19	**Copy and past joins**	**Could be more efficient to user, to generate a similar join.**
Figured Out	Analysis and Study	
Interface Components Related	Joins	
Main Action Hampered	Query Formulation	
Usability-ODC Framework Attributes		
Artifact Category	Manipulation	
Artifact Subcategory	Keyboard press	
Task Category	Task-facilitation	
Task Subcategory	Alternatives	
Nielsen Heuristics	Flexibility and efficiency of use, and Consistency and standards	
20	**Comments in Filters**	**Improve the highlighting of comments. It doesn't matter if the commentls will be showed separated or if the comments would have a different color. Just examples...**
Figured Out	Community Ideas	
Interface Components Related	Filters	
Main Action Hampered	Query Comprehension	
Usability-ODC Framework Attributes		

Continued on next page

[33] https://www.outsystems.com/ideas/6627/ability-to-copy-and-paste-filters-in-aggregates-including-in-bul

Continuation of Table A.1		
ID	**Issue**	**Description**
Artifact Category	Representation	
Artifact Subcategory	Presentation of information/results	
Task Category	Task-facilitation	
Task Subcategory	Keeping the user task on track	
Nielsen Heuristics	Match between system and the real world	
Community Posts		
Post		Likes
Comments and Activate/Deactivate on Aggregates (Filters, Joins)[34]		13
Comments on aggregate filter[35]		6
Comments inside Aggregates[36]		7
21	**Disable filters**	**Option to disable filter's effect.**
Figured Out	Community Ideas	
Interface Components Related	Filters	
Main Action Hampered	Query Formulation and Query COmprehension	
Usability-ODC Framework Attributes		
Artifact Category	Representation	
Artifact Subcategory	Mouse click	
Task Category	Task-mapping	
Task Subcategory	Functionality	
Nielsen Heuristics	User control and freedom	
Community Posts		
Post		Likes
Comments and Activate/Deactivate on Aggregates (Filters, Joins)[37]		13
Aggregate Filter option[38]		9
Temporarily disable aggregate Filters[39]		2
Allow to disable (not remove) aggregate filters[40]		5

Continued on next page

[34]https://www.outsystems.com/ideas/2058/comments-and-activate-deactivate-on-aggregates-filters-joins
[35]https://www.outsystems.com/ideas/7664/comments-on-aggregate-filter
[36]https://www.outsystems.com/ideas/2891/comments-inside-aggregates
[37]https://www.outsystems.com/ideas/2058/comments-and-activate-deactivate-on-aggregates-filters-joins
[38]https://www.outsystems.com/ideas/1765/aggregate-filter-option
[39]https://www.outsystems.com/ideas/7170/temporarily-disable-aggregate-filters
[40]https://www.outsystems.com/ideas/7565/allow-to-disable-not-remove-aggregate-filters

Continuation of Table A.1		
ID	**Issue**	**Description**
22	**Turn filter edition more accessible**	**Change aggregate filter without openning aggregate.**
Figured Out	Community Ideas	
Interface Components Related	Filters	
Main Action Hampered	Query Formulation	
Usability-ODC Framework Attributes		
Artifact Category	Manipulation	
Artifact Subcategory	Mouse click	
Task Category	Task-facilitation	
Task Subcategory	Alternatives	
Nielsen Heuristics	Flexibility and efficiency of use	
Community Posts		
Post		Likes
Change Aggregate filter without opening Aggregate[41]		23
23	**Change filter without necessity to open the modal of expression editor**	**Turn the insertion and edition process more faster.**
Figured Out	Analysis and Study	
Interface Components Related	Filters	
Main Action Hampered	Query Formulation	
Usability-ODC Framework Attributes		
Artifact Category	Manipulation	
Artifact Subcategory	Direct Manipulation	
Task Category	Task-facilitation	
Task Subcategory	Alternatives	
Nielsen Heuristics	Flexibility and efficiency of use	
24	**Increase the readability of filters (text highlighting) withou open expression editor modal.**	
Figured Out	Analysis and Study	
Interface Components Related	Filters	
Main Action Hampered	Query Comprehension	

Continued on next page

[41]https://www.outsystems.com/ideas/3309/change-aggregate-filter-without-opening-aggregate

Continuation of Table A.1		
ID	**Issue**	**Description**
Usability-ODC Framework Attributes		
Artifact Category	Representation	
Artifact Subcategory	Presentation of information/results	
Task Category	Task-facilitation	
Task Subcategory	Keeping the user task on track	
Nielsen Heuristics	Match between system and the real world	
25	**Remove aggregate filter if rule is empty.**	
Figured Out	Analysis and Study, Community Ideas, and User Testing	
Interface Components Related	Filters	
Main Action Hampered	Query Formulation	
Usability-ODC Framework Attributes		
Artifact Category	Representation	
Artifact Subcategory	Presentation of information/results	
Task Category	Task-facilitation	
Task Subcategory	Task/function automation	
Nielsen Heuristics	Consistency and standards	
Community Posts		
Post		Likes
Remove Aggregate filter if rule is empty[42]		10
26	**Bin button to delete filter is not visible.**	
Figured Out	Community Ideas	
Interface Components Related	Filters	
Main Action Hampered	Query Formulation	
Usability-ODC Framework Attributes		
Artifact Category	Representation	
Artifact Subcategory	Screen layout	
Task Category	Task-mapping	
Task Subcategory	Functionality	
Nielsen Heuristics	User control and freedom, and Recognition rather than recall	
Community Posts		
Post		Likes
Continued on next page		

[42]https://www.outsystems.com/ideas/5056/remove-aggregate-filter-if-rule-is-empty

Continuation of Table A.1

ID	Issue	Description
Don't make me scroll to delete filters[43]		8
Move delete filter button to be near the actual filter in an aggregate[44]		3
27	**Group by attributes' names are not perceptive**	**For example if two different entities have attributes called "Name"and a user group by this two attribuytes, they will be presented as "Name1"and "Name2". Moreover, there is not any reference to attribute's entity.**
Figured Out	Analysis and Study, Community Ideas, and User Testing	
Interface Components Related	Aggregation Functions	
Main Action Hampered	Query Comprehension	
Usability-ODC Framework Attributes		
Artifact Category	Language	
Artifact Subcategory	Nameling/labeling	
Task Category	Task-mapping	
Task Subcategory	Interaction	
Nielsen Heuristics	Visibility of system status, and Error prevention	
Community Posts		
Post		Likes
Set attribute name when 'Group by Id' to EntityNameId[45]		4
28	**View area of group bys.**	**There is no list view with all aggregated attributes, in order to improve the efficiency of query comprehension.**

Continued on next page

[43] https://www.outsystems.com/ideas/5686/dont-make-me-scroll-to-delete-filters
[44] https://www.outsystems.com/ideas/7145/move-delete-filter-button-to-be-near-the-actual-filter-in-an-agg
[45] https://www.outsystems.com/ideas/6456/set-attribute-name-when-group-by-id-to-entitynameid

Continuation of Table A.1		
ID	**Issue**	**Description**
Figured Out	Analysis and Study, Community Ideas, and User Testing	
Interface Components Related	Aggregation Functions	
Main Action Hampered	Query Comprehension	
Usability-ODC Framework Attributes		
Artifact Category	Representation	
Artifact Subcategory	Presentation of information/results	
Task Category	Task-mapping	
Task Subcategory	Navigation	
Nielsen Heuristics	Visibility of system status	
Community Posts		
Post		Likes
Aggregate Group By Tab[46]		19
29	**When the user clicks in "add attribute"(calculated attribute), set the cursor automatically to value**	**This is useful to turns the process of adding faster, reducing the user's effort, increasing the pleasant of use.**
Figured Out	Analysis and Study, and Community Ideas	
Interface Components Related	Calculated Attributes	
Main Action Hampered	Query Formulation	
Usability-ODC Framework Attributes		
Artifact Category	Manipulation	
Artifact Subcategory	Mouse click	
Task Category	Task-facilitation	
Task Subcategory	Keeping the user task on track	
Nielsen Heuristics	Consistency and standards, and Recognition rather than recall	
Community Posts		
Post		Likes
Set cursor to 'value' after adding new attribute to an aggregate[47]		1
Continued on next page		

[46] https://www.outsystems.com/ideas/7322/aggregate-group-by-tab
[47] https://www.outsystems.com/ideas/7528/set-cursor-to-value-after-adding-new-attribute-to-an-aggregate

Continuation of Table A.1		
ID	**Issue**	**Description**
30	**Inconsistent data formats between filters and test values**	**"The test values for Aggregate filters requires YYYY-MM-DD, but data in the results is displayed as MM/DD/YYYY. If you enter MM/DD/YYYY into the test values, you get an error."**
Figured Out	Community Ideas	
Interface Components Related	Filters	
Main Action Hampered	Query Formulation and Query Comprehension	
Usability-ODC Framework Attributes		
Artifact Category	Language	
Artifact Subcategory	Nameling/labeling	
Task Category	Task-mapping	
Task Subcategory	Functionality	
Nielsen Heuristics	Consistency and standards	
Community Posts		
Post		Likes
Allow date entry into test values in same format as results are displayed[48]		4
31	**Inconsistent of sorting between aggregate properties and aggregate popup on action.**	
Figured Out	Community Ideas	
Interface Components Related	Actions and Nodes	
Main Action Hampered	Query Comprehension	
Usability-ODC Framework Attributes		
Artifact Category	Representation	
Artifact Subcategory	Presentation of information/results	
Task Category	Task-facilitation	
Task Subcategory	Alternatives	
Nielsen Heuristics	Consistency and standards, and Flexibility and efficiency of use	
Continued on next page		

[48] https://www.outsystems.com/ideas/4985/allow-date-entry-into-test-values-in-same-format-as-results-are-

Continuation of Table A.1		
ID	**Issue**	**Description**
Community Posts		
Post		Likes
Consistency between Aggregate properties and popup info balloon when hovering[49]		10
32	**Aggregates' Search engine.**	**General query search box.**
Figured Out	Analysis and Study, and Community Ideas	
Interface Components Related	General Layout, Sources, Joins, Filters, Sorting, Aggregation Functions, Calculated Attributes, and Query Result Table	
Main Action Hampered	Query Comprehension	
Usability-ODC Framework Attributes		
Artifact Category	Manipulation	
Artifact Subcategory	Keyboard press	
Task Category	Task-mapping	
Task Subcategory	Functionality	
Nielsen Heuristics	Flexibility and efficiency of use, and Recognition rather than recall	
Community Posts		
Post		Likes
Aggregate - Highlight/Find Usage by Source[50]		4
33	**Freeze formulation areas while scroll horizontal in query result table.**	
Figured Out	Analysis and Study, User Interviews, and Community Ideas	
Interface Components Related	General Layout	
Main Action Hampered	Query Comprehension	
Usability-ODC Framework Attributes		
Artifact Category	Representation	
Artifact Subcategory	Screen layout	
Task Category	Task-mapping	
Task Subcategory	Navigation	
Nielsen Heuristics	Visibility of system status	

Continued on next page

[49] https://www.outsystems.com/ideas/5740/consistency-between-aggregate-properties-and-popup-info-balloon-w

[50] https://www.outsystems.com/ideas/3412/aggregate-highlight-find-usage-by-source

Continuation of Table A.1		
ID	**Issue**	**Description**
Community Posts		
Post		Likes
Freeze Sources, Filters, Sorting and Test Values in AGGREGATE[51]		10
34	**Duplicate tables.**	
Figured Out	Community Ideas	
Interface Components Related	Sources	
Main Action Hampered	Query Formualtion	
Usability-ODC Framework Attributes		
Artifact Category	Manipulation	
Artifact Subcategory	Keyboard press	
Task Category	Task-facilitation	
Task Subcategory	Alternatives	
Nielsen Heuristics	Flexibility and efficiency of use	
Community Posts		
Post		Likes
Aggregates: Duplicate Tables / Joins[52]		1
35	**Duplicate joins.**	
Figured Out	Community Ideas	
Interface Components Related	Joins	
Main Action Hampered	Query Formualtion	
Usability-ODC Framework Attributes		
Artifact Category	Manipulation	
Artifact Subcategory	Keyboard press	
Task Category	Task-facilitation	
Task Subcategory	Alternatives	
Nielsen Heuristics	Flexibility and efficiency of use	
Community Posts		
Post		Likes
Aggregates: Duplicate Tables / Joins[53]		1
36	**Duplicate filters.**	
Figured Out	Analysis and Study	

Continued on next page

[51] https://www.outsystems.com/ideas/5935/freeze-sources-filters-sorting-and-test-values-in-aggregate
[52] https://www.outsystems.com/ideas/7582/aggregates-duplicate-tables-joins
[53] https://www.outsystems.com/ideas/7582/aggregates-duplicate-tables-joins

Continuation of Table A.1		
ID	**Issue**	**Description**
Interface Components Related	Filters	
Main Action Hampered	Query Formualtion	
Usability-ODC Framework Attributes		
Artifact Category	Manipulation	
Artifact Subcategory	Keyboard press	
Task Category	Task-facilitation	
Task Subcategory	Alternatives	
Nielsen Heuristics	Flexibility and efficiency of use	
Aggregates: Duplicate Tables / Joins[54]		1
37	**Distinct function**	**Without extending aggregates' expressiveness is the "Group by of all columns of the result".**
Figured Out	Analysis and Study, and Community Ideas	
Interface Components Related	Query Result Table	
Main Action Hampered	Query Formulation	
Usability-ODC Framework Attributes		
Artifact Category	Language	
Artifact Subcategory	Nameling/labeling	
Task Category	Task-facilitation	
Task Subcategory	Alternatives	
Nielsen Heuristics	Consistency and standards, and Match between system and the real world	
Community Posts		
Post		Likes
Aggregate with Distinct[55]		107
38	**Count Distinct function**	**Expressiveness problem (distinct in all could be "replaced"by group bys, but COUNT DISTINCT cannot be reached using group bys).**

Continued on next page

[54] https://www.outsystems.com/ideas/7582/aggregates-duplicate-tables-joins

[55] https://www.outsystems.com/ideas/Idea_View.aspx?IdeaID=2179&IdeaName=aggregate-with-distinct&utm_source=community&utm_medium=email&utm_campaign=idea-comment

Continuation of Table A.1

ID	**Issue**	**Description**
Figured Out	Community Ideas	
Interface Components Related	Aggregation Function, and Query Result Table	
Main Action Hampered	Query Formulation	
Usability-ODC Framework Attributes		
Artifact Category	Manipulation	
Artifact Subcategory	Mouse click	
Task Category	Task-mapping	
Task Subcategory	Functionality	
Nielsen Heuristics	Flexibility and efficiency of use	
Community Posts		
Post		Likes
Aggregate: Count Distinct[56]		13
39	**Drag and drop input parameters to automatically create filters into Aggregates.**	
Figured Out	Community Ideas	
Interface Components Related	Actions and Nodes, and Filters	
Main Action Hampered	Query Formulation	
Usability-ODC Framework Attributes		
Artifact Category	Representation	
Artifact Subcategory	Object movement	
Task Category	Task-facilitation	
Task Subcategory	Alternatives	
Nielsen Heuristics	Flexibility and efficiency of use	
40	**Button to clear all the test values inserted**	**It could be a good resource if users have many values inserted and they want to clear all. It could be very annoying if they need to clear one at a time.**
Figured Out	Analysis and Study	
Interface Components Related	Test Values	

Continued on next page

[56]https://www.outsystems.com/ideas/1889/aggregate-count-distinct

Continuation of Table A.1		
ID	**Issue**	**Description**
Main Action Hampered	Query Formulation	
Usability-ODC Framework Attributes		
Artifact Category	Representation	
Artifact Subcategory	Mouse click	
Task Category	Task-facilitation	
Task Subcategory	Alternatives	
Nielsen Heuristics	Flexibility and efficiency of use	
41	**Add variable into aggregate puts again the entity already inserted**	**When dragging a variable of type identifier (e.g. CompanyID) the entity Company is added even if it was added before. The only thing that it should add is the filter (if there isn't one already).**
Figured Out	Analysis and Study	
Interface Components Related	Actions and Nodes, and Sources	
Main Action Hampered	Query Formulation	
Usability-ODC Framework Attributes		
Artifact Category	Representation	
Artifact Subcategory	Object movement	
Task Category	Task-facilitation	
Task Subcategory	Keeping the user task on track	
Nielsen Heuristics	User control and freedom	
42	**Query compreension at a glance**	**To comprehend what data the query will retrive (query structure) the user need to switch between tabs.**
Figured Out	Analysis and Study, and User Interviews	
Interface Components Related	General Layout	
Main Action Hampered	Query Comprehension	
Usability-ODC Framework Attributes		

Continued on next page

Continuation of Table A.1		
ID	**Issue**	**Description**
Artifact Category	Representation	
Artifact Subcategory	Screen layout	
Task Category	Task-mapping	
Task Subcategory	Navigation	
Nielsen Heuristics	Visibility of system status	
43	**"Hide others"bug in group by**	**When someone clicks on “hide others” above an aggregation column, there is no visible effect (gray columns stay visible).**
Figured Out	Analysis and Study	
Interface Components Related	Aggregation Functions, and Query Result Table	
Main Action Hampered	Query Comprehension	
Usability-ODC Framework Attributes		
Artifact Category	Manipulation	
Artifact Subcategory	Mouse click	
Task Category	Task-mapping	
Task Subcategory	Functionality	
Nielsen Heuristics	Flexibility and efficiency of use	
44	**Highlight new additions (feedback)**	**Highlight elements added in last interaction. For example sources, joins, group bys sorts, etc...**
Figured Out	Analysis and Study	
Interface Components Related	Sources, Joins, Filters, Sorting, Aggregation Functions, Calculated Attributes, Static Entities, and Query Result Table	
Main Action Hampered	Query Formulation	
Usability-ODC Framework Attributes		
Artifact Category	Representation	
Artifact Subcategory	No-message feedback	
Task Category	Task-facilitation	
Task Subcategory	Keeping the user task on track	
Nielsen Heuristics	Visibility of system status	

Continued on next page

Continuation of Table A.1		
ID	**Issue**	**Description**
45	**Turn visible query formulation options**	**Some query formulation options like aggregation functions or calculated attributes are only visible if the user uses accelerators like right-click on columns. As well as possible, try to show these options without pain simplicity of interface. Improve "Recognition, not recall"without harm "Aesthetic and minimalist design".**
Figured Out	Analysis and Study, and User Testing	
Interface Components Related	General Layout	
Main Action Hampered	Query Formulation and Query Comprehension	
Usability-ODC Framework Attributes		
Artifact Category	Representation	
Artifact Subcategory	Object appearance	
Task Category	Task-facilitation	
Task Subcategory	Alternatives	
Nielsen Heuristics	Recognition rather than recall	
46	**It's not possible to remove more than one aggregated attribute at the same time.**	
Figured Out	Analysis and Study, and User Testing	
Interface Components Related	Aggregation Functions, Calculated Attributes, and Query Result Table	
Main Action Hampered	Query Formulation	
Usability-ODC Framework Attributes		
Artifact Category	Manipulation	
Artifact Subcategory	Mouse click	
Task Category	Task-mapping	
Task Subcategory	Functionality	
Nielsen Heuristics	Flexibility and efficiency of use, and Consistency and standards	
Continued on next page		

Continuation of Table A.1		
ID	**Issue**	**Description**
47	**It's not possible to change aggregation functions**	**To change an aggregation function (e.g. count, max, etc.) is necessary to remove the attribute and to add again.**
Figured Out	Analysis and Study, and User Testing	
Interface Components Related	Aggregation Functions, and Query Result Table	
Main Action Hampered	Query Formulation	
Usability-ODC Framework Attributes		
Artifact Category	Manipulation	
Artifact Subcategory	Mouse click	
Task Category	Task-mapping	
Task Subcategory	Functionality	
Nielsen Heuristics	Flexibility and efficiency of use	
48	**Add another join of two tables that are already merged.**	**When for example already exists a join between A and B with foreign key Key1 and we need to add another join between A and B but with the foreign key Key2, the interface should add automatically the entities twice (e.g., A2 join B using Key2).**
Figured Out	User Testing	
Interface Components Related	Sources and Joins	
Main Action Hampered	Query Formulation	
Usability-ODC Framework Attributes		
Artifact Category	Manipulation	
Artifact Subcategory	Mouse click	
Task Category	Task-facilitation	
Task Subcategory	Task/function automation	
Nielsen Heuristics	Error prevention, and Match between system and the real world	

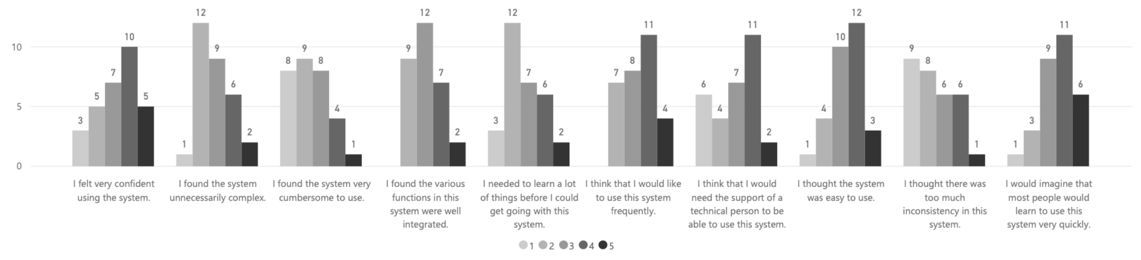
Continued on next page

Continuation of Table A.1		
ID	**Issue**	**Description**
49	**The users don't know that exist hidden attributes**	**If the users don't have experience using the platform they don't know that some attributes are hidden.**
Figured Out	User Testing	
Interface Components Related	Query Result Table	
Main Action Hampered	Query Formulation and Query Comprehension	
Usability-ODC Framework Attributes		
Artifact Category	Representation	
Artifact Subcategory	Object appearance	
Task Category	Task-facilitation	
Task Subcategory	User error tolerance	
Nielsen Heuristics	Visibility of system status, and Match between system and the real world	
50	**Function symbol of aggregated attributes isn't clickable.**	
Figured Out	User Testing	
Interface Components Related	Aggregation Functions, and Query Result Table	
Main Action Hampered	Query Comprehension	
Usability-ODC Framework Attributes		
Artifact Category	Manipulation	
Artifact Subcategory	Mouse click	
Task Category	Task-facilitation	
Task Subcategory	Keeping the user task on track	
Nielsen Heuristics	Consistency and standards, and Visibility of system status	
51	**No visual identifier to show that is a filter edition option**	**Reported by a user in an User Test.**
Figured Out	User Testing	
Interface Components Related	Filters	
Continued on next page		

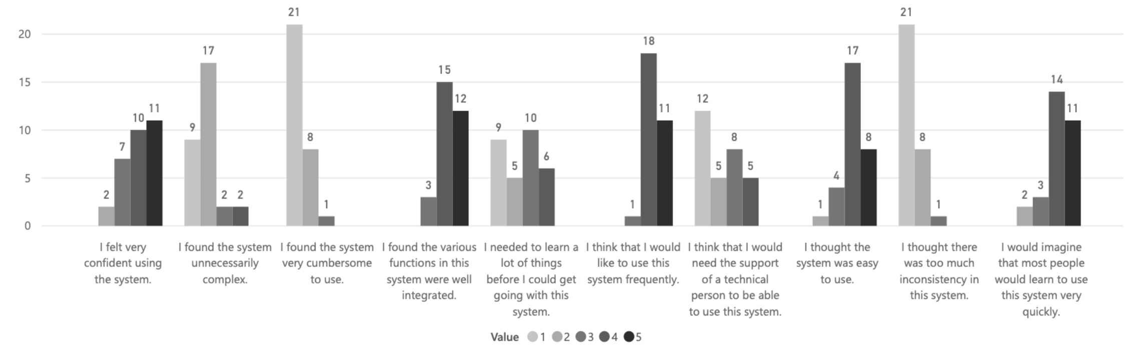
Continuation of Table A.1		
ID	**Issue**	**Description**
Main Action Hampered	Query Formulation	
Usability-ODC Framework Attributes		
Artifact Category	Representation	
Artifact Subcategory	Presentation of information/results	
Task Category	Task-mapping	
Task Subcategory	Functionality	
Nielsen Heuristics	Visibility of system status	
End of Table A.1		

Appendix B

Usability Tests Results



(a) Existing Interface



(b) Final Prototype

Figure B.1: Comparison of the System Usability Scale results (SUS) between the Existing Interface and the Final Prototype.

Table B.1: Users' profile survey results. (Existing interface usability tests - 30 users)

Questions	Yes	No			
Have you a degree in Computer Science or similar?	66.67%	33.33%			
Are you familiar with relational operators (joins)?	80.00%	20.00%			
	Never use	**Almost never**	**Occasionally sometimes**	**Almost every time**	**Frequently use**
Have you already used OutSystems?	30.00%	26.67%	13.33%	3.33%	26.67%
Have you used a query language (SQL or other)?	6.67%	20.00%	46.67%	13.33%	13.33%
Have you already used relational databases?	10.00%	13.33%	30.00%	23.33%	23.33%
How often do you use spreadsheet applications?	0.00%	16.67%	43.33%	10.00%	30.00%
How often do you use business intelligence software?	33.33%	33.33%	20.00%	3.33%	10.00%
	No expe-rience	**<= 6 months**	**<= 1 year**	**1-3 years**	**>= 4 years**
How long do you use the platform?	40.00%	13.33%	10.00%	10.00%	26.67%
	Never use	**Some weeks ago**	**Some months ago**	**Last year**	**Some years ago**
When was the last time that you have used SQL to build queries?	16.67%	40.00%	13.33%	13.33%	16.67%
	1	**2**	**3**	**4**	**5**
How do you define your SQL expertise level?	22.22%	7.41%	33.33%	29.63%	7.41%

Table B.2: Users' profile survey results. (Paper prototype usability tests - 15 users)

Questions	**Yes**	**No**			
Have you a degree in Computer Science or similar?	66.67%	33.33%			
Are you familiar with relational operators (joins)?	86.67%	13.33%			
	Never use	**Almost never**	**Occasionally sometimes**	**Almost every time**	**Frequently use**
Have you already used OutSystems?	33.33%	26.67%	20.00%	6.67%	13.33%
Have you used a query language (SQL or other)?	26.67%	6.67%	33.33%	6.67%	26.67%
Have you already used relational databases?	0.00%	20.00%	40.00%	20.00%	20.00%
How often do you use spreadsheet applications?	0.00%	13.33%	46.67%	6.67%	33.33%
How often do you use business intelligence software?	46.67%	33.33%	6.67%	6.67%	6.67%
	No experience	**<= 6 months**	**<= 1 year**	**1-3 years**	**>= 4 years**
How long do you use the platform?	53.33%	13.33%	13.33%	6.67%	13.33%
	Never use	**Some weeks ago**	**Some months ago**	**Last year**	**Some years ago**
When was the last time that you have used SQL to build queries?	26.67%	33.33%	26.67%	13.33%	0.00%
	1	**2**	**3**	**4**	**5**
How do you define your SQL expertise level?	26.67%	0.00%	40.00%	13.33%	20.00%

Table B.3: Users' profile survey results. (Final prototype usability tests - 30 users)

Questions	**Yes**	**No**			
Have you a degree in Computer Science or similar?	76.67%	23.33%			
Are you familiar with relational operators (joins)?	80.00%	20.00%			
	Never use	**Almost never**	**Occasionally sometimes**	**Almost every time**	**Frequently use**
Have you already used OutSystems?	23.33%	30.00%	26.67%	0.00%	20.00%
Have you used a query language (SQL or other)?	16.67%	13.33%	46.67%	10.00%	13.33%
Have you already used relational databases?	6.67%	20.00%	46.67%	3.33%	23.33%
How often do you use spreadsheet applications?	0.00%	16.67%	46.67%	10.00%	26.67%
How often do you use business intelligence software?	30.00%	33.33%	30.00%	0.00%	6.67%
	No experience	**<= 6 months**	**<= 1 year**	**1-3 years**	**>= 4 years**
How long do you use the platform?	43.33%	10.00%	10.00%	16.67%	20.00%
	Never use	**Some weeks ago**	**Some months ago**	**Last year**	**Some years ago**
When was the last time that you have used SQL to build queries?	20.00%	33.33%	13.33%	20.00%	13.33%
	1	**2**	**3**	**4**	**5**
How do you define your SQL expertise level?	7.69%	26.92%	30.77%	26.92%	7.69%

Table B.4: Success Rate of the scenarios by user group when users tested the existing interface (30 users).

Testing Scenario	User Group	Not Achieved	Partially Achieved	Achieved
C1	OutSystems Developer	90.00%	0.00%	10.00%
	Software Developer	80.00%	0.00%	20.00%
	Citizen Developer	100%	0.00%	0.00%
	Total	90.00%	0.00%	10.00%
M1	OutSystems Developer	0.00%	10.00%	90.00%
	Software Developer	10.00%	10.00%	80.00%
	Citizen Developer	30.00%	10.00%	60.00%
	Total	13.33%	10.00%	76.67%
C2	OutSystems Developer	0.00%	90.00%	10.00%
	Software Developer	60.00%	30.00%	10.00%
	Citizen Developer	90.00%	10.00%	0.00%
	Total	50.00%	43.33%	6.67%
C3	OutSystems Developer	0.00%	0.00%	100.00%
	Software Developer	0.00%	20.00%	80.00%
	Citizen Developer	50.00%	10.00%	40.00%
	Total	16.67%	10.00%	73.33%
F1	OutSystems Developer	50.00%	0.00%	50.00%
	Software Developer	70.00%	20.00%	10.00%
	Citizen Developer	80.00%	10.00%	10.00%
	Total	66.67%	10.00%	23.33%
C4	OutSystems Developer	0.00%	40.00%	60.00%
	Software Developer	30.00%	60.00%	10.00%
	Total	15.00%	50.00%	35.00%
M4	OutSystems Developer	50.00%	0.00%	50.00%
	Software Developer	100.00%	0.00%	0.00%
	Total	75.00%	0.00%	25.00%
Total	OutSystems Developer	27.14%	20.00%	52.86%
	Software Developer	50.00%	20.00%	30.00%
	Citizen Developer	70.00%	8.00%	22.00%
	Total	**46.84%**	**16.84%**	**36.32%**

Table B.5: Success Rate of the scenarios by user group when users tested the paper prototype (15 users).

Testing Scenario	User Group	Not Achieved	Partially Achieved	Achieved
C1	OutSystems Developer	60.00%	0.00%	40.00%
	Software Developer	100.00%	0.00%	0.00%
	Citizen Developer	80.00%	0.00%	20.00%
	Total	80.00%	0.00%	20.00%
M1	OutSystems Developer	0.00%	0.00%	83.33%
	Software Developer	0.00%	0.00%	80.00%
	Citizen Developer	0.00%	0.00%	60.00%
	Total	0.00%	0.00%	75.00%
C2	OutSystems Developer	0.00%	20.00%	80.00%
	Software Developer	0.00%	40.00%	60.00%
	Citizen Developer	0.00%	20.00%	80.00%
	Total	0.00%	26.67%	73.33%
C3	OutSystems Developer	0.00%	0.00%	100.00%
	Software Developer	0.00%	0.00%	100.00%
	Citizen Developer	0.00%	0.00%	100.00%
	Total	0.00%	0.00%	100.00%
F1	OutSystems Developer	0.00%	0.00%	100.00%
	Software Developer	0.00%	0.00%	100.00%
	Citizen Developer	60.00%	0.00%	40.00%
	Total	20.00%	0.00%	80.00%
C4	OutSystems Developer	0.00%	0.00%	100.00%
	Software Developer	0.00%	20.00%	80.00%
	Total	0.00%	10.00%	90.00%
M4	OutSystems Developer	20.00%	0.00%	80.00%
	Software Developer	0.00%	0.00%	100.00%
	Total	10.00%	0.00%	90.00%
Total	OutSystems Developer	11.43%	2.86%	85.71%
	Software Developer	14.29%	8.57%	77.14%
	Citizen Developer	28.00%	4.00%	68.00%
	Total	**17.90%**	**5.14%**	**76.95%**

Table B.6: Success Rate of the scenarios by user group when users tested the Final Prototype (30 users).

Testing Scenario	User Group	Not Achieved	Partially Achieved	Achieved
C1	OutSystems Developer	70.00%	0.00%	30.00%
	Software Developer	80.00%	0.00%	20.00%
	Citizen Developer	100.00%	0.00%	0.00%
	Total	83.33%	0.00%	16.67%
M1	OutSystems Developer	0.00%	10.00%	90.00%
	Software Developer	0.00%	0.00%	100.00%
	Citizen Developer	10.00%	0.00%	90.00%
	Total	3.33%	3.33%	93.33%
C2	OutSystems Developer	0.00%	20.00%	80.00%
	Software Developer	20.00%	40.00%	40.00%
	Citizen Developer	20.00%	40.00%	40.00%
	Total	13.33%	33.33%	53.33%
C3	OutSystems Developer	0.00%	0.00%	100.00%
	Software Developer	0.00%	0.00%	100.00%
	Citizen Developer	20.00%	0.00%	80.00%
	Total	6.67%	0.00%	93.33%
F1	OutSystems Developer	10.00%	10.00%	80.00%
	Software Developer	20.00%	0.00%	80.00%
	Citizen Developer	60.00%	0.00%	40.00%
	Total	30.00%	3.33%	66.67%
C4	OutSystems Developer	0.00%	0.00%	100.00%
	Software Developer	0.00%	30.00%	70.00%
	Total	0.00%	15.00%	85.00%
M4	OutSystems Developer	10.00%	0.00%	90.00%
	Software Developer	30.00%	0.00%	70.00%
	Total	20.00%	0.00%	80.00%
Total	OutSystems Developer	12.86%	5.71%	81.43%
	Software Developer	21.43%	10.00%	68.57%
	Citizen Developer	42.00%	8.00%	50.00%
	Total	**25.43%**	**7.90%**	**66.67%**

Table B.7: Average of the time needed for each scenario and user group in the existing interface (extracted from usability tests with 30 users).

Testing Scenario	User Group	Not Achieved	Partially Achieved	Achieved
C1	OutSystems Developer	0:02:14	-	0:03:46
	Software Developer	0:03:30	-	0:04:20
	Citizen Developer	0:04:47	-	-
	Total	0:03:30	-	0:04:03
M1	OutSystems Developer	-	0:01:18	0:02:06
	Software Developer	0:07:53	0:04:28	0:04:55
	Citizen Developer	0:08:25	0:08:26	0:04:11
	Total	0:08:09	0:04:44	0:03:44
C2	OutSystems Developer	-	0:02:31	0:04:09
	Software Developer	0:02:38	0:03:58	0:02:15
	Citizen Developer	0:03:09	0:03:07	-
	Total	0:02:53	0:03:12	0:03:12
C3	OutSystems Developer	-	-	0:01:01
	Software Developer	-	0:00:54	0:01:40
	Citizen Developer	0:01:50	0:02:45	0:01:57
	Total	0:01:50	0:01:50	0:01:33
F1	OutSystems Developer	0:04:46	-	0:06:05
	Software Developer	0:08:44	0:10:56	0:07:17
	Citizen Developer	0:08:46	0:10:57	0:08:10
	Total	0:07:25	0:10:56	0:07:11
C4	OutSystems Developer	-	0:01:23	0:02:09
	Software Developer	0:02:11	0:01:51	0:02:03
	Total	0:02:11	0:01:37	0:02:06
M4	OutSystems Developer	0:02:38	-	0:03:55
	Software Developer	0:04:47	-	-
	Total	0:03:42	-	0:03:55

Table B.8: Average of the time needed for each scenario and user group in the final prototype (extracted from usability tests with 30 users).

Testing Scenario	User Group	Not Achieved	Partially Achieved	Achieved
C1	OutSystems Developer	0:01:54	-	0:02:02
	Software Developer	0:01:50	-	0:02:22
	Citizen Developer	0:04:10	-	-
	Total	0:02:38	-	0:02:12
M1	OutSystems Developer	-	0:02:36	0:02:31
	Software Developer	-	-	0:04:02
	Citizen Developer	0:06:50	-	0:05:00
	Total	0:06:50	0:02:36	0:03:51
C2	OutSystems Developer	-	0:02:22	0:03:04
	Software Developer	0:02:09	0:04:17	0:03:00
	Citizen Developer	0:03:14	0:04:55	0:03:34
	Total	0:02:41	0:03:51	0:03:12
C3	OutSystems Developer	-	-	0:00:47
	Software Developer	-	-	0:01:01
	Citizen Developer	0:02:52	-	0:01:11
	Total	0:02:52	-	0:01:00
F1	OutSystems Developer	0:11:17	0:06:14	0:05:10
	Software Developer	0:06:59	-	0:08:36
	Citizen Developer	0:07:12	-	0:08:59
	Total	0:08:29	0:06:14	0:07:35
C4	OutSystems Developer	-	-	0:00:59
	Software Developer	-	0:01:45	0:01:14
	Total	-	0:01:45	0:01:07
M4	OutSystems Developer	0:00:45	-	0:03:01
	Software Developer	0:01:18	-	0:04:24
	Total	0:01:02	-	0:03:42

Table B.9: Average and Standard Deviation of the System Usability Scale (SUS) results.

	Average		**Standard Deviation**	
Question	**Existing Interface**	**Final Prototype**	**Existing Interface**	**Final Prototype**
I think that I would like to use this system frequently.	3.40	4.33	1.00	0.54
I found the system unnecessarily complex.	2.87	1.90	1.01	0.79
I thought the system was easy to use.	3.40	4.07	0.97	0.73
I think that I would need the support of a technical person to be able to use this system.	2.97	2.20	1.27	1.14
I found the various functions in this system were well integrated.	3.07	4.30	0.91	0.64
I thought there was too much inconsistency in this system.	2.40	1.33	1.22	0.54
I would imagine that most people would learn to use this system very quickly.	3.60	4.13	1.04	0.85
I found the system very cumbersome to use.	2.37	1.33	1.13	0.54
I felt very confident using the system.	3.30	4.00	1.24	0.93
I needed to learn a lot of things before I could get going with this system.	2.73	2.43	1.11	1.12