Facultade de Informática
UNIVERSIDADE DA CORUÑA

TRABALLO FIN DE GRAO
GRAO EN ENXEÑARÍA INFORMÁTICA
MENCIÓN EN SISTEMAS DE INFORMACIÓN

Tool for the semi-automatic generation of software for Digital Libraries

Estudante: María Delfina Ramos Vidal
Dirección: Alejandro Cortiñas Álvarez
Dirección: María de los Ángeles Saavedra Places

A Coruña, setembro de 2020 .

To my family, for giving me a dream and encouraging me in every step of the way.

Acknowledgements

To my mentors, Alejandro and Ángeles, without whom this thesis wouldn't be possible, for believing in me even against the adversities. For they would always meet me with great advice and uplifting words, even in the difficult circumstances we had to face due to the COVID-19.

To my family and those who are far, far away, for trusting me even when I wouldn't trust myself, for their unconditional support during all the up and downs I faced along the last five years.

To my friends and classmates, for turning this time in to the best years of my life, for gifting me laughter when I didn't feel like laughing, and for always meeting me with a coffee and a smile when I most needed a break.

Abstract

The objective of this end-of-degree thesis is to develop a tool that allows generating source code for different applications with features from the domain of Digital Libraries.

In order to achieve this goal, it was decided to perform the development of a Software Product Line (SPL) to implement the variability of Digital Libraries applications. To accomplish that, it was necessary, first of all, to perform an exhaustive analysis of the domain in order to define the requirements of the product and the generation tool, and to determine the variance of the SPL. The project began with the aforementioned analysis, which was employed as a basis to decide the most relevant features to our prototype. Thenceforth, the design, development and testing of a complete and functional application including the selected features. Finally, the corresponding variation was added to the code from the application so, among the SPL workframe, different applications can be generated. The last step was to create the application to manage the app generation tool.

In the development, PostgreSQL was used for the storage of information, as well as Java, Spring and Hibernate for the implementation of the web server, and Vue.js for the web client. In the case of the software product line, spl-js-engine was used as a derivation engine for product generation, and Vue.js for the web interface designated for the generation tool.

The end-of-degree thesis was managed following an iterative and incremental methodology for software development, therefore we split the development process into weekly iterations in each of which a different set of functionalities was carried out.

Resumo

El objetivo de este trabajo de fin de grado es desarrollar una herramienta que permita generar código fuente para diferentes aplicaciones con funcionalidades del dominio de las Bibliotecas Digitales.

Para lograr este objetivo, se decidió realizar el desarrollo de una línea de productos de software (LPS) para implementar la variabilidad de las aplicaciones para Bibliotecas Digitales. Para lograr eso, primero fue necesario realizar un análisis exhaustivo del dominio para definir los requisitos del producto y la herramienta de generación, y para determinar la varianza del SPL. El proyecto comenzó con el análisis mencionado previamente, que sirvió de base para decidir las características más relevantes de nuestro prototipo. A continuación, el diseño, desarrollo y prueba de una aplicación completa y funcional que incluye las características seleccionadas. Finalmente, se agregó la variabilidad correspondiente al código desde la aplicación para que, en el marco de la LPS, se puedan generar diferentes aplicaciones. El último paso fue crear la aplicación para administrar la herramienta de generación de aplicaciones.

En el desarrollo se empleó PostgreSQL para el almacenamiento de información, así como Java, Spring e Hibernate para la implementación del servidor web y Vue.js para el cliente web. En el caso de la línea de productos de software, se utilizó spl-js-engine como motor de derivación para la generación de productos, y Vue.js para la interfaz web designada a la herramienta de generación.

El trabajo de fin de grado se gestionó siguiendo una metodología iterativa e incremental para el desarrollo de software, por lo tanto, dividimos el proceso de desarrollo en iteraciones semanales en cada una de las cuales se llevó a cabo un conjunto diferente de funcionalidades.

Keywords:

- Code generation
- Software product line
- Digital library
- REST service
- Vue.js
- Spring
- Hibernate
- PostgreSQL
- Generation engine

Palabras chave:

- Generación de código
- Líneas de producto software
- Biblioteca digital
- Servicio REST
- Vue.js
- Spring
- Hibernate
- PostgreSQL
- Motor de generación

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

When information technologies collide with the literary scope, the idea of a digital library is born. Although there can be plenty of different scenarios, let them be a digital library, just a catalogue -when there is no digital copy stored in the system- or an institutional repository -an archive for digital copies of the intellectual output of an institution-, among other examples, we will talk about digital libraries as an umbrella term.

In present times, said digital libraries have become a valuable channel to broadcast cultural material since they make it possible for the society to have access to knowledge, content, and research results that establish a unique cultural heritage. Hence, the development of a digital library is not exempt from complexity and requires, in general terms, an intricate and expensive project. Information digitization processes imply high costs themselves. Besides, the features of each library entail that support software must be custom-made, which means an unreasonable cost that can constitute a critical barrier towards the development of the library.

The main objective for this thesis is to present a tool which allows the generation of source code corresponding to web applications of digital libraries following an approach based on Software Product Lines, which represents an ambitious and recent software engineering field in software development. Said application would be built adapted to the peculiarities of the concrete library to be generated, minimizing both the application and the code's complexity. The purpose of this tool is to offer the final user a solution to the aforementioned problems, providing a product equipped with the desired features at the same time as being as manageable as possible, implemented with the lesser amount of resources and avoiding the presence of unnecessary functionalities that make its usage and maintenance more laborious.

1.2 Goals

To achieve the main goal settled for this project, creating a SPL-based generation tool capable of producing web applications to manage and visualize the information of a digital library starting from a set of given functionalities, the following specific objectives must be achieved:

- In-depth **analysis of the work domain** regarding digital libraries, taking into special consideration how the information is modelled in such an specific field.
- **Define the set of functionalities** that will be part of the auto-generated web applications, including typical functionalities for a digital library, such as: management of book editions, works and users, support for digital resources regarding those books, search for and consult information about an specific authority or work, manage organizations related, manage the physical locations where the edition took place.
- **Analyse and model the variability** of the software product line (SPL), selecting the variable features included in the product, and defining mandatory rules and constraints.
- **Develop the baseline** for the SPL, that is, a web application to manage and run a digital library that includes all the features.
- Annotate source code to make it suitable for the generation engine to generate the functional products.
- Provide a **user-friendly generation tool** through a web interface that allows the user to select the features desired for the final product and generates the applications with a straightforward approach.
- Generate the source code to deploy a digital library web application using a software product line derivation engine.
- Deliver **quality products**, which do not contain known error, for which a series of tests, both manual by the application and automatic, must be carried out.
- Offer **safe products**, that may grant access to the data solely to those users that have been authenticated, for which a security system would be required.

Chapter 2

State of the art and technological fundamentals

In this chapter, we focus on existing tools that perform similar functions to the required objectives, but do not provide a complete solution to the problem. This part of the development provided a better understanding of how the project should be implemented. On a first note, the state of art will be established, taking an in-depth look into the work domain and, lastly, it will close presenting the technologies that will be employed during the implementation of the solution.

2.1 State of the art

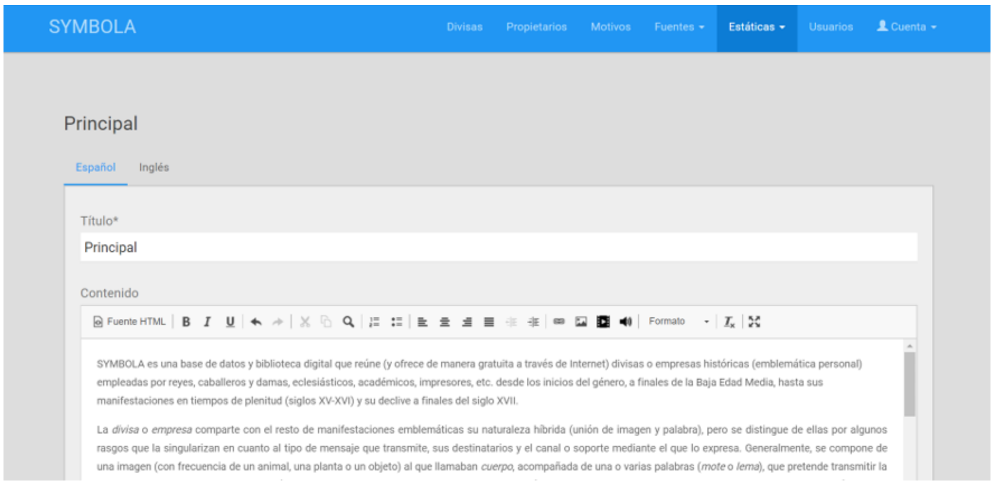
Based on the general ambitions of the project and noting that it is composed of two well-defined elements, a digital library platform and a derivation tool for SPL, an exhaustive research was carried out for each element, analyzing other offers already existing in the industry that could satisfy our necessities. Additionally, this section will summarize the research carried out to grant that the digital libraries generated by the SPL provide the features that an experienced user would expect, both as customer and management, and are supported by a database that enables storing all the relevant data in a digital library.

2.1.1 Digital Libraries

To begin with, platforms have been sought to manage digital libraries, amid which plenty of alternatives have been found. An in-depth analysis of some specific platforms and their most remarkable features was accomplished. An overview of said analysis will be carried out in this section.

Some of the libraries analyzed have the main purpose of giving visibility to different projects in the field of Hispanic languages in order to affect the profitability and recognition of efforts and results, the development of new work instruments, and the opening of new lines of research or the enhancement of existing lines in new registries. The majority of them were constructed based on the OAI-PMH interoperability protocol, whose mission is to promote interoperability standards that facilitate content diffusion on the internet. One of the most outstanding library among them, specially to the team developing this thesis due to their proximity to the mentors is **BIDISO (Biblioteca Digital Siglo de Oro) [1]**, Golden Century Digital Library. Regarding BIDISO, the way they model the management features and the administration interface was the most profitable, with user-friendly approaches, giving the information in a straightforward manner. Another two libraries from this portal worth mentioning are:

- **SYMBOLA Divisas o empresas históricas [2]**, library specialized in historical emblems, from whom stands out its administration side management, including their feature of offering static pages to model additional information about the library. An example of their interface can be seen in figure 2.1



Título*
Principal

Contenido

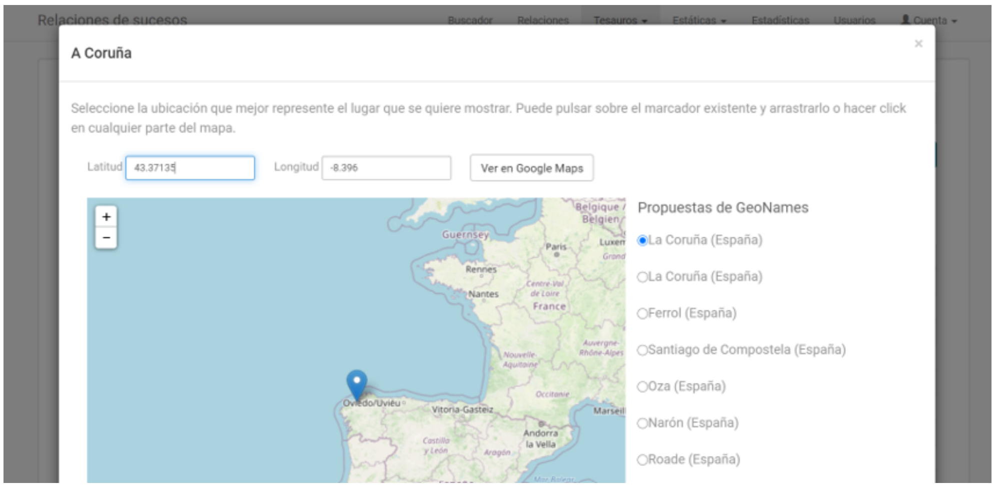
SYMBOLA es una base de datos y biblioteca digital que reúne (y ofrece de manera gratuita a través de Internet) divisas o empresas históricas (emblemática personal) empleadas por reyes, caballeros y damas, eclesiásticos, académicos, impresores, etc. desde los inicios del género, a finales de la Baja Edad Media, hasta sus manifestaciones en tiempos de plenitud (siglos XV-XVI) y su declive a finales del siglo XVII.

La *divisa* o *empresa* comparte con el resto de manifestaciones emblemáticas su naturaleza híbrida (unión de imagen y palabra), pero se distingue de ellas por algunos rasgos que la singularizan en cuanto al tipo de mensaje que transmite, sus destinatarios y el canal o soporte mediante el que lo expresa. Generalmente, se compone de una imagen (con frecuencia de un animal, una planta o un objeto) al que llamaban *cuerpo*, acompañada de una o varias palabras (*mote* o *lema*), que pretende transmitir la

Figure 2.1: Screenshot of Symbola's static pages management

- **CBDRS Catálogo y biblioteca digital de Relaciones de sucesos [3]**, Digital Library of News Pamphlets. Stands out their advanced search feature, enabling an exhaustive and restrictive filtering of the data, in addition of the geographical location management of where an edition took place, as well as an event, and the inclusion of maps for its physically representation, visible in figure 2.2.

The project **Edicion Crítica [4]**, based on the study of the publishing industry in Galicia during the Franquist period, was also taken into account. Protrudes their intricate data model, an ambitious approach that takes into account all different types of literary publications, them



Seleccione la ubicación que mejor represente el lugar que se quiere mostrar. Puede pulsar sobre el marcador existente y arrastrarlo o hacer click en cualquier parte del mapa.

Latitud 43.37135 Longitud -8.396

Propuestas de GeoNames
- [✓ La Coruña (España)]
- La Coruña (España)
- Ferrol (España)
- Santiago de Compostela (España)
- Oza (España)
- Narón (España)
- Roade (España)

Figure 2.2: Screenshot of location management from CBDRS

being editions, works, parts of works, collections of editions, and modelling the different relationships that can happen among them, as well as their authorities, which can sign each work with a different pseudonym, and the organizations in charge of executing the edition of each work.

The project from the University of A Coruña, **Biblioteca virtual galega [5]**, *galician virtual library*, from which we took into special consideration their data model, where they include authors with different pseudonyms, although they don't register which work is signed under each alias, and the relation among works contained in a single edition, with their respective digitized pages to access the data from the original copy of the edition.

To summarize, the remarkable characteristics as a whole for this kind of applications are:

- They tend to be massive platforms that provide an extensive number of functionalities to manage a library and implement a database according to the requirements.
- Collaborative edition, they enable the possibility that many users manage the same data.
- Internal reviewing process, in addition to the collaborative edition, they provide a method to ensure that the data updated to the library is accurate, quality data, by means of internal revision among different team members.
- Geographic Information System (GIS) that enables the location of the different publications in a map.
- The interoperability necessary to share valuable data between organizations, improving the performance of the library.

- Management of different types of literary publications such as editions of a work, collection of editions, parts of works, standalone works. As well as management for their respective authorities, let them be writers, editors, printers or illustrators, to name a few.
- User management, allowing different authentication means, even registration through social networks.
- Support for digital resources that supplement the information available regarding publications and their authors in different formats to match the user needs and the style of the text.
- Complex browser that offer the opportunity to search for an element by different criteria at the same time.
- The navigation on these libraries usually takes profit of the relationships among the entities of the database to find data in a easier way and go from one item to the other without effort.
- User-friendly interface, considering the profile of most users, specialized in literature, the design of these web pages is highly intuitive and transparent.

2.1.2 Software Product Lines

On the other hand, software product line (SPL) derivation tools have been sought, among which we can highlight:

AHEAD Tool Suite (ATS) [6] or **FeatureHOUSE** [7], both tools that offer suppor for Feature Oriented Programming (FOP). FOP is based on the premise of implementing separately the code for each feature and, after that, combines everything with a concrete mechanism depending of the tool. AHEAD is a Java tools collection based on an architecture model that enables the representation of an arbitrary number of software devices as a nested set of equations. FeatureHOUSE is a tool for software artifacts composition independently of language, which allows to combine artifacts written in different programming languages, but this implies programming adapters for each language.

AspectJ [8] is a seamless aspect-oriented extension to the Java programming language. Aspect Oriented Programming (AOP) is a programming paradigm that aims to increase modularity by allowing the separation of cross-cutting concerns. It does so by adding additional behavior to existing code without modification of the code itself. Instead, we declare separately which code is to modify. Its main disadvantaje is that the way of programming gets really compromised to adapt to AOP, which hinders the possibility of escalating the product once it is generated.

Antenna [9] provides a set of Ant tasks suitable for developing wireless Java applications. It is based on conditional compilation, one of the most important and popular techniques to implement variable systems. Using preprocessors, code can be annotated with directives to include or exclude statements depending on feature selections. That way, products can be customized to the needs of a customer. Due to this mark and exclude principle, preprocessors are simple to use and understandable and enable a fine-grained way to implement variability [10]. Preprocessors can be adapted to our case but Antenna only supports Java, it would be impossible to create a variable web client, and its annotations are really complex.

For this purpose, spl-js-engine [11] (which was developed by the directors of this thesis, providing first-hand experience) combines the main advantages of Feature Oriented Programming with the best features of Preprocessors and does so with simple, user-friendly annotations based on JavaScript instead of implementing a new sintax. All the mentioned techonologies are accessible on FeatureIDE, the Eclipse Framework employed to create the feature model, and one of the most well-known tools for SPL.

2.1.3 Related work

This thesis is accomplished with the direction of two professionals that have been an active part of the Database Laboratory from University of A Coruña, which devotes invaluable first-hand experience in matter, not just in concrete projects regarding the creation of digital libraries, besides the development of tools that simplify the generation of complex information systems. With the collaboration of the laboratory, remarkable projects were executed by the University of A Coruña. Some examples can be:

An open source system for the creation, maintenance and exploitation of Digital Libraries [12]: Compression and Indexation, carried by the Databases Laboratory, where they had to main goals: to enhance the behaviour of compression and search algorithms when the text is written in a romance language and, in addition, publishing the algorithms and tools developed as open source.

Advanced tools for the implementation of Digital Libraries [13], also carried by the Databases Laboratory, whose major goals was the development of advanced tools to facilitate the establishment of digital libraries on the web, while improving the quality of their services, from a multidisciplinary point of view. Some features from this project that have quite some relevance are the tool for rapid prototyping digital libraries and the tool to feed digital libraries that permits XML markup of documents.

Furthermore, the European Comission funded a Network of Excellence on Digital Libraries, called DELOS [14], whose main objectives were research and technology transfer, although it was abandoned in 2009. On their last period, DELOS was working on the development of a Digital Library Reference Model, designed to meet the needs of the next-generation systems, and a globally integrated prototype implementation of a Digital Library Management System.

2.2 Technologies employed

During the development of this project several technologies have been used, as listed below:

- **PostgreSQL [15]**: Open source object-relational database management system. Features multi-version concurrency control (MVCC) and the use of read locks.
- **spl-js-engine [11]**: JavaScript tool for the implementation of Software Product Line derivations.
- **Vue.js [16]**: Open-source model–view–view-model progressive JavaScript framework for building user interfaces and single-page applications. One of the advantages is that Vue works with components, elements that encapsulate reusable code. Said components allow the modularized developments and ease the scalability of the project.
- **Maven [17]:** build automation tool used primarily for Java projects, like in this case, where it was employed for the implementation of the digital library.
- **Node.js [18]**: Open-source, cross-platform, JavaScript runtime environment (framework) that executes JavaScript code outside a web browser, focused on speed and scalability.
- **NPM (Node Package Manager) [19]:** package manager for the JavaScript programming language. It is the default package manager for the runtime environment Node.js. Employed to develop the digital library web client and the web interface fo the generation tool.
- **Spring [20]**: Application framework that offers an integral programming and configuration model for Java based applications. Joined with **Spring Boot** [21] to facilitate the configuration of the project, making it automatic.
- **Hibernate [22]**: Object-relational mapping (ORM) tool for the Java programming language. It provides a framework for mapping an object-oriented domain model to a relational database.

Chapter 3

Methodology and project planning

This chapter gives an in-depth description of the development methodology applied to this project, as well as the implemented mechanisms for planning and tracking. Furthermore, the methodology followed when it comes to the development of the Software Product Line is specified.

3.1 Development methodologies

First, to accomplish the goals determined for this system, an iterative and incremental methodology, feature-driven, was elected. The whole project was completed following an agile methodology [23]. These methodologies are characterized by allowing to adapt the way of working according to the conditions of the project and to achieve flexibility and immediacy in the response to adjust the project and its development to the specific circumstances of the environment. In conclusion, they allow a fast adaptability to change and continuous refinement of requirements, as well as good response to error detection.

This type of approach advocates adaptive planning, evolutionary development, early delivery, and continual improvement, and it encourages flexible responses to change [24], providing a series of advantages compared to traditional software engineering. Agile methodologies simplify project planning and estimation, since it is performed for each iteration instead of having to organize the entirety of the project in one sitting, favouring progress tracking by means of meetings held at the end of each iteration. Furthermore, at the end of the iteration a working product is demonstrated to collaborators, allowing their participation in the development process, minimizing overall risk, and granting the product to adapt to changes quickly.

The development methodology applied to this project is inspired by Scrum [25], a framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value. It is described as lightweight, simple to understand but difficult to master. Since this is an individual project, it has been necessary to adapt this work method to the characteristics of the end of degree project, so it has not been possible to apply all the good practices that it proposes.

From here on, Scrum adaptation to this project is outlined, indicating which technique and practices were suitable for this case, considering its concrete characteristics.

The Scrum Framework itself is very simple. It defines only some general guidelines with only a few rules, roles, artifacts and events. Nevertheless each of these components is important, serves a specific purpose and is essential for a successful usage of the framework.

3.1.1 The Scrum roles

One of the first things to understand is how Scrum roles differ from traditional project execution roles. While there are only three main roles in Scrum, they don't automatically align with titles familiar to most of us.

Product owner is a scrum development role for a person who represents the business or user community and is responsible for working with the user group to determine what features will be in the product release. They are responsible for maximizing the value of the product. In addition, they are the sole person responsible for managing the Product Backlog.

A **scrum master** is the facilitator for an agile development team. Scrum is a methodology that allows a team to self-organize and make changes quickly, in accordance with agile principles. The scrum master manages the process for how information is exchanged.They are responsible for promoting and supporting Scrum by helping everyone understand Scrum theory, practices, rules, and values.

Scrum team (aka. Development team) is formed by people who must fulfill all technical needs to deliver the product or the service. They must be self-organized, versatile, and responsible enough to complete all required tasks. It is extremely important that they are self-organized and cross-functional.

In the framework for this thesis, the product owner role was played by the directors of the thesis and the author herself, given that the definition and refinement of the Product Backlog was collectively completed by all the members of the team, as well as the Scrum Master role. On the other hand, the Development Team is solely composed by the author of the thesis.

This adaptation of the methodology implies crucial modifications with respect to the Scrum Theory since the Product Owner role was carried out by several people, as well as the Scrum Master role, while the Development Team was formed by just one member instead of the seven usual people.

3.1.2 Scrum artifacts

The SCRUM artifacts are used to help define the workload coming into the team and currently being worked upon the team. Artifacts defined by Scrum are specifically designed to maximize transparency of key information so that everybody has the same understanding of the artifact. There are many more artifacts, for example, User stories, Release backlog, Burn-up chart etc. But we will concentrate on the core three.

The **product backlog** is a collection of user stories which present functionality which is required/wanted by the product team. It is the single source of requirements for any changes to be made to the product. The Product Backlog evolves as the product and the environment in which it will be used evolves.

The **sprint backlog** is a list of tasks identified by the Scrum team to be completed during the Scrum sprint. During the sprint planning meeting, the team selects some number of product backlog items, usually in the form of user stories, and identifies the tasks necessary to complete each user story.

The **increment** is a sum of all the Product Backlog items completed during a Sprint and the value of the increments of all previous Sprints. At the end of a Sprint, the new Increment must be "Done," which means it must be in usable condition and meet the Scrum Team's definition of "Done".

Along the development, all the Scrum artifacts defined above were applied to the project. The Product Backlog was initially established in the preliminary analysis stage and it was refined progressively in the upcoming meeting for planning and reviewing. During those reunions, the Sprint Backlog was also defined for the iteration that was about to start and the Increment revision was done over the finished sprint.

3.1.3 Scrum events

SCRUM relies on all aspects of the team being and working transparently. With this core ethos in mind, the methodology is structured around a number of key event for ensuring the two other pillars: Inspection and adaptation. Prescribed events are used in Scrum to create regularity and to minimize the need for meetings not defined in Scrum. Other than the Sprint itself, which is a container for all other events, each event in Scrum is a formal opportunity to inspect and adapt something.

- **Sprint**: The heart of Scrum is a Sprint, a time-box during which a "Done", usable, and potentially releasable product Increment is created.

- **Daily Scrum**: time-boxed event held every day of the Sprint where the Development Team plans work for the next 24 hours.

- **Sprint planning**: meeting where the work to be done during the Sprint is organized.
- **Sprint review**: held at the end of the Sprint to inspect the Increment and adapt the Product Backlog if needed.
- **Sprint retrospective**: occurs after the Sprint Review and prior to the next Sprint Planning. Its purpose is to inspect how the last Sprint went, identify, and order the major items that went well and potential improvements and create a plan for implementing improvements to the way the Scrum Team does its work.

For this project, the life cycle was subdivided into several Sprints, each one incorporated the development of a defined set of functionalities. At the end of each Sprint and right before starting the next one, a review and planning meeting was held, equivalent to the Sprint Review and Planning events. During these reunions, the work done during the finished sprint was revised, attending the errors, and defining future changes to be done. Likewise, the Product Backlog refinement was done and the next elements to be implemented were selected.

Since the Development Team was constituted by just one person, it was not conceivable to hold Daily Scrum meetings, nor Retrospective Sprint meetings.

3.2 Methodology support tools

In this section are collected the tools exploited in the project's development, which aided the utilization of the selected methodology.

Gitlab

One of the main tools employed was an instance of Gitlab deployed and maintained by the research group to which the directors of this thesis belong, the Database Laboratory. Gitlab is an open source version repository managed under Git that, moreover, holds multitude of functionalities to ease the integration of agile methodologies such as: an issue tracking system, tasks and milestones management, issue tagging and dashboard for their organization, wiki management, issue-tracking management and collaboration features, among others [26].

Inside Gitlab three projects were created: the first one to manage the source code for the digital library server, the second one for the web client of the library, and the third and last for the web generation tool. All of them were assigned to the same group, which eased the management of milestones and issues.

When developing the project, a workflow based in GitFlow was stablished, meaning that the development was organized in different branches. The main branch, the Master branch, where everything that is pushed to it must work correctly, and consequently the secondary branches created from each issue, the ones where the code will actually be modified to add or modify features to get the desired result.

The common procedure of this method consists in opening the issue, creating a new branch for said issue, make the necessary changes over that branch and when the implemented functionally is finished and works correctly, perform a merge request that consists in merging the code of the branch with the code located in the master branch and proceeding to close the issue. Therefore, the work done for each feature will go into its own branch. This allows to have a clear separation of the code during the development and in case an error occurs during the implementation, it can be easily located and fixed.

Other tools

In addition to Gitlab, during the implementation of the project the following tools were used:

- **Eclipse [27], Visual Studio Code [28]:** both integrated development environments employed for the elaboration of the digital library and the generation tool.
- **Adobe XD [29]:** vector-based user experience design tool for web apps and mobile apps, employed to design the prototypes of the web interface.
- **Draw.io [30] and StarUML [31]:** diagram software for making flowcharts, process diagrams, org charts, UML, ER, and network diagrams to represent the design of the project.
- **Google Chrome and Firefox** browsers employed to run the applications and run manual tests against their functionalities.
- **Overleaf [32]**: Collaborative cloud-based LaTeX editor used for writing and editing research papers.

3.3 SPL development methodology

A software product line is a set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way [33]. SPL are emerging as viable and important development paradigm. They enable rapid market entry and flexible response to market changes; at the same time, they provide a capability for mass customization.

In the generation of the SPL, the spl-js-engine derivation engine was chosen since it was thought to be the best fit for the project. Some of the decisive criteria was that it is a flexible tool and supports any type of files, in addition to the previous experience of the team with said tool. The derivation engine was combined with the scaffolding technique, which consists on generating code from a set of predefined templates and a specification given by the developer.

This tool constructs the products based on the templates, that consist on an annotated source code file, and the specification of the product features [34].

The methodology implemented can be appreciated in the figure 3.1.

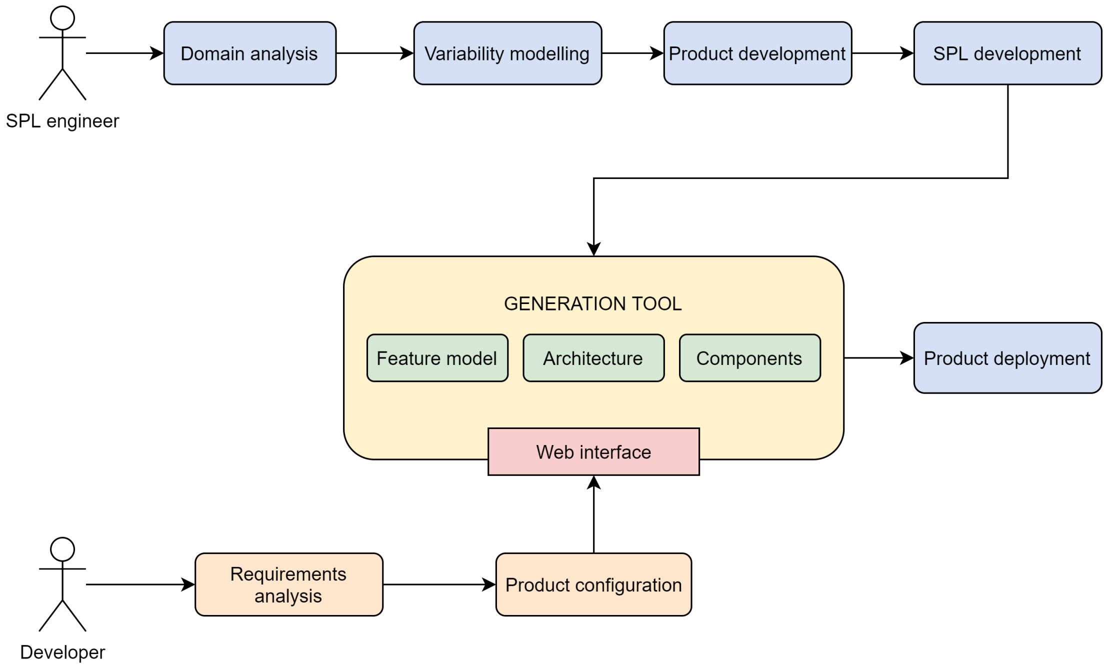


Figure 3.1: SPL development methodology

Firstly, an analysis about the SPL domain was carried out to define the features that would be a part of the resulting products and, based on that, define the requirements to build said product. Furthermore, the generation tool requirements were determined.

Once the features were settled, the variability model was generated, determining common features to all SPLs as well as those that can vary, resulting in a feature model.

After the analysis was completed, product development, the consequent digital library web application. This process was guided by the previous analysis such that organization and implementation of the components pursued the ease of inclusion and exclusion of said components on the generated products.

Eventually the implementation and testing of the digital library took place, and right after the development of the Software Product Line took off, using spl-js-engine tool. During this stage, the source code of the library was annotated, and the web generation tool interface was implemented.

Reached this point of the process, the development of the digital library tool was completed. From here on it can be employed to generate products starting from a product configuration, in other words, a set of selected features.

3.4 Planning and tracking

This section details the resources necessary to carry out the project, as well as the planning and estimation carried out initially and the actual monitoring of the planning, along with deviations in time and cost.

3.4.1 Resources

The resources, both human and technical, necessary to carry out this project.

Human resources

For the realization of this thesis we have had a team of three people formed by the directors and the author of the work, who have assumed the roles of Product Owner, accomplishing the product backlog management; analyst, carrying out planning and design; and as a developer, carrying out the implementation and testing of the different functionalities.

Technical resources

Within the technical resources used in the development of this project can be differentiate two groups: hardware resources and software resources.

Regarding hardware resources, a laptop has been used for the sake of the implementation and testing of all the functionalities defined for the system, as well as the thesis' diagrams and documentation elaboration.

Regarding software resources, those mentioned in section 3.2 are included.

3.4.2 Planning

At the beginning of the project, a preliminary analysis was carried out in order to determine its scope and the technologies to be used in it. The preliminary analysis phase had a duration of four weeks and in it the following tasks were carried out:

- **Technology study:** an analysis of the main JavaScript available frameworks was carried out to determine which was the most appropriate to develop the web interface of the application. After studying the advantages and disadvantages of each one of them, the decision was made to use Vue.js since it is a simple and easy framework to use but at the same time powerful enough for the development of this project. The previous experience of the team working with Vue.js was also crucial at the time of the decision.

- **Definition of system actors:** an analysis of the possible roles and actors that could interact with the system, defining the functionalities to which each of them would have access.

- **Data model definition:** The initial version of the Product Backlog, defining the functionalities of the system.

- **Initial product backlog implementation:** the data model design was performed to determine how the information from the application would be stored.

- **Elaboration of mock-ups:** design of interface prototypes for the web application by using mock-ups, in order to refine functionalities and requirements and how everything must fit in the desired final product.

The software development process split into weekly sprints, resulting in eight iterations, combining with the realization of this document. The work to be done in each sprint was planned so that the author would work 10 hours every day, every day of the week, in the performance of analysis, design, implementation tasks and tests. In addition, it was estimated that the monitoring and planning meeting corresponding to each sprint would last approximately one hour.

This planning results in 70 hours of work in each sprint by the author, resulting in a total of 560 hours estimated for software development. The total sum of hours invested in the preliminary phase of the analysis, 40 hours, must be added to this amount, resulting in a grand total of 600 work hours estimated for the development of this project. Once the time cost was estimated, the monetary cost for the realization of the project was estimated, considering the standard salary per hour for a professional, as stated in table 3.1. Said estimation can be found in the table 3.2.

Role	Salary per hour
Product owner	30€
Analyst	25€
Developer	15€

Table 3.1: Salary per hour for each role

Member	Product owner hours	Analyst hours	Developer hours	Cost
Ángeles Saavedra	10	-	-	300€
Alejandro Cortiñas	10	15	-	675€
Delfina Ramos	20	30	560	9.750€
Total	**40**	**45**	**560**	**10.725**

Table 3.2: Planned project costs

3.4.3 Schedule tracking

This section will analyze the monitoring of the project on the initial planning, detailing the work done in each sprint with respect to its planning and mentioning the deviations in time and cost suffered together with the causes thereof. It should be noted that the number of stories assigned to each sprint is not the same, since not all of them count with the same complexity and the development time required for its implementation is not equivalent.

It must be taken into account that, following the guidelines of the methodology used, in the Review and Planning meeting of each sprint a revision of the user stories implemented in the iteration was performed, and the remediation and implementation of the errors and improvements detected for the next sprint was planned.

Sprint 1

For the first Sprint the assigned tasks were the creation of the web server skeleton and the creation of the web client skeleton using Vue CLI. Also, it was required to create the feature model using FeatureIDE and generate the XML file for the features. Lastly, during this sprint it was decided to implement the basic functionalities to login and logout from the app, register, view an user's profile and delete an account, corresponding to user stories from 1 to 14, 45 and 46.

Sprint 2

For the second Sprint the assigned tasks were those related to the creation of an edition, displaying a list of editions both from the admin management and the public access. User stories 15, 20, 21, 26, 32, 33, 38 and 47.

Sprint 3

For the third sprint the assigned tasks were those related to the creation and management of authorities and the thesaurus for organizations and locations, both on the server side and the client side. From the product backlog the equivalents are the user stories 17, 18, 19, 22, 24, 28, 29, 40, 48 and 49.

Sprint 4

For the fourth sprint the assigned tasks were those related to the creation and adaptation of the generation tool, implementing the skeleton of web client. From the SPL product backlog the equivalents are the user stories 1 to 3. The ideal progress would be to start implementing the server side of file loading in this sprint, since it was never done before by the author and it may present some difficulties.

Sprint 5

For the fifth sprint the assigned tasks the ideal progress would be to start scribbling the memory of the thesis and do parallel implementations among the programming of the application and the drafting of the document since the time lapse available was really small. Simultaneously, the features related to loading data should begin here with the images for the covers and profile pics and the implementation of the entities for the digital resources. Also, from the generation tool the importation and exportation of specification should be implemented, without static pages.

Sprint 6

For the sixth sprint the assigned tasks were those related to the creation and management of works and begin the loading of files. Equivalent user stories: 16, 23, 25, 27, 34, 39. Also, from the generation tool, the generation of static pages should be implemented by the end of the sixth sprint. The user stories from 4 to 6 should be implemented.

Sprint 7

For the seventh sprint the assigned tasks were the search functionalities for the product, user stories 35 to 37 and the correction of pending errors. In this sprint the annotation of the product to model the variability should be started at least.

Sprint 8

For the eighth and last sprint the product prototype should be completed so that the user story number 7 from the SPL product backlog can be implemented and working. By the end of this sprint the memory for the thesis should be done and corrected.

Chapter 4

Software product line analysis

This chapter details the analysis carried out for the development of the product line software. In this process it was necessary to define the requirements of the product and the tool generation, as well as analyze and model the variability of the SPL.

4.1 Product requirements

In this section the actors defined for the product are detailed. Six actors have been determined, whose hierarchy can be seen in the figure 4.1.

4.1.1 Actors

- **User:** abstract actor that represents the functionalities common to all the library users. These functionalities are: access the library and view literary publications' details, view authorities' public profiles, and search for publications by title as per authority, year or location with a simple search, or select an advanced search by multiple fields such as title, authority with differentiated roles (author, editior or other), location, language, a date or a date range, or the ISBN code.

- **Anonymous user:** can register, log into the app and search specific content, such as details for the published editions or works, and the authorities of each of the elements.

- **Social network user:** can access the digital resources by authentication with a social network account.

- **Reporters**: can have complete access to the publications stored in the library once the account is approved by an admin, as well as their authorities and download the digital resources associated to them, export the details available for each element in a convenient format, such as CSV, TXT, JSON, EXCEL or PDF, as well as a collection of elements.

- **Collaborators**: can create new content for the digital library and modify existing content, as well as consult all the information. They can publish the content they create, save it as a draft or mark it as pending of review by an admin.
- **Admin**: can access all the functionalities offered by the application, can create new publications, review contributions by collaborators and publish them, schedule the release of new content in the future, register new collaborators and approve the registration of a new reporter.

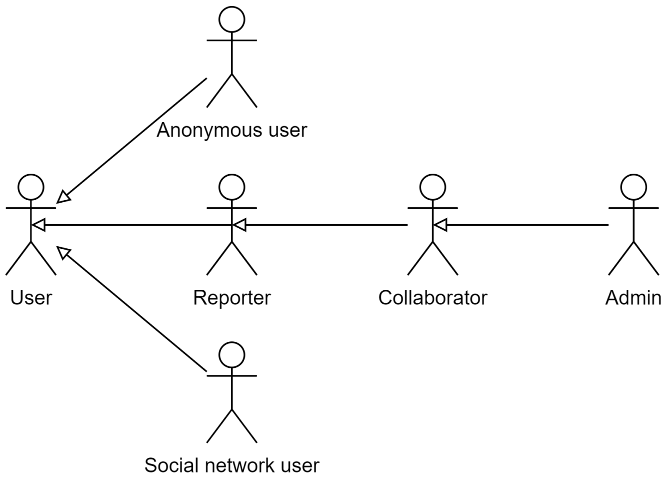


Figure 4.1: Actor diagram

4.1.2 Requirements

This section details the functional and non-functional requirements defined for the product. Initially the requirements are listed in a general way to later detail them in the product backlog in the form of user stories.

Functional requirements

The functional requirements established for the application are as follows:

- **Publication elements management**, creation of a new element with the input from a form, update of said element and deletion if needed.
- Possibility to model **different type of publications** such as editions, works, collections or parts, and the corresponding relations among them taking into account that a collection can contain several editions, and edition can be conformed by several works or parts of works.

- **Reporters management**, registration of a new reporter, approval of said new reporter, authentication, view profile, modify personal information and delete an account.
- **Collaborators management**, creation of a new collaborator account from the administration panel, view profile, modify personal data, delete said account.
- **Elements management**, admin and collaborators can create, save drafts if the form is not complete, mark it as pending for review, admin can review collaborators work, can schedule publications, publish them for everyone to see, update and delete them from the database.
- Edition **location management**, create locations related to each edition, save geographical location with coordinates, save historical references to defined coordinates, modify their data and delete them.
- **Digital resource management**, creation of files associated to each publication that represent a digital copy of the original copy we refer to, update the file and delete said file from our disk.
- **Authority management**, can create an authority in relation to each of the publications, storing the alias under which the work was signed and the role developed in the production of said work. View the data corresponding to the authority, modify the information, delete the authority, in addition to storing the contact information, including the social networks.
- **Organization management**, can create organizations related to the publications, them being the editorials or the library that stores the physical copy, view details from said organization, modify their data, delete the organization from the database, and store information about the different organization directors over the years.
- **Export features**, can export the data into a file of the format desired.
- **Management features**, can view statistics about who uploads more resources or who consults more information.
- **Search features**, can view the result list from the search in map or in list, can have simple search by one term or advanced search by many parameters.
- **Internationalization** of the messages displayed in the application.
- **Creation of static pages** to store additional information about the library or the management team, for example.

- **Management of favorite elements**, the reporters can mark a published item or an authority as a favorite to keep track of new updates and have easy access to them.

Non-functional requirements

The non-functional requirements refer to the characteristics that the program should have to improve the user experience and those defined for the application are as follows:

- **Ease of usage:** the web application must be easy to use, for which it must be developed as an intuitive and simple web interface that facilitates operations.
- **Easy installation/deployment:** the system should be easy to install and configure, so documentation with appropriate instructions must be provided.
- **Security:** the application must ensure that the data is accessible only by authorized users, controlling access to them through some system of security.
- **Compatibility:** the web application must display correctly in Google Chrome and Firefox browsers.

Product backlog

Following the selected methodology, a product stack was developed that constituted the only source of requirements during application development. General requirements previously mentioned were detailed in the form of user stories that, with the advancement development, they were refined until obtaining a sufficient level of detail for their implementation. Likewise, the product stack was reordered based on the priorities of every moment, until the final version shown below.

1. As an anonymous user I want to register a new account as a Reporter and authenticate myself using my login and password, or login as a Collaborator or Admin using the same information.
2. As an anonymous user I want to authenticate into the web using a social network account.
3. As an authenticated user I want to logout from the application.
4. As an Admin I want to approve new Reporter account registered to give them permission to access the digital resources.
5. As an administrator I want to register new collaborators associated to the digital library into the application.

6. As a user I want to be able to access my profile and view my public data.
7. As a user I want to be able to modify my personal data.
8. As an admin I want to be able to view a list displaying all the registered collaborator accounts with their most relevant data.
9. As a user I want to be able to access the profile of a collaborator and visit an external link associated to their account to get more information.
10. As an admin I want to be able to modify the data corresponding to a registered collaborator.
11. As an admin I want to be able to delete an account corresponding to a collaborator.
12. As an admin I want to be able to view a list displaying all the registered reporters accounts with their most relevant data, including registration date, and their status, them being approved to access the digital resources or not.
13. As an admin I want to be able to approve a new reporter to grant them access to the digital resources.
14. As an admin I want to be able to delete a reporter account.
15. As an user I want to be able to navigate the web and see a list with all the editions that I have published.
16. As an user I want to be able to navigate the web and see a list with all the works that I have published.
17. As an user I want to be able to navigate the web and see a list with all the authorities that I have published.
18. As an admin or collaborator I want to be able to see a thesaurus storing all the organizations that I have stored.
19. As an admin or collaborator I want to be able to see a thesaurus storing all the edition locations that I have stored.
20. As a collaborator or administrator I want to navigate the management web and see a list with all the editions stored, including drafts and those pending of review in addition to the published ones.

21. As a collaborator or administrator I want to be able to create a new edition from the input collected from a form. The form must provide an input for the cover of the edition in addition to the basic data.

22. As a collaborator or administrator I want to be able to include the information regarding to the authorities related to said edition without leaving the form, being able to select an author already present in our database or create a new author from scratch, repeating the process as many times as authorities I need.

23. As a collaborator or administrator I want to be able to include the information regarding works, collections, parts or organizations without leaving the form.

24. As a collaborator or administrator I want to be able to create a place of edition only if it is related to an edition, and only one place can be associated to each edition.

25. As a collaborator or administrator I want to include digital resources to consult the content of the edition from a digital copy. Said resources can be PDF files, EPUB files, videos, audio files, digitized pages in image format or external files referenced from a link.

26. As a collaborator or administrator I want to be able to save an incomplete form as a draft and if once the form in filled, if I am a collaborator, I want to mark it as pending for review by an admin. If I am an admin, I want to be able to publish the edition instantly or schedule the release date to an upcoming day.

27. As a collaborator or administrator I want to be able to create a new work with the same features as an edition, taking into account that a work does not have a location associated.

28. As a collaborator or administrator I want to be able to create a new authority, including a list of its known alias and its contact information with a list of their available social network accounts, modify the data of said authority or remove all of its information from the database.

29. As a collaborator or administrator I want to be able to create a new organization.

30. As an user I want to be able to mark an author as a Favorite Author to add it to my list of favorite authors and have easy access to the new updates related to said author.

31. As an user I want to be able to mark a publication as favorite to add it to my list of favorite publications and have easy access any time I want to read it.

32. As an admin or collaborator I want to be able view a management list with the most relevant data from a publication, including their status, which can be a draft, pending for revision, published, or revised but not published.

33. As an admin I want to be able to review the data included in an item that was marked as "pending for revision" and accept it, marking it as revised after selecting if it will be published in the same moment or scheduled.

34. As a user I want to be able to download the digital resources available for each publication in case it is a file or visit the resource in case it is an external file.

35. As a user I want to be able to search for editions and display the result as a simple text list, as a grid with the cover of each edition, or as a map displaying the edition location. I want to be able to perform a simple search by a single field, it being title, author, year or location, or perform an advanced search by multiple fields among the public data available for a edition.

36. As a user I want to be able to search for works and display the result as a simple text list or as a grid with the cover of each work. Also, I want to be able to perform a simple search by a single field, it being title, author or year, or perform an advanced search by multiple fields among the public data available for a work.

37. As a user I want to be able to search for authorities and display the result as a simple text list or as a grid with a profile picture of each authority.

38. As a user I want to be able to navigate the web and visualize a list of editions, as a simple text list or as a grid with an image of each cover. The list can be navigated using an infinite scroll or with a pagination system.

39. As a user I want to be able to navigate the web and visualize a list of works, as a simple text list or as a grid with an image of each cover. The list can be navigated using an infinite scroll or with a pagination system.

40. As a user I want to be able to navigate the web and visualize a list of authorities, as a simple text list or as a grid with an profile picture of each author. The list can be navigated using an infinite scroll or with a pagination system.

41. As a user I want to be able to export the public data available regarding a single publication into a file that can be a PDF, a TXT, a JSON, an EXCEL or a CSV file.

42. As a user I want to be able to export the public data available regarding an single authority into a file that can be a PDF, a TXT, a JSON, an EXCEL or a CSV file.

43. As a user I want to be able to export the bibliographic reference of a single edition into a file that can be a PDF, a TXT, a JSON, an EXCEL or a CSV file.

44. As a user I want to be able to select several items from the public list of publications or authorities to export them into a file.

45. As an administrator or collaborator I want to be able to filter the list of registered collaborators to find an specific collaborator.

46. As an administrator or collaborator I want to be able to filter the list of registered reporters to find an specific reporter.

47. As an administrator or collaborator I want to be able to filter the list of publications (editions, works, collections or parts) to find an specific item.

48. As an administrator or collaborator I want to be able to filter the list of authorities to find an specific authority.

49. As an administrator or collaborator I want to be able to filter the thesaurus for organizations and locations to find the elements matching the criteria.

50. As an administrator or collaborator I want to be able to perform an advanced search over all the elements stored in the database.

51. As an administrator or collaborator I want to be able to perform an edition search over an selected area in a map, displaying all the editions whose location edition is contained in that area.

52. As an administrator I want to be able to see the statistics for the digital library which would offer information such as which collaborator inputs more information into the database, which publications are frequently consulted, which reporters access the digital resources more frequently, etc.

4.2 Variability modeling

The development of a software product line implies managing its variability. To do this, it is necessary to define the variation points between the products, determining the common elements (*commonalities*) and the variable elements (*variabilities*). The common characteristics are those that are present in all the products of the line, while that variable characteristics may be common to several products, but not all [35].

The variability of the software product line developed in this project has been represented using a feature model, which can be seen in figure 4.2 in a collapsed version, although the full

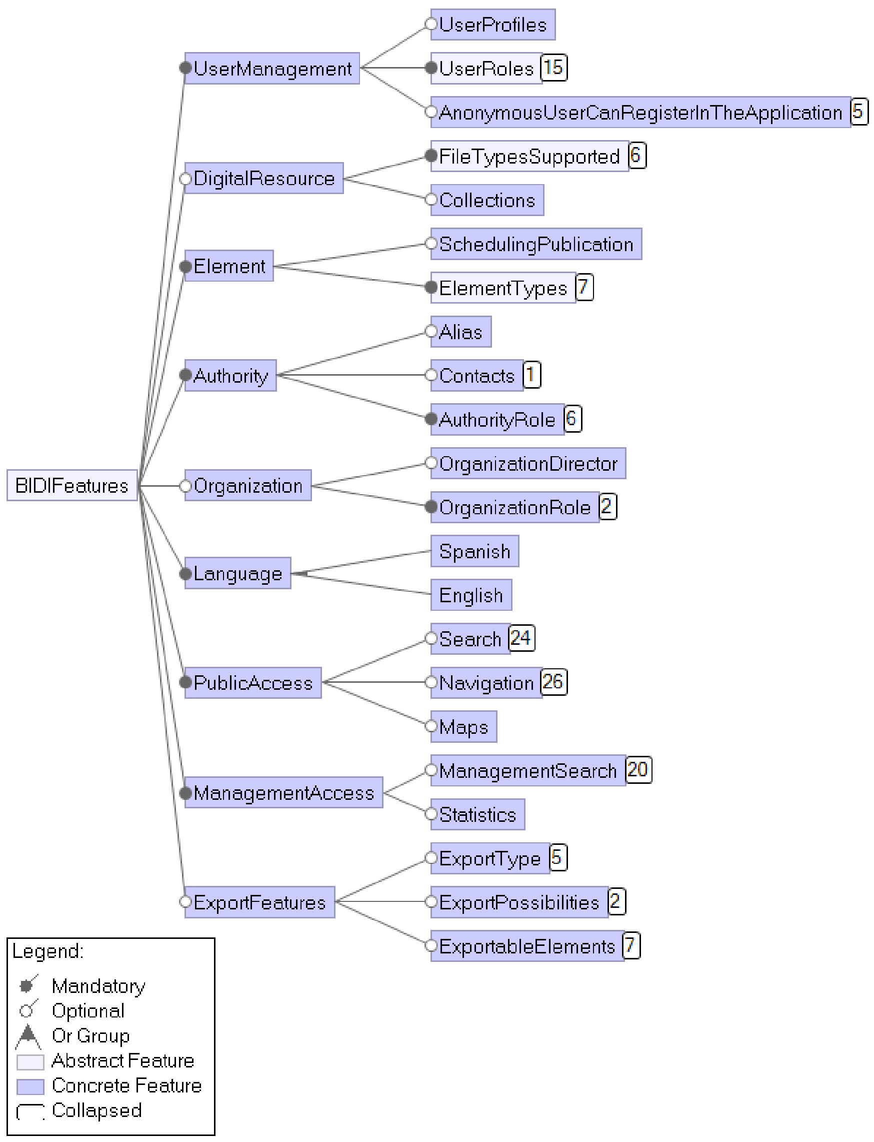


Figure 4.2: Collapsed feature model

model had to be fragmented in several figures for better legibility. A *feature model* is a tree that hierarchically structures the set of functionalities of the system. Within this structure, each characteristic can be decomposed in several sub-characteristics that can be mandatory, optional or alternative. The mandatory characteristics represent the elements common to all products (*commonalities*), while optional characteristics represent variable elements (*variabilities*).

The collection of features modelled is based on the list of features desired for the applications compiled in the requirementes and features mentioned in section 4.1.2 and 4.1.2 after deciding individually which ones could be variable and which ones needed to be mandatory.

A closer look to the different sets of features is available in appendix A, specially for those cases that are not included in the chapter due to space reasons.

As can be seen in figure 4.2, there are different types of relationships between the characteristics defined for the product based on your requirements. The *OR* operator indicates that at least one of the characteristics that are part of the relationship must be selected, being able to select more than one (notation visible in figure 4.3); while the *alternative* operator indicates that you must select a single characteristic from among those that are part of that relationship (notation visible in figure 4.4).

Serving as a more detailed example, figure 4.3 shows the subfeature Navigation (we can see it under Public Access in figure 4.2), and its subfeatures, among we can find the decision of which *HomePage* should be used, or in which way editions, works and authorities can be listed (as lists or with thumbnails/images), as well as decisions regarding pagination.

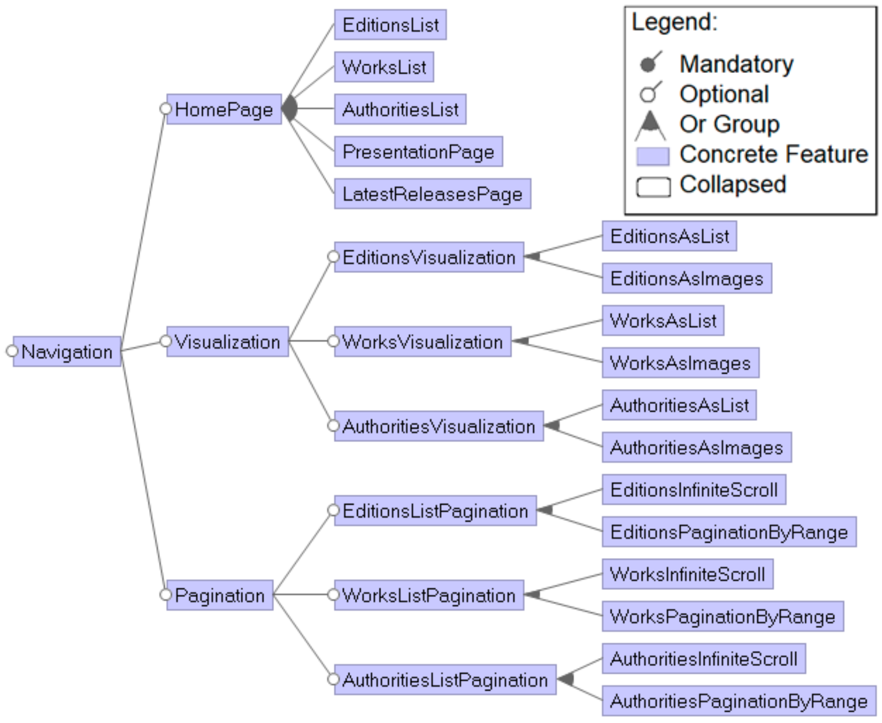


Figure 4.3: Zoom-in of navigation features

In addition, the following restrictions listed in figure 4.5 have been defined for the selection of characteristics.

To be able to store *Collections* of *Digital Resources*, at least one of the features corresponding to *Digitalized Pages*, *Audio Files* or *Video Files* must be selected, since they are the only cases in which one edition might be represented in several files instead of only one unique file representing the whole publication.

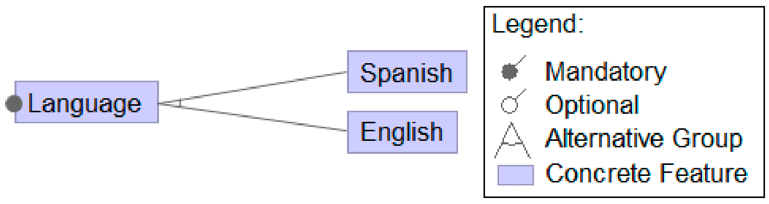
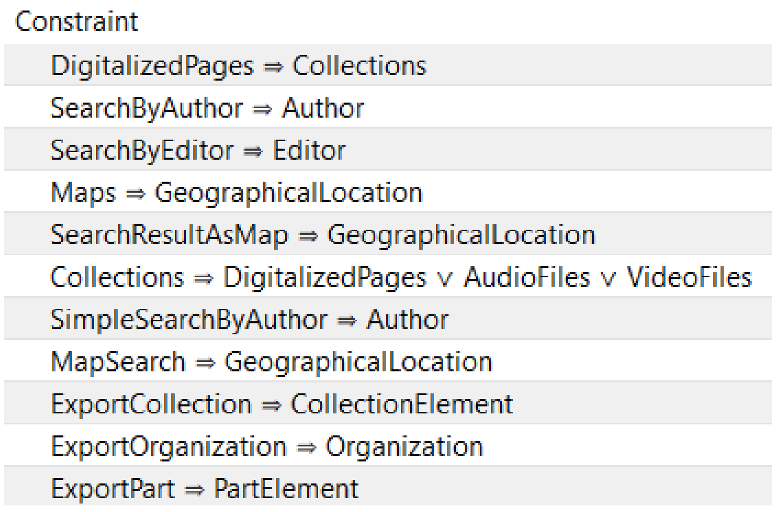


Figure 4.4: Zoom-in of language selection



Constraint
DigitalizedPages ⇒ Collections
SearchByAuthor ⇒ Author
SearchByEditor ⇒ Editor
Maps ⇒ GeographicalLocation
SearchResultAsMap ⇒ GeographicalLocation
Collections ⇒ DigitalizedPages ∨ AudioFiles ∨ VideoFiles
SimpleSearchByAuthor ⇒ Author
MapSearch ⇒ GeographicalLocation
ExportCollection ⇒ CollectionElement
ExportOrganization ⇒ Organization
ExportPart ⇒ PartElement

Figure 4.5: List of constraints for the variability model

All the features including maps depend on *Geographical Location* since it would be impossible to represent a position in a map without the information of their coordinates. Likewise, to have the features related to the Authority Role "*Author*" such as *Search By Author* or *Simple Search By Author*, the feature *Author* from *Authority Role* must be selected. The same case happens with Editor, if the feature *Search By Editor* is desired, the *Authority Role: Editor* must be selected.

Finally, when it comes to the *Export Features*, to be able to select that export type, the entity related to it must be modelled in those cases they are not mandatory. For example, it is not necessary for *ExportEdition* since *Edition* is mandatory, but it is required for *ExportOrganization*.

4.3 Generation tool requirements

This section defines the requirements for the tool that allows a developer to configure a product and then use the software product line to build it.

4.3.1 Actors

Just one actor needed to be defined since only one user will generate the applications:

User of the tool: complete access to all the functionalities provided by the generation tool.

4.3.2 Requirements

This section details the functional and non-functional requirements defined for the generation tool, initially listing them in a general way and later detail them in the product backlog.

Functional requirements

The functional requirements defined for the generation tool are as follows:

- **Visualization and selection of features:** tree-like display and selection permitted for all the features available for the product.
- **Import and export a concrete specification:** offer options to import and export the specification file of a product in JSON format.
- **Download as ZIP:** offer the option of downloading the complete source code for the application into a ZIP file.
- **Generation of static pages:** possibility to model static pages to include additional information e.g. research team members of the library or "About us" page.
- **Generation of a product:** can auto-generate the product with the given features selected previously.

Non functional requirements

The non-functional requirements defined for the tool are as follows:

- **Ease of usage:** the tool should be easy to use, so it should be developed an intuitive and simple web interface that facilitates the generation of products.
- **Compatibility:** the web interface of the tool must be displayed correctly in several browsers like Google Chrome and Firefox.

Product backlog

The general requirements listed above were detailed in the form of user stories for the product backlog, then used as the source of requirements during the development of the tool, and its shown below.

1. As a user of the tool I want to be able to visualize the features offered for the products. These features ought to be presented in a hierarchical structure, so that each feature should contain those sub-features depending on it, and it must ensure that the relationships and constraints defined in the feature model are checked.

2. As a user of the tool I want to be able to select or deselect the desired features for the final product. A checkbox should be displayed beside the feature to enable this.

3. As a user of the tool I want to be able to add a title to the Digital Library that will be generated. Said title must be inserted in a text field. In case no title is specified, a default title will be assigned.

4. As a user of the tool I want to be able to create static pages in addition to the features offered to include extra information about the digital library. The static pages might be a simple HTML page created with a text editor inserted in the web generation.

5. As a user of the tool I want to be able to export the specification containing a list of the selected features for the product and a list of static pages. The file must be a JSON file that will contain the list of selected features and the list of static pages created.

6. As a user of the tool I want to be able to import an specification file. The file must be a JSON file that would contain the list of features to be selected. It can also contain a list of static pages to be created.

7. As a user of the tool I want to generate a final product with said selected features and the static pages I have created. The final product should be compressed into a ZIP file and be downloadable, ready for the deployment.

Chapter 5

Product construction

In this chapter, we complement the analysis of the products to generate, showing the architecture, complete data model and the user interface, and we describe the design and implementation of a particular product of the SPL. Specifically, we decided to build a new version for the BIDISO library [1] mentioned in chapter 2.1. This way, we selected the subset of features presented in this library, and we carried out their implementation. In terms of Software Product Lines Engineering, implementing the whole set of features of a product line can be a huge labour, depending on the domain, and usually there is a work of selection and prioritization of features in order to decide which ones should be implemented first. This is the case for the domain chosen for this thesis, since the total set of features is very large. The advantage of this process of prioritization is the possibility of generating products before finishing the whole development of the SPL.

5.1 Analysis

This section describes those aspects of the product analysis not covered in the section.

5.1.1 System architecture

This section describes the general architecture of the system, performing a high level decomposition of the components that comprise it. As you can see in the figure 5.1, the web application is based on the client-server architecture, and therefore it is composed of two differentiated elements, the server or back-end and the client or front-end. Said architecture favors the separation and independence between both elements, facilitating exchange or modification among them without affecting other parts of the system. In addition, it allows the existence of different clients for the same server and the replication of components to cope with high demand.

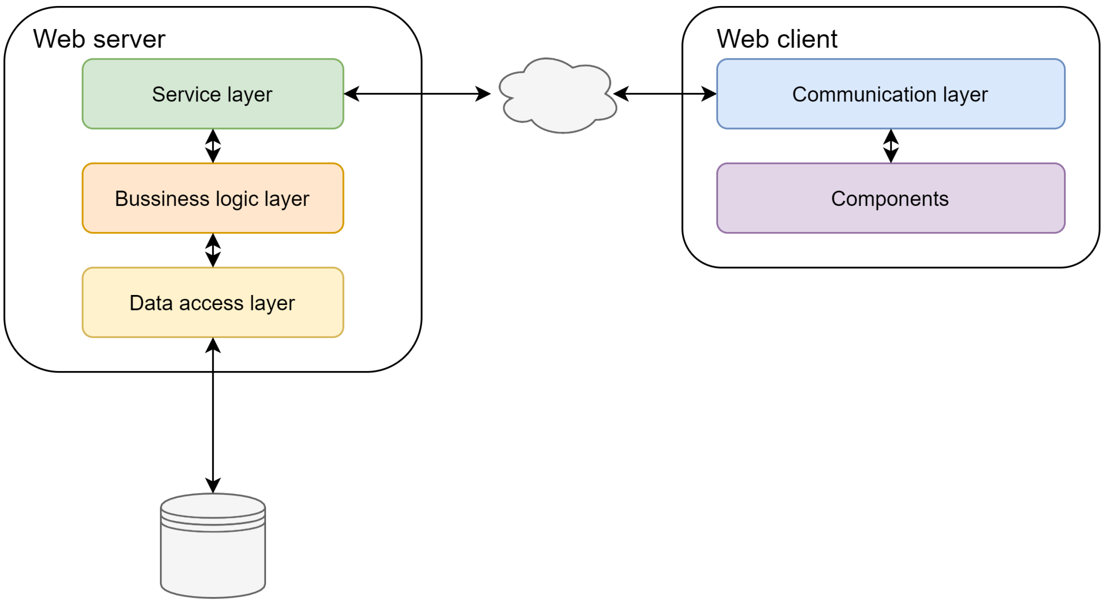


Figure 5.1: System's architecture diagram

Server

The server is composed by a set of services that host the application logic, the information retrieval, persistence and treatment operations. These services are handed to the client through an API, an interface in charge of exposing the set of endpoints that can be invoked from outside.

The server is internally structured following a layered architecture. In this kind of structure system functionality is organized in separated layers, each of one uses just the services available on the layer immediately below them by means of an interface. This architecture favors independence between layers and is also changeable and portable. If its interface is unchanged, a new layer with extended functionality can replace an existing layer without changing other parts of the system. Furthermore, when layer interfaces change or new facilities are added to a layer, only the adjacent layer is affected [36].

The server is structured in a three-tier architecture, as it can be seen in figure 5.1, and it has the following layers:

- **Data access layer**: in charge of the communication with the database, including persistence operations and information retrieval.
- **Bussiness logic layer:** wraps up the main functionalities of the application, implementing the data processing by means of the available functionalities.
- **Service layer:** offers an interface compound by a set of functionalities invocable from

outside. This layer is in charge of managing the petitions received from the client invoking data access layer services and sending the response to the client with the result obtained after the execution of the method. Based on REST (*Representational State Transfer*) paradigm, a software architectural style that defines a set of constraints to be used for creating Web services and enables the communication with other apps through HTTP requests.

Client

The client is the part in charge of presenting the graphical interface to the final users. Based on the actions performed by the user, the client sends the corresponding HTTP requests to the server and updates the screen with the new information when it gets the response,

5.1.2 User interface

This section is destined to provide a high-level explanation of the user interface of the application, manifesting the general structure of the most important screens constituting it, as well as the navigation between them.

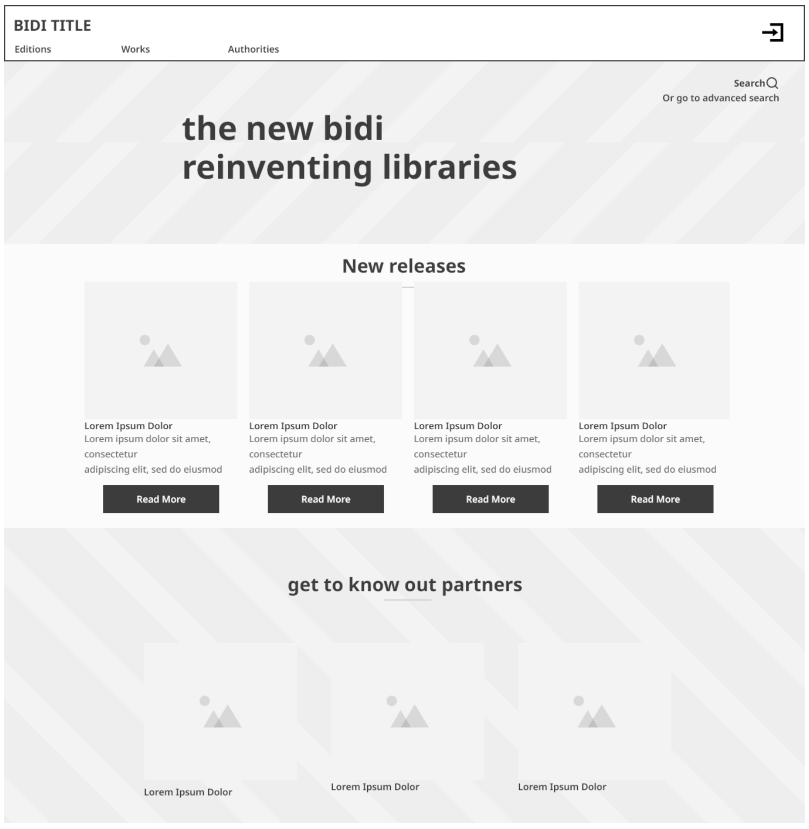


Figure 5.2: Home Page for Latest Releases Page

In the preliminary analysis phase, the detailed design of the web interface was completed through the use of prototypes, in order to facilitate its future implementation and the analysis of requirements. The most distinctive mockups will be explained in detail below, the rest are shown in Appendix B for space reasons.

The main page for the Digital Library is a part of the variability model, so different pages were designed as a Home page. In case the Latest Releases Page is desired, the prototype can be seen in figure 5.2. This page presents an horizontal menu at the top that allows navigation over the web and the possibility of login into the web application. Its main feature is the list that advertises the latest releases updated to our library, which can come really handy for contemporary libraries that include publications frequently. In addition, it can contain a static list of the members of the library.

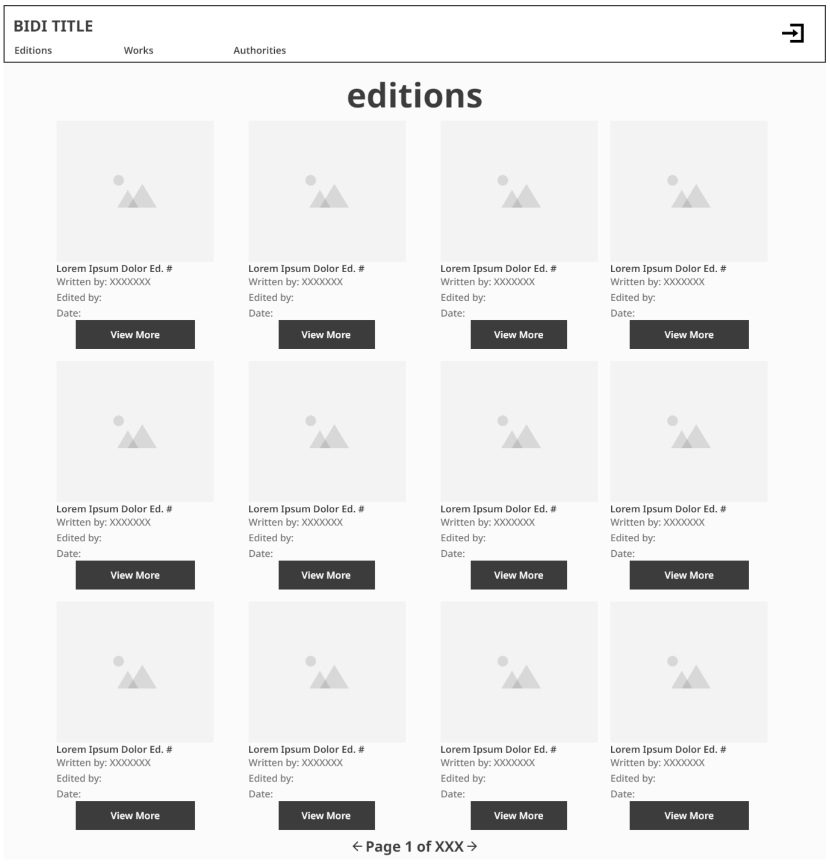
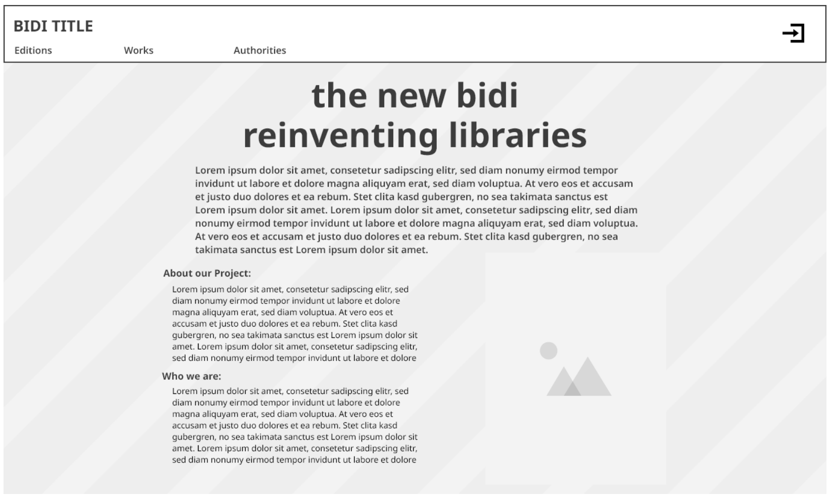


Figure 5.3: Home Page for Editions List Page

Another possibility is to set one of the navigation pages as the home page, for example, the list of Editions published in the digital library, as seen in figure 5.3. In this page, instead of seen just the new releases, we can see all the editions published in the Digital Library. It can be more appropriate for cases of historical libraries or research libraries, where the focus is on the quality of the content instead of its appeal to the public. In case the editions list is selected as Home page, the visualization with images would be preferred over the text list, due to its visual attractiveness. More variability can be found inside this page, since the list can be loaded with an infinite scroll or by pagination. Also, this option can be adapted to any list that may be relevant for the Digital Library, either works, collections or authorities.



the new bidi
reinventing libraries

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

About our Project:

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore

Who we are:

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore

Figure 5.4: Home Page for Presentation Page

In case non of those options covered the needs of the Digital Library, the option of Presentation Page can be chosen instead. This feature allows us to configure an static page as our default Home page and, as it was mentioned earlier, static pages can be modelled at the will of the creator with a simple text editor an basic knowledge of HTML coding. This possibility can be really useful for research teams that may want to introduce their project to let more people know who they are and what their works is about. An example of an hypothetical static page can be seen in figure 5.4.

The feature of Advanced Search might have a dedicated page to accommodate a big browser where we can fill several search criteria fields with the desired parameters. This page gives the option of selection where the lookup must be performed, in the editions database, the works database or the authorities database, depending of what the user is looking for. From there we can switch to the simple search criteria. The search page can be seen in figure 5.5.

One of the key pages for the admin side is the management for publications, either editions, works, parts or collections. Note that when we access the admin side of the web, the layout changes and the menu switches to a vertical position, providing space for more management pages an access to the thesaurus. In the figure 5.6 it can be seen an example for the editions list, where we have the elements displayed in a table with the relevant information to distinguish them and see administration details, such as the status, which is displayed with a

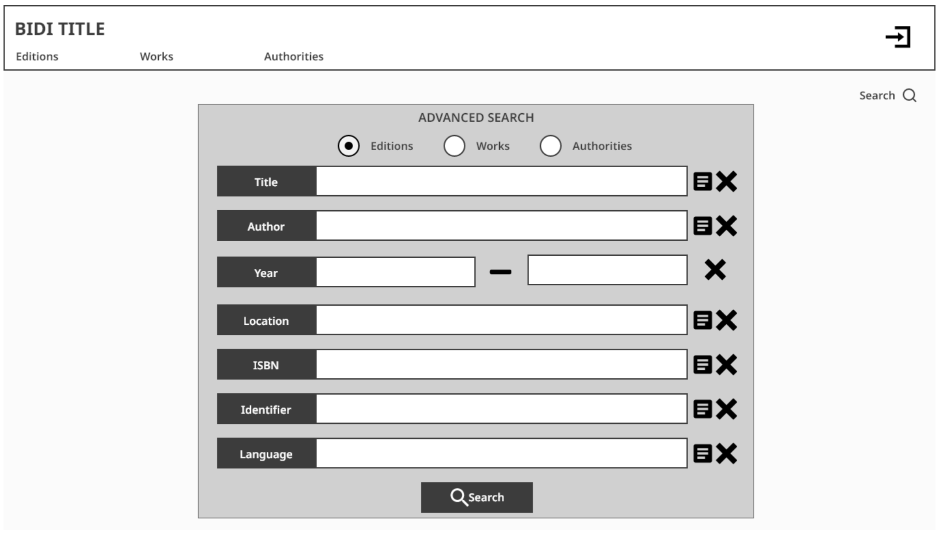
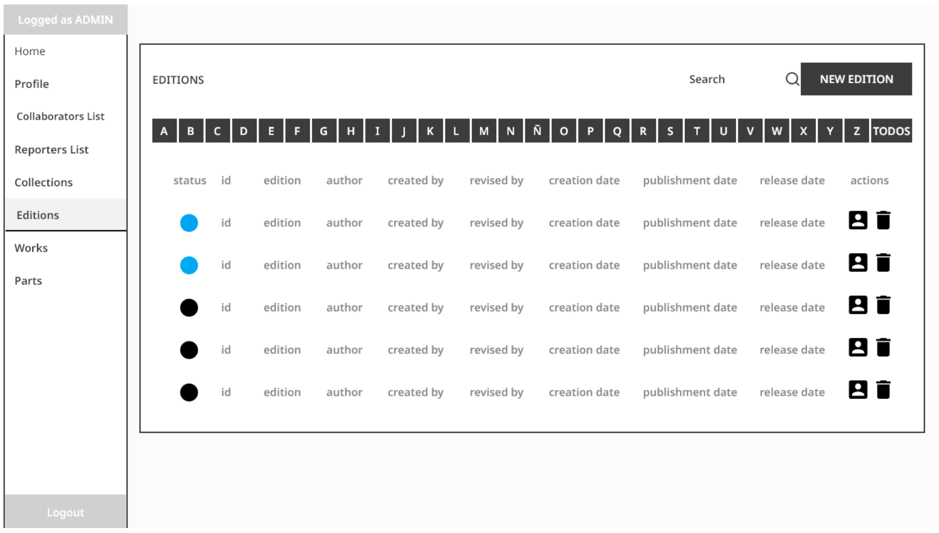


Figure 5.5: Advanced search page

chip indicating whether it is published, revised, pending for revision or saved as draft. The list can be navigated alphabetically and it allows access to each edition, to modify said element or to remove it from the database without having to open a new page. This table also allows to filter the elements by any field, inserting the keywords that must be matched at the top search field. Lastly, from this page we can access the form to create a new edition.



status	id	edition	author	created by	revised by	creation date	publishment date	release date	actions
●	id	edition	author	created by	revised by	creation date	publishment date	release date	
●	id	edition	author	created by	revised by	creation date	publishment date	release date	
●	id	edition	author	created by	revised by	creation date	publishment date	release date	
●	id	edition	author	created by	revised by	creation date	publishment date	release date	
●	id	edition	author	created by	revised by	creation date	publishment date	release date	

Figure 5.6: Admin management - Editions list

After selecting the creation of a new edition, the form for said purpose is shown, as seen in figure 5.7. The main purpose of this page is to facilitate the process of data loading into the system, therefore, the design was conceived so that a user could create an edition and all of their related items from the same page if necessary. In order to achieve that, after the main data necessary for the edition is provided, the form provides several accordion tabs that conform a new sub-form, one per each entity that needs to be associated. In the figure B.23 we can appreciate that the input form for an authority follows the same line as the main fields of its father form, with the peculiarity that it allows the user either to create a new authority or to create a new association between the new edition and an authority that is already stored in the system. To do that, the user can start typing the name of the authority and, if there is a match in the database, the rest of fields disappear and the authority gets added to a list of authorities related to that specific edition. This behaviour was decided based on the idea that, if the authority already exists, the user would not want to update it for each edition, and in case the user wants to update a certain authority they just have to search it in the authority list and edit it from there. The rest of entities follow the same approach, although

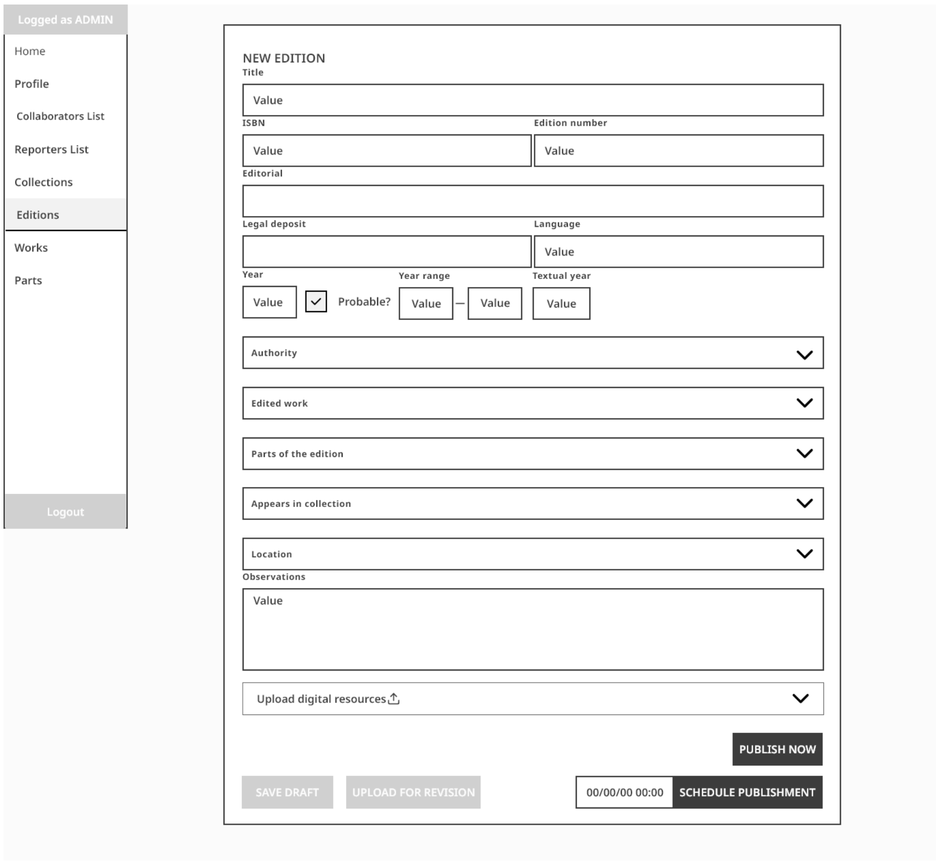


Figure 5.7: Admin management - Edition form

their visual representation is located in Appendix B due to lack of space. Likewise, when it comes to modelling a work with related publications, a collection or a part, the accordion system is applied. In the case of creating a new authority, the only thing that is collected as an accordion is the list of known alias and the set of contact information, including the list of social network accounts in case they are modelled.

The implementation of a new edition location is noteworthy in the case that geographical location is included. If that was the case, the ideal behaviour would be to pin the location in a map, get the name from the coordinates and be able to rename that pin in case that historical location is needed, in which case we want to store the ancient name of the place where the edition was originally conceived. Nonetheless, features including maps will not be implemented in this prototype but the corresponding mock-up in Appendix B can be used as reference for future work.

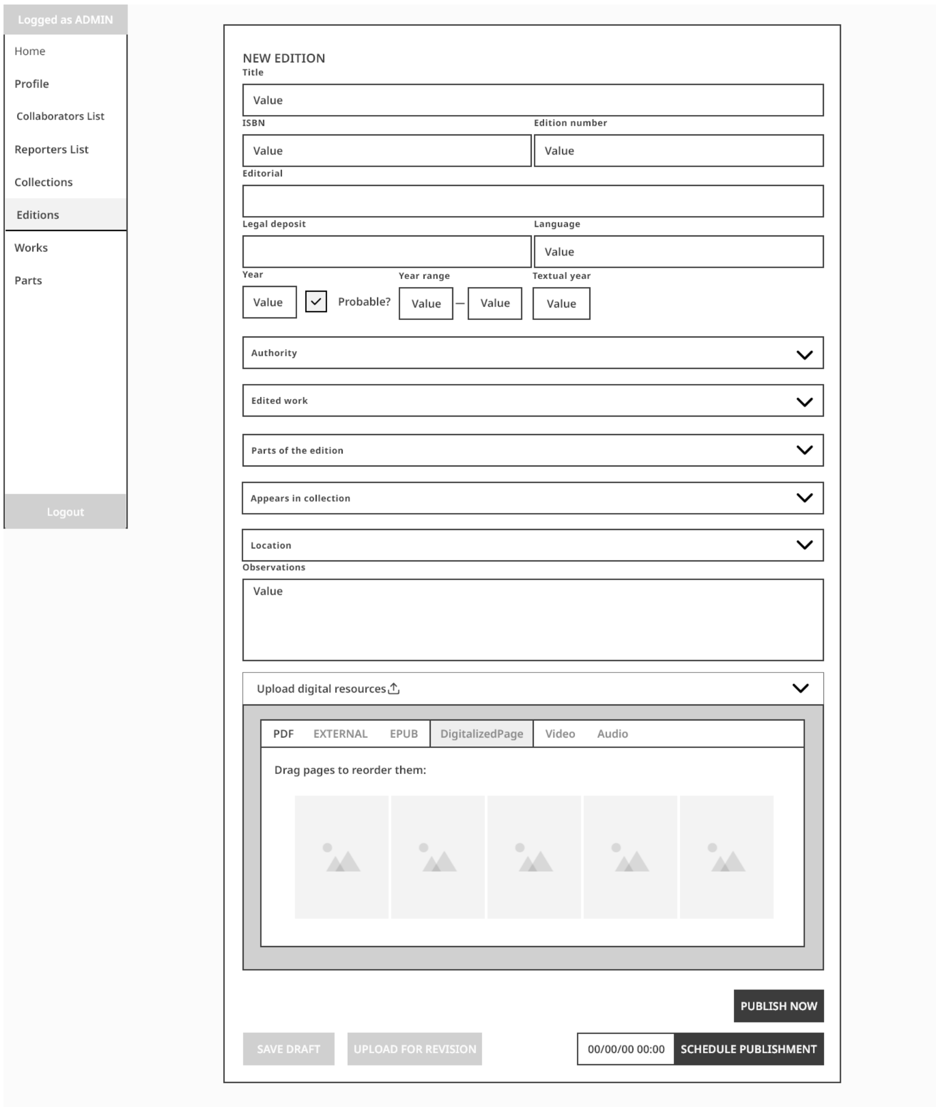
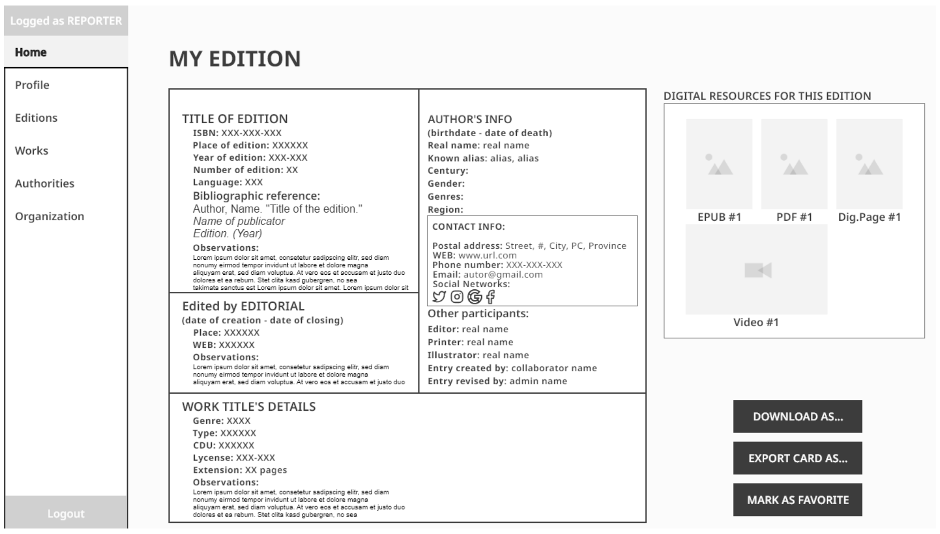


Figure 5.8: Admin management - Edition form with digitized pages

Lastly, taking into account that the basic feature of a digital library is to offer digital copies of a publication to its users, the digital resource upload was implemented following the same schema, an user-friendly and accessible file input that allows loading several files at the same time, with one tab per file type. The interface changes according to the type of input desired, facilitating the process in each case. In figure 5.8 the upload of digitalized pages is illustrated.

Finally, one of the most common requests for the web application will be to display the detailed information available regarding a publication. The same quandary is present, just like in the creation, it is not just an edition or a work, the web must display in a legible manner the data regarding the edition, its authors, the work or works related, the collections that include it, the organization in charge of editing the publication and the set of digital resources available for each of them. Trying to avoid an infinitely long page, a grid based on small boxes for each element was preferred as depicted in figure 5.9, which also allowed the information to be encapsulated and gives the user a clear separation between the different elements composing the edition itself.



MY EDITION

TITLE OF EDITION
ISBN: XXX-XXX-XXX
Place of edition: XXXXXX
Year of edition: XXX-XXX
Number of edition: XX
Language: XXX
Bibliographic reference:
Author, Name. "Title of the edition."
Name of publicator
Edition. (Year)
Observations:
Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit

Edited by EDITORIAL
(date of creation - date of closing)
Place: XXXXXX
WEB: XXXXXX
Observations:
Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo

AUTHOR'S INFO
(birthdate - date of death)
Real name: real name
Known alias: alias, alias
Century:
Gender:
Genres:
Region:

CONTACT INFO:
Postal address: Street, #, City, PC, Province
WEB: www.url.com
Phone number: XXX-XXX-XXX
Email: autor@gmail.com
Social Networks:

Other participants:
Editor: real name
Printer: real name
Illustrator: real name
Entry created by: collaborator name
Entry revised by: admin name

WORK TITLE'S DETAILS
Genre: XXXX
Type: XXXXXX
CDU: XXXXXX
Lycense: XXX-XXX
Extension: XX pages
Observations:
Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea

DIGITAL RESOURCES FOR THIS EDITION

Figure 5.9: Edition details

Due to space issues it became unaffordable to list all the mock-ups designed and explain them in detail, although they can all be found in Appendix B.

5.1.3 Conceptual data model

This section presents the data model designed in the preliminary analysis phase to determine how the structure of the database should be implemented. To this end, the conceptual model of the database was developed, specifying the entities, their attributes and the relationships that exist among them. This data model distinct itself because it varies depending on the features selected from the feature model explained in section 4.2. Since this engineering field is relatively new and still evolving, no standard notation was found for variability representation in class diagrams, so a proper notation had to be created from scratch.

The condition of the selection of a feature is represented as “[if *featureName*]”, meaning that the element annotated will exist if the condition is true i.e. the feature is selected. On the contrary, if an element exists only if the feature is not present, the notation will be “[*if **not** featureName*]”. To represent changing relationships a dotted line was chosen, annotated with the name of the feature that would active its existence.

The stated data model may be examined in the figure 5.10, although due to sizing issues closeup figures will be added to support the explanation:

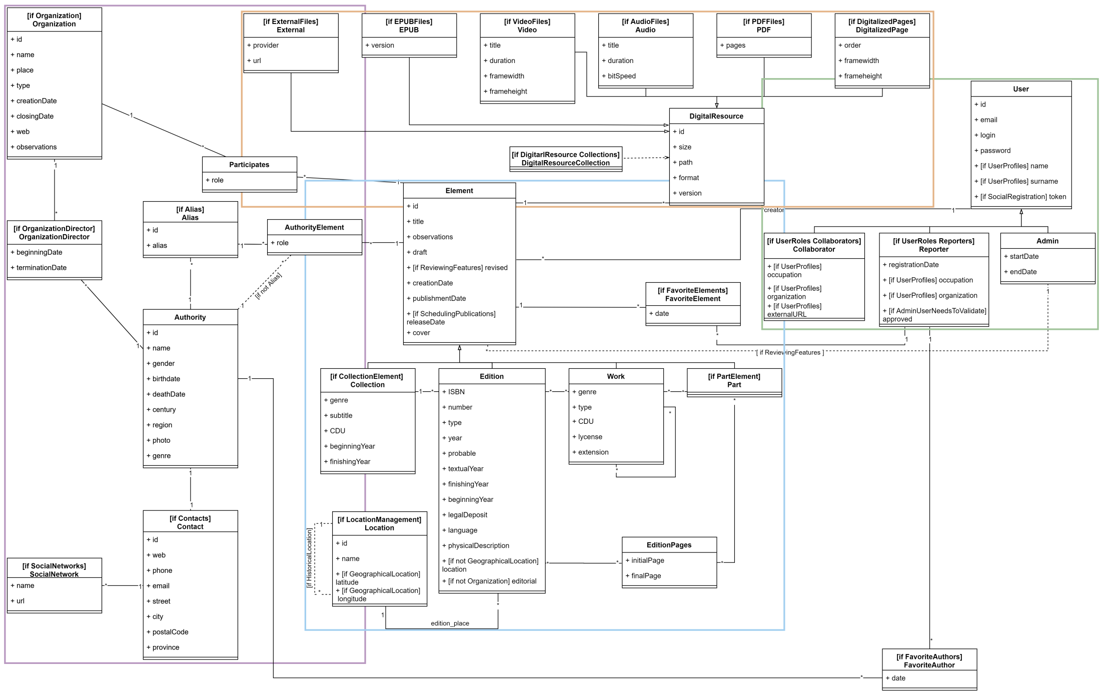


Figure 5.10: Data model

Firstly, taking a look to the section with the blue border, we can see the classes related to the publications of the digital library, and can be seen in the figure 5.11. Since the purpose of a library is to store literary publications, this section should be considered as the core of the data model, without which the application would lose its purpose. It contains the following classes:

- **Element:** This class is a generalization that models the common attributes to all the elements and takes charge of the relationships with the rest of classes that are not subclasses from itself. Each element can have several authorities and an authority can par-

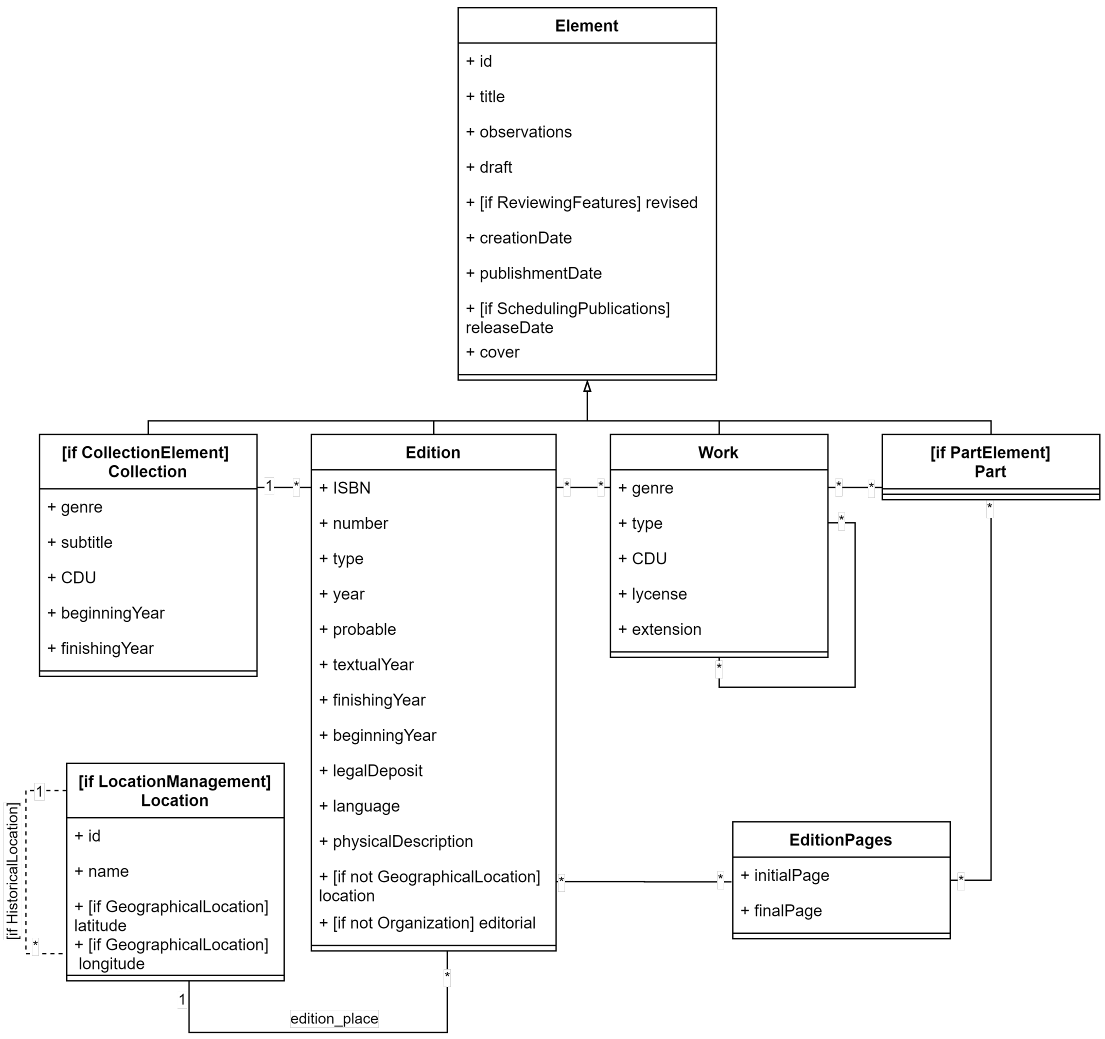


Figure 5.11: closeup to the element classes

ticipate in several elements, so **Element** should have a *ManyToMany* relationship with **Authority**, although in this case it is desired to store which role is developed by which authority in each publication (writer, editor, prologuist, translator or illustrator) so that a intermediate class is created under the name **AuthorityElement** (that can be seen in figure 5.12). **Element** has a *OneToMany* relationship with **AuthorityElement**, an **Element** can have many authors or be anonymous, and an **AuthorityElement** will only be stored in case they have an **Element** associated. A similar situation occurs between **Organization** and **Element**, since an **Organization** can develop different roles for each publication, it can be a Library or a Editorial, and an **Element** can be edited by an organization different to the one storing its physical copy. To be able to model this, the class **Participates**. **Element** has a *OneToMany* relationship with **Participates**. **Ele-**

ment has a *OneToMany* relationship with **DigitalResource** since an Element can have several resources associated but a Digital Resource will only correspond to one element. **Element** has a *ManyToOne* relationship with **User** that represents which User (Admin or Collaborator) loaded an element into the database. **Element** has a *ManyToOne* relationship with **Admin** that models which **Admin** revised an element already created by a Collaborator before publishing it and only exists if the feature *ReviewingFeatures* is selected. Finally, **Element** has a *OneToMany* relationship with **Reporter** and only exists if the feature *FavoriteElement* is selected.

- Id: Element identifier.
- Title: Title of the literary publication.
- Observations: Additional comments regarding the publication.
- Draft: Boolean representing the draft status. True if the element is saved as a draft, false if the element is ready for publishment.
- Revised: Only if *ReviewingFeatures* are active. Boolean representing if an element was revised by an Admin or not.
- CreationDate: Date and time at which an item was created, it is not setted if the item is saved as a draft.
- PublishmentDate: Date and time at which an element is complete and, in case there is ReviewingFeatures, revised by an Admin. Ready to be released to the public.
- ReleaseDate: Date and time at which an element is made public on the web, can be different from PublishmentDate if the option SchedulingPublications is active, in other case this attribute does not exist.
- Cover: Image of the physical cover of a literary publication.

• **Collection:** Type of literary publication that compiles a set of editions. Only modelled if the feature *CollectionElement* is selected. **Collection** has a *OneToMany* relationship with **Element** as it was considered that a Collection is a rare element and an **Edition** will only be present in one **Collection** if the case is given. See figure 5.11.

- Genre: Literary genre of the Collection.
- Subtitle: Additional title to complement the Element title in necessary case.
- CDU: Corresponds to the Universal Decimal Classification or, in spanish, *Clasificación Decimal Universal*, an extended classification system for libraries.
- Beginning year: Year at which the collection began its composition.

  - Finnishing year: Year at which the collection is considered finished and published.

- **Edition:** The most common type of literary publications together with **Work**, represents the "printed" copy of a literary text, although in present days a digital copy can be considered an edition without being physically printed. An **Edition** can edit several **Works**, even if the most common case is that each Edition represents only one Work, so **Edition** has a *ManyToMany* relationship with **Work**. The participation is partial since an **Element** has a *ManyToMany* relationship with **Authority** can be conformed by **Element** has a *ManyToMany* relationship with **Authority** and a **Element** has a *ManyToMany* relationship with **Authority** can be a simple text without edition. **Edition-Part** needs the backup of an intermediate class to model the order in which the Parts must be stored, so **Edition** has a *ManyToMany* relationship with **Part**. Lastly, if the feature *LocationManagement* is selected, the class **Location** is created and has a *OneToMany* relationship with **Edition**. Given the case that *LocationManagement* is not selected, the edition location would be modelled as a simple attribute.

  - ISBN: Code assigned to each edition of a book.
  - Number: Number of edition that is being stored.
  - Type: Type of edition.
  - Year: Year of edition.
  - Probable: Boolean that represents if the year is exact or a probable year. If probable is false, the exact year of edition is known; if true, the year is unknown and but can be delimited in a range.
  - Textual year: If probable is true, the value of textual year is stored. It depends of the values beginning year and finnishing year, so if a edition was published among 1960 and 1968, the textual year would be 196?.
  - Finnishing year: The ending year of the estimated range of publishing for an edition.
  - Beginning year: The beginning year of the estimated range of publishing for an edition.
  - Legal deposit: Number associated to the legal deposit of the edition.
  - Language: Language of the edition.
  - Physical description: Textual description of the physical aspect of an edition.
  - Location: In case the *LocationManagement* feature is not present the location of edition is stored as an attribute with the name of the place.

  - Editorial: In case the *Organization* feature is not present the editorial is stored as an attribute with the name of editorial.

- **Work:** Represents a literary text without the edition process. **Work** can be present in several **Editions**, as it was stated previously. **Work** can have a *ManyToMany* relationship with **Part**. **Work** can even contain another **Work**, so it has a *ManyToMany* relationship with itself.

  - Genre: Literary genre of the Collection.
  - Type: Type of work.
  - CDU: Corresponds to the Universal Decimal Classification or, in spanish, *Clasificación Decimal Universal*, an extended classification system for libraries.
  - Lycense: Number of the author's rights lycense.
  - Extension: Extension of the work, in number of pages.

- **Part:** Represents a fragment of a Work that can be referenced multiple times. **Part** has relationships with **Work** and **Edition** but they will not be mentioned since they were explained in the paragraphs corresponding to those classes previously. Part does not have special attributes.

- **Edition pages:** Represent the order in which a **Part** is located inside an **Edition**.

  - Initial page: First page of the Edition at which the Part starts.
  - Final page: Last page of the Edition at which the end of the Part is met.

- **Location:** This class only is present in case that *LocationManagement* feature is selected and represents the physical place at which the edition of a work took place. **Location** has a *OneToMany* relationship with **Edition**. In case the feature *HistoricalLocation* is selected, **Location** can have a *OneToMany* relationship with itself to reference the same physical place with different names along the years. See figure 5.11.

  - Id: Identifier of location.
  - Name: Name of the location.
  - Latitude: In case the *GeographicalLocation* feature is selected, the coordinates for the latitude of the place is stored.
  - Longitude: In case the *GeographicalLocation* feature is selected, the coordinates for the longitude of the place is stored.

Afterwards, taking a look to the section with the lilac border, we can see the classes related to the authorities stored in the digital library, and can be seen in the figure 5.12. It contains the following classes:

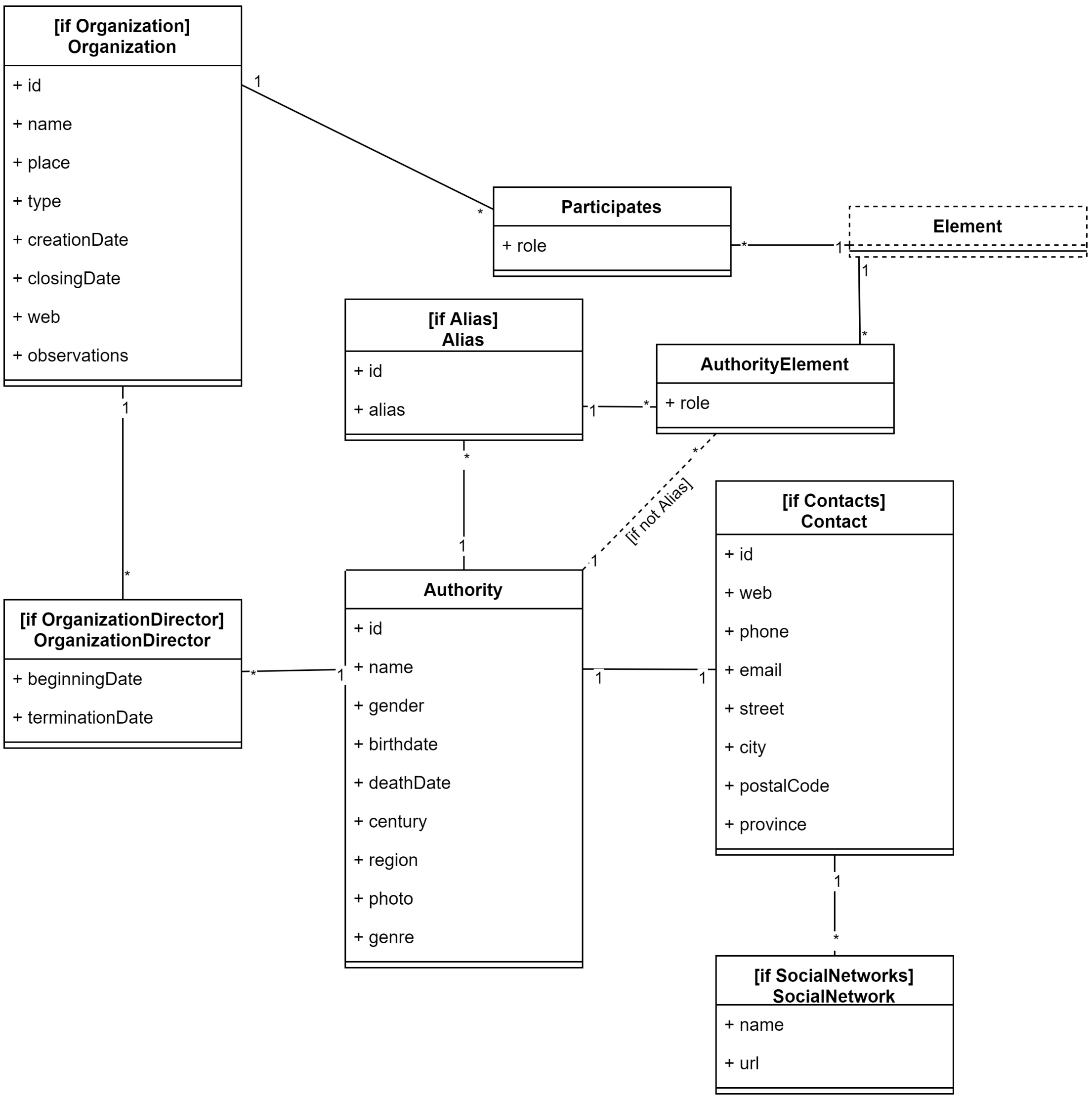


Figure 5.12: Closeup to the authority and organization classes

- **Authority:** The most relevant class of this section since it models all of the information available about an authority. A detailed study of this class is needed since it can be a strong point of variability after the inclusion of the SPL annotations. To begin with, an **Authority** can have a list of known **Alias** in case the *Alias* feature is desired, stored in a separated class with a *OneToMany* relationship. As it was mentioned earlier, the authors of the **Elements** of the library are modelled with an intermediate class called **AuthorityElement** which, in case *Alias* exists, stores which alias was used to sign each of the elements, in which case **Authority** has a *OneToMany* relationship with **Alias**. The variability comes when the user does not desire to store any alias for any authority, in that case the class **Alias** does not exist, so the relation from **Authority** to **Alias** disappears too, switching to a direct *OneToMany* relation from **Authority** to **AuthorityElement** instead. When it comes to modelling the director of an organization after selecting the homonym feature, it is desired to know the period each person spent working as a director of each organization, and for that purpose the class **OrganizationDirector** is created with a *ManyToOne* relation with **Authority** since for each period of time only one authority can be set as director. In case the feature *OrganizationDirector* is not selected, **Authority** and **Organization** will not have any kind of association. Lastly, **Authority** can have a reference to the contact information of said **Authority** in case the feature for this option is selected, which results in a *OneToOne* relationship.

  – Id: Identifier of Authority.
  – Name: Name of the authority.
  – Gender: Gender identity of the authority.
  – Birthdate: Date of birth of the authority.
  – Deathdate: Date of death of the authority, in case they are deceased.
  – Century: Century of activity of the authority, useful in case of historical libraries.
  – Region: Originary region of the authority.
  – Photo: Profile picture of the authority.
  – Genre: Main literary genre they wrote.

- **Contact:** Stores the contact information of an **Authority**. At the same time it can contain a set of Social network accounts belonging to the authority, in case *SocialNetworks* feature is selected.

  – Id: Identifier of Contact.
  – Web: Link to the web page of the authority.
  – Phone: Phone number to contact the authority.
  – Email: Email address of the authority.
  – Street: Street of the postal address of the authority.
  – City: City of the postal address of the authority.

  - Postal code: Postal code of the authority.
  - Province: Province of the postal address of the authority.

- **Social Network:** Stores a reference to a social network account in case the *SocialNetworks* feature is selected.

  - Name: Name of the social network.
  - Url: Link to the profile of the authority.

- **Alias:** Stores an alias associated to an authority if the feature *Alias* is selected. **Alias** has a *OneToMany* relationshio with **AuthorityElement**.

  - Id: Identifier of Alias.
  - Alias: Pseudonym used by the author.

- **AuthorityElement:** This class was mentioned in the paragraph discussing the **Element** and in the previous paragraphs regarding the variability of the model.

  - Role: Role developed by an authority in the creation of an element, can be Author, Editor, Illustrator, Prologuist, Translator or Printer.

- **Organization:** Stores the information available about an Organization in case the according feature was selected. In addition to the aforementioned relation with *OrganizationDirector*, which is a *OneToMany*, it possesses a relation with **Element** by means of an intermediate class, **Participates**, used to model the role an **Organization** takes in the creation of an **Element**.

  - Id: Identifier of Organization.
  - Name: Name of the organization.
  - Place: Location where the sede of the organization was.
  - Type: Type of organization.
  - Creation date: Date of opening of the organization.
  - Closing date: Date of closing of the organization.
  - Web: Web page associated to the organization.
  - Observations: Comments relevant to the content of the library made by the collaborators.

- **OrganizationDirector:** Models the director of an organization in a period of time. Its relationships where already mentioned so they will be skipped to avoid redundancy.
  - Beginning date: Date in which an authority took the role of director of an organization.
  - Termination date: Date in which an authority dropped the role of director of an organization.
- **Participates:** Models the relation among an organization and an element. Its relationships where already mentioned so they will be skipped to avoid redundancy.
  - Role: Role developed by an organization in the existence of an element, can be Editorial or Library.

Furthermore, taking a look to the section with the green border, we can see the classes related to the users with access to the digital library, and can be seen in the figure 5.13. It contains the following classes:

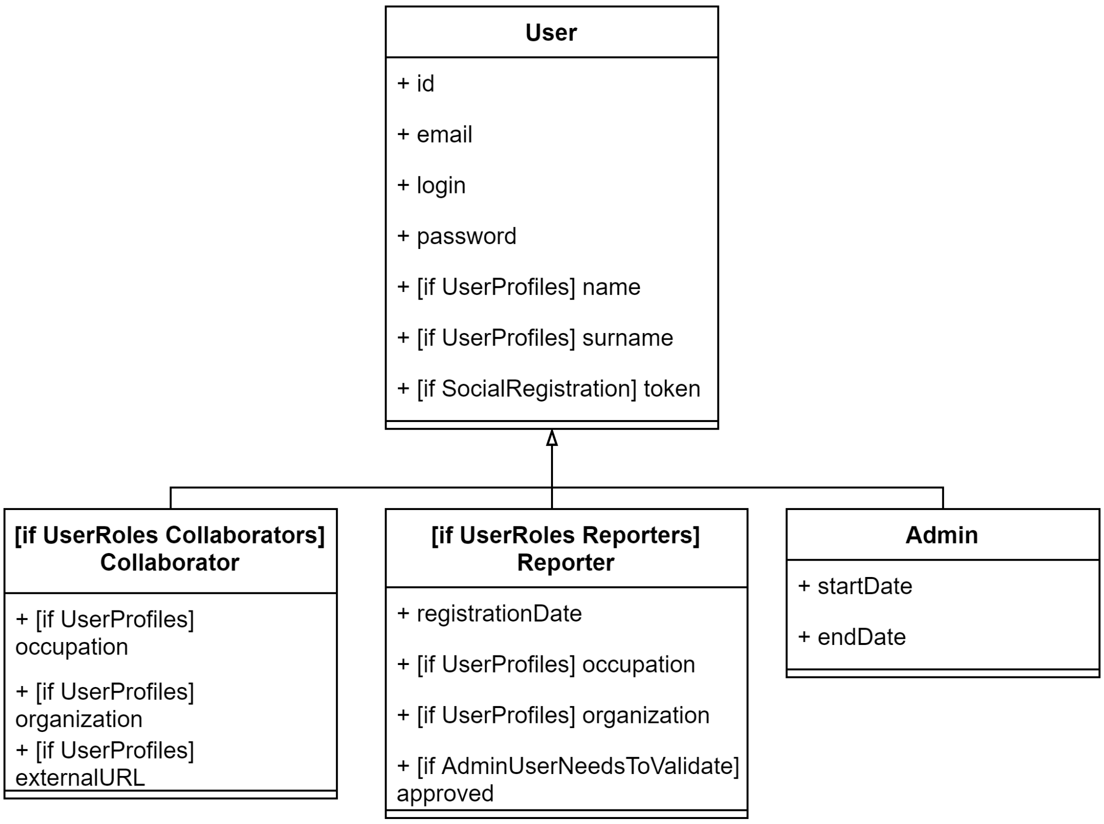

User
+ id
+ email
+ login
+ password
+ [if UserProfiles] name
+ [if UserProfiles] surname
+ [if SocialRegistration] token
[if UserRoles Collaborators] Collaborator
+ [if UserProfiles] occupation
+ [if UserProfiles] organization
+ [if UserProfiles] externalURL
[if UserRoles Reporters] Reporter
+ registrationDate
+ [if UserProfiles] occupation
+ [if UserProfiles] organization
+ [if AdminUserNeedsToValidate] approved
Admin
+ startDate
+ endDate

Figure 5.13: Closeup to the user classes

- **User:** This class is a generalization that models the common attributes to all the users and takes charge of the relationships with the rest of classes that are not sub-classes from itself. **User** has a *ManyToOne* relationship with **Element** to keep track of which publication was uploaded to the database by which user (Admin or Collaborator, although this constraint is modelled in the server).
  - Id: Identifier of User.
  - Email: Email address of each individual user:
  - Login: Login name used to access the application by the user.
  - Password: Password to access the application.
  - Name: Name of the user, only in case that the extended UserProfiles feature is selected.
  - Surname: Surname of the user, only in case that the extended UserProfiles feature is selected.
  - Token: Token to store the social registration in case that SocialRegistration feature is selected.
- **Admin:** Sub-class of **User** that models those user with Admin authority, which means they have full access to the features offered by the digital library and can modify its content. If the feature *ReviewingFeatures* is selected, the login of the Admin is stored as the reviser of the Element and becomes responsible of their publishing. This situation is modelled with a *ManyToOne* relationship from **Admin** to **Element**.
  - Start date: Date in which they started working for the digital library.
  - End date: Date in which they terminated working for the digital library in case they are not a part of the administration team anymore.
- **Collaborator:** Sub-class of **User** that stores the users with Collaborator authority in case the *UserRoles Collaborators* feature is selected. This class is composed by the people that conforms the research team behind the digital library, the workers, the scholars, etc. The collaborators have almost full access to all the features of the application, although they are subordinated to the approval of an admin in the case *ReviewingFeatures* are needed.
  - Occupation: Position the collaborator occupies.
  - Organization: Organization the collaborator is working for.

  - External URL: Link to an external page where more information about the collaborator can be found.

- **Reporter:** Sub-class of **User** that stores the users with Reporter authority in case the *UserRoles Reporters* feature is selected. This class is composed by the people with public access to the digital library and its digital resources. When the features *FavoriteElements* and *FavoriteAuthors* are selected, this class is related to them with *OneToMany* relationships.

  - Registration date: Date and time of registration in the application.
  - Occupation: Occupation of the reporter.
  - Organization: Organization the reporter is part of.
  - Approved: Boolean signaling if the new account has been approved or not by an admin, only present if *AdminUserNeedsToValidate* is selected.

Lastly, the orange section corresponding to the digital resources classes can be seen at the appendix C. No further explanation will be given since it does not present any peculiarity or complexity on its design, just a dependency among **DigitalResourceCollection** and **DigitalResource** to represent there must not exist a Collection of an element that does not exist.

It must be highlighted that the data model represented in these figures is composed by all the possible features that can conform a product from the SPL, but when it comes to the generation of the digital library it will vary according to the variabilities mentioned in each case.

5.2 Design

5.2.1 Technological architecture of the system

The intention of this section is to fulfill the architecture analysis carried out in the previous section taking an in depth study of the technologies included in each of the components of the application. Figure 5.14 illustrates the technological architecture of the system.

Database

The selected technology to implement the information storage was PostgreSQL, free and open-source relational database management system (RDBMS) emphasizing extensibility and

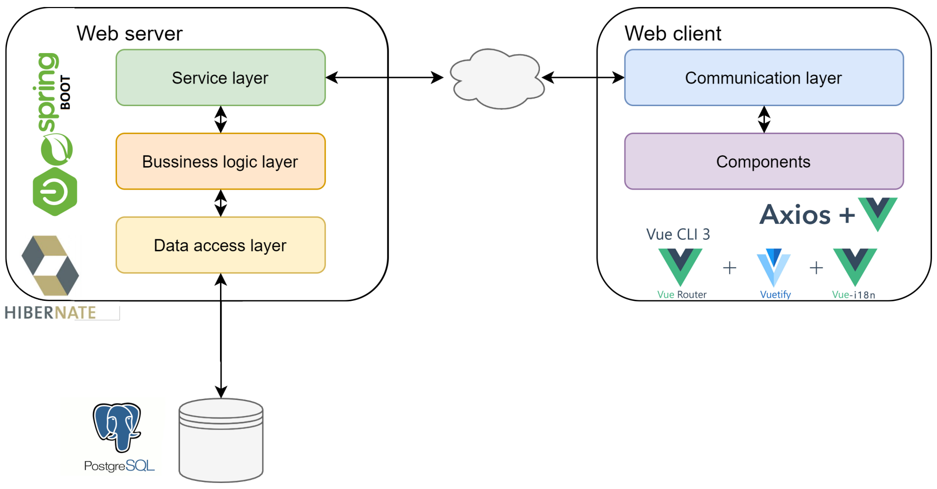


Figure 5.14: Technological architecture for the system

SQL compliance. Two databases were created, the first one to carry out automated and manual tests against the REST service, and the second one to store the real data generated by the application since its execution.

Server

In the development of the server, it has been chosen to use Open JDK 11 together with the functionalities offered by the Spring framework. In addition, Hibernate has been adopted, an object-relational mapper that has facilitated data persistence and information retrieval from the database by supporting work with annotated objects rather than directly with the database tables.

The internal structure of the server follows the upcoming schema:

- **Repositories** for the **Data Access Layer**, classes in charge of the communication with the storage service which carry out the information retrieval and persistence operations. Said repositories are based on the Data Access Object (DAO) pattern, whose main advantage is the capability to isolate the Data Access Layer from the rest of the application in a way that modifications on the storage service do not affect other parts of the system. Repositories offer the aforementioned services to the Business Logic Layer using an interface.

- **Services** for the **Business Logic Layer**, classes that cluster a set a related functionalities later offered to the layer above by means of an interface. Services use those functionalities extended by the repositories to implement the methods' logic.

- **Controllers** for the **Service Layer**, classes in charge of receiving the requests from the client and sending back the response after invoking the methods extended by the services. Controllers get HTTP requests from the client and deliver a response in JSON format through another HTTP request.

To regulate and ensure the security of the application, use has been made of Spring Security, an access control and authentication framework used to safeguard applications based on Spring [37]. This framework allowed the configuration of authentication in the application and provided access restrictions to requests through the use of filters. Connecting to communication between the client and the server, the *JSON Web Token (JWT)* standard has been used to manage user authorization.

5.2.2 Client

On the client side, the preferred implementation was based on Vue.js, a JavaScript framework used to develop web interfaces and *Single Page applications (SPA)*; and Vuetify, a framework that combines the power of Vue.js and the aesthetic of a Material Design component

To complement Vue's functionalities, the following libraries have been used:

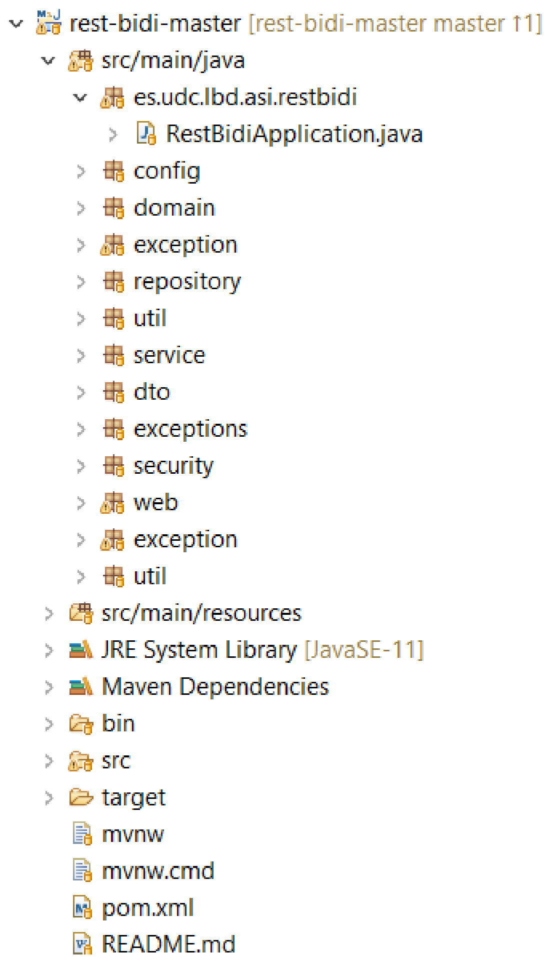
- **Vue-router:** Vue's official router for SPA [38]. This library allows navigation among pages with the possibility to configure interceptors to execute actions beforehand or afterwards.
- **Vue i18n:** internationalization plugin of Vue.js [39]. It easily integrates some localization features to any Vue.js Application, enabling the automatic translation of any desired display to the expected language specified in the locale.
- **axios**: HTTP client based on promises employed on the development of Javascript applications [40]. It works both on browsers as well as in Node.js platforms. Axios eases the implementation of web services that manage information in any file format and enables the configuration of the interceptors to the requests, making it possible to execute certain activities before they get sent to the server or, when they come as responses, before they reach the final component.

5.2.3 Application design

This section will detail the internal structure of the application components.

Server

The server package structure is depicted in figure 5.15. From each package protrudes:



```
rest-bidi-master [rest-bidi-master master ↑1]
  src/main/java
    es.udc.lbd.asi.restbidi
      RestBidiApplication.java
    config
    domain
    exception
    repository
    util
    service
    dto
    exceptions
    security
    web
    exception
    util
  src/main/resources
  JRE System Library [JavaSE-11]
  Maven Dependencies
  bin
  src
  target
  mvnw
  mvnw.cmd
  pom.xml
  README.md
```

Figure 5.15: Package structure of the application server

- src/main/java: contains all the Java classes implementing the server functionalities. Its internal structure goes as follows
  - **restbidi:** contains the main class, responsible of the execution of the application.
  - **config:** contains the files in charge of the configuration of the application, including the security configuration and the properties.
  - **model:** contains the entities and DTOs that conform the application.
    - **domain**: contains the classes responsible of modelling how the information will be stored in entities inside the database.
    - **exception**: stores the general exceptions for the Business Logic Layer.
    - **repository**: contains the interfaces and repositories (Data Access Objects) of the application.
    - **service**: contains the DTOs and the services of the application.

  - **security:** contains the classes responsible of keeping the security checks of the application.
  - **web:** contains the controllers responsible of managing the HTTP requests.

- **src/main/resources:** contains the configuration file.

Detailing some elements for the main packages:

Entities

Persistent classes that represent the entities designed in the data model (figure 5.10). These classes have a sequence of attributes that display both the properties defined for each entity in the data model, such as relationships between entities. Also, they have read and write methods (getters and setters) to allow the entrée to its attributes, constructors, and the equals, hashcode, and toString methods.

All entities are annotated with @Entity in order to hint to the object-relational mapper which are persistent classes and must be mapped to the database. In addition, other Hibernate annotations were applied on attributes to determine how the tables should be generated in the database.

Repositories (DAOs)

Elements in charge of providing methods to perform the operations of persistence and recovery of the information of the database. There is a repository for each entity.

All repositories are annotated with @Repository and extend the GenericDaoJpa that implements the EntityManager interface adopted to manage persistence. This configuration allows make use of the benefits provided by Spring Data, such as the implementation automatic method following a naming convention, the ability to implement queries by using the @Query annotation, or the facilities when performing the pagination of results.

Services

Classes containing most of the server logic implementation. Each service groups together a set of related functionalities, and all are annotated with @Service. They all count with an interface, through which they offer their methods to the other components, and with a class that performs the implementation of those methods.

The services, through dependency injection, make use of the methods exposed by repositories and by other services through their interfaces.

Controllers

Classes in charge of the external exposure of the services through a REST API. Controllers are responsible of receiving the requests from the client, invoking the corresponding methods from the service and sending back the response to the client. All of them include the annotation @RestController by Spring, which eases the creation of REST APIs since it translates Java objects into JSON object in the responses sent by the methods.

In order to map the received request URL Spring offers the annotations @RequestMapping, @PostMapping, @GetMapping, @PutMapping and @DeleteMapping, to which it must be added the pattern that the URL must follow. The first one designates that the requests matching the pattern must be attended by the annotated class while the rest are employed at method level to signal what type of petition that method is going to be attending, them being POST, GET, PUT or DELETE respectively.

Controllers, just like services, make use of the DTOs (Data Transfer Object) when the information exchanged among client and server cannot be represented by any of the entities defined on the data model.

Client

The client follows a package structure that can be seeing in figure 5.16. For each package we may highlight:

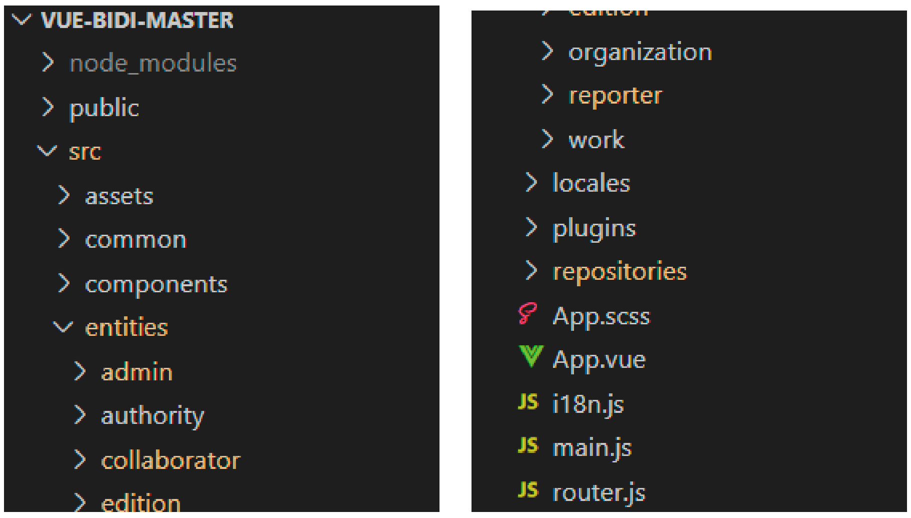


Figure 5.16: Package structure of the application client

- **node_modules:** contains all the npm libraries referenced in the application.

- **public**: the public folder from the application containing the index.html file.
- **src**: contains the application code, the internal hierarchy contains the following packages:
  - **assets**: application images for those general cases applicable to any generated product.
  - **components**: contains the main components of the application, independent of the entity or authenticated user.
  - **entities**: contains one folder for each of the applications entities, them being the Admin visualization, Authority, Collaborator, Edition, Organization, Reporter and Work.
  - **locales**: contains the translated messages needed to apply the internationalization of the application.
  - **plugins**: configutarion of Vuetify.
  - **repositories**: contains the files in charge of the connection between the client and the server, one per resource on the server side.

Components

The web interface is composed by a set of components, reusable Vue intances. At the time of implementation the strategy Single File Components was chosen, meaning that each component was implemented in a single file with .vue extension, formed by three differentiated sections: template, script and style.

The first one, template, defines the view of the component employing a syntax based on HTML that allows the linking of the DOM to the data from the Vue instance. This template system eases the management of several events as well as the data representation since it provides methods for looping, conditional rendering, etc.

On the second hand, script defines the component behaviour and data, implemented in JavaScript. Other components', application files or external libraries' imports take place here. In addition, script implements the methods called during the component's execution, like the ones responsible of sending the requests to the server and await the response.

Lastly, the styles of the elements that compose the user interface are defined, interface defined on the template section.

5.3 Implementation and testing

5.3.1 Implementation

This section covers the most complex implementation issues and other issues that complement the information in the previous sections.

Features implemented

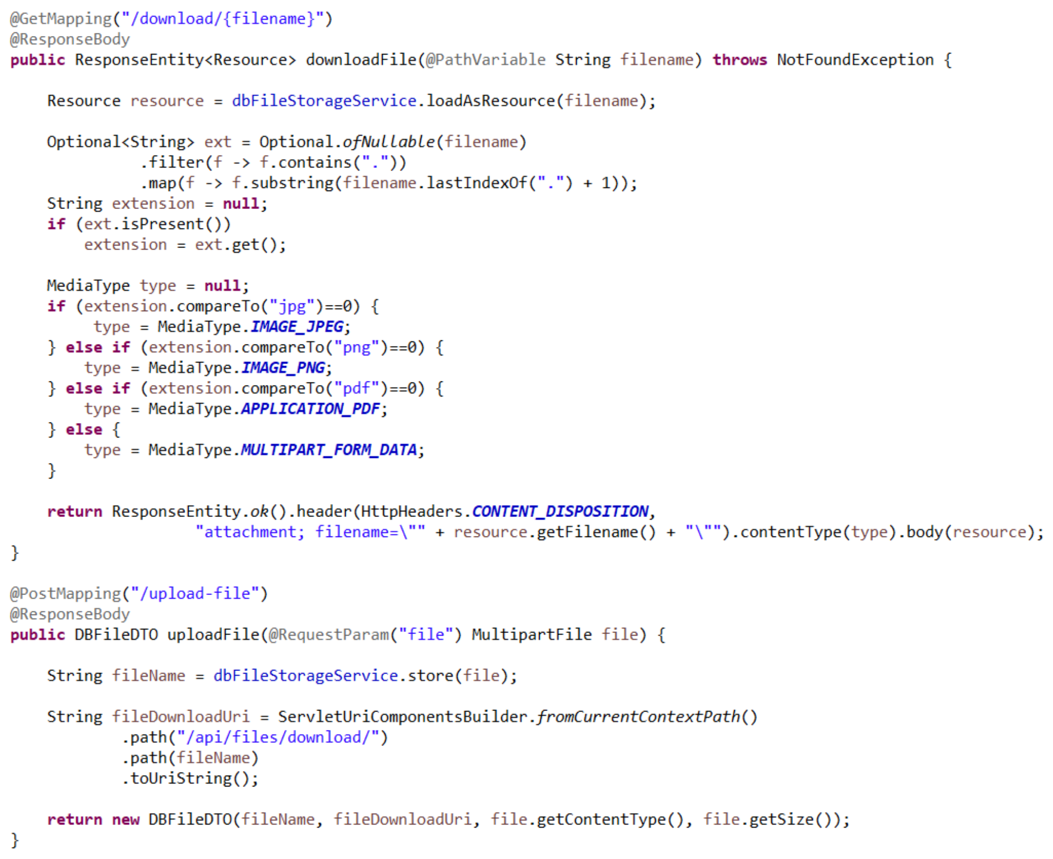
Just as it was stated at the beginning of this section, for the product based on BIDISO a set of features was selected and implemented. Specifically, X out of 149 features (see figure 4.2) were implemented. The features Authority and Language were implemented in its whole, including their subfeatures. The list of features implemented from the remaning can be found below:

- *UserManagement*: *UserProfiles*, *UserRoles* with all the subfeatures corresponding to *AdminFeatures*, *CollaboratorsFeatures* and *AnonymousFeatures*, *AnonymousUserCanRegisterInTheApplication* and *AdminUserNeedsToValidate*.
- *DigitalResource*: *FilesSupported*, *ExternalFiles* and *PDFFiles*.
- *Elements*: *SchedulingPublication*, *ElementTypes*, *WorElement*, *EditionElement*, *LocationManagement* and *HistocialLocation*
- *Organization*: *OrganizationRole* with all its subfeatures.
- *PublicAccess*: *SeachResult* with all its subfeatures, *SimpleSearch* with all its subfeatures and *Navigation* with all its subfeatures.
- *ManagementAccess*: *ManagementSearch* and *Filter* with all its subfeatures.

Storage of pictures and digital resources

One of the main features to be implemented was the storage of files of any kind, required to store not only the digital resources associated to each publication, but also the profile picture for each authority and the cover image for each publication. It supposed a challenge among the functionalities outside of the SPL implementation.

First of all, a new Java class had to be implemented outside of the data model in order to store the information of each file associated to the corresponding entity, which would keep the information gathered from the file such as fileName, fileType and the fileDownloadUri, in addition to the size. To improve the performance of the application it was decided to store the files on the hard disk instead of the database, since keeping an array of bytes in the DDBB would involve the processing of each file everytime we want to retrieve them. Taking into account that one of the most frequent uses of the product will be the listing of the cover images and profile pictures, is would be really resource consuming to do so each time for a massive amount of data.Moreover, a resource dedicated exclusively to file management was



```java
@GetMapping("/download/{filename}")
@ResponseBody
public ResponseEntity<Resource> downloadFile(@PathVariable String filename) throws NotFoundException {

    Resource resource = dbFileStorageService.loadAsResource(filename);

    Optional<String> ext = Optional.ofNullable(filename)
             .filter(f -> f.contains("."))
             .map(f -> f.substring(filename.lastIndexOf(".") + 1));
    String extension = null;
    if (ext.isPresent())
        extension = ext.get();

    MediaType type = null;
    if (extension.compareTo("jpg")==0) {
         type = MediaType.IMAGE_JPEG;
    } else if (extension.compareTo("png")==0) {
        type = MediaType.IMAGE_PNG;
    } else if (extension.compareTo("pdf")==0) {
        type = MediaType.APPLICATION_PDF;
    } else {
        type = MediaType.MULTIPART_FORM_DATA;
    }

    return ResponseEntity.ok().header(HttpHeaders.CONTENT_DISPOSITION,
                "attachment; filename=\"" + resource.getFilename() + "\"").contentType(type).body(resource);
}

@PostMapping("/upload-file")
@ResponseBody
public DBFileDTO uploadFile(@RequestParam("file") MultipartFile file) {

    String fileName = dbFileStorageService.store(file);

    String fileDownloadUri = ServletUriComponentsBuilder.fromCurrentContextPath()
            .path("/api/files/download/")
            .path(fileName)
            .toUriString();

    return new DBFileDTO(fileName, fileDownloadUri, file.getContentType(), file.getSize());
}
```

Figure 5.17: Extract from the Resource dedicated to files

implemented with the URI */api/files*, whose most requested methods will be those shown in figure 5.17. The *storageLocation* had to be added to the configuration properties file for the application, with the path to which the uploaded files must be directed.

In order to request the upload of a file, the HTTP request was implemented using **axios** [40], in charge of its configuration. The file are passed as MultiPart files, which are established as the standard of Spring according to its guide and comes auto-configured, saving up a lot of implementation. In the case of the cover of an edition, like in the snippet below, the petition is executed with **axios** and it receives the response in the type *DBFile* defined in the server, so that we can directly store that object as a reference from edition.

```javascript
if (this.edition.cover) {
          const formData = new FormData();
          formData.append("file", this.edition.cover);
          const res = await HTTP.post(
            "http://localhost:8080/api/files/upload-file",
            formData,
            {
              headers: {
                "Content-Type": "multipart/form-data"
              }
            }
          );
          this.edition.cover = res.data;
        }
```

Listing 5.1: Client petition for upload-file

For the download of files, the file extension had to be extracted from the returned object by parsing its name in order to adjust the media type of each of the file types supported by the application generated, in this case PDF, PNG and JPG.

5.3.2 Testing

In this section we comment on the type of tests that have been carried out to ensure quality and compliance with customer requirements.

The tests that have been carried out in this project are known as functional tests, which are specific, concrete and exhaustive tests to prove and validate that the software meets the specified requirements. There are several types of functional tests, but those that have been performed in this project are known as acceptance software quality tests.

During these tests, people (often from the demographic area the software is designed for) prove the software to make sure it can handle required tasks in real-world scenarios, according to specifications. Acceptance tests let the developers know if an application complies with the expected performance and they focus on the users who are in charge of validating the functionality and performance of the application.

These evaluations have been performed throughout the entire development process, first by the development team, checking the correct functioning of the new functionalities, added to the project during the implementation of each of the iterations, and then, in each of the sprint review meetings, by checking if the new functionalities fulfilled the requirements and if the interface was adequate, intuitive and easy to use.

The requested changes as a result of these sprint review were:

- Minor changes on the design of the data model, retiring irrelevant information from user profiles and instead linking an external URL to collaborator's profile since they are expected to be associated to an organization that will keep that data for us.
- Add the possibility to export the bibliographic reference to the feature model.

- Add pagination choices to the feature model, giving the option to have infinite scroll or numerated pages.
- Change the interface to have different menu bars for both administration and public access. For the administration it will be a side bar, for the public access it will be a top bar with less items.

Chapter 6

Software product line construction

6.1 Analysis

In this section, we explain the analysis of the software product line that were set out in chapter 4, describing the system architecture, the user interface, the design and its implementation and testing.

6.1.1 System architecture

This section will describe the system architecture for the SPL with a high level of abstraction by performing a high-level decomposition of the system components but without yet mentioning the technologies that support them.

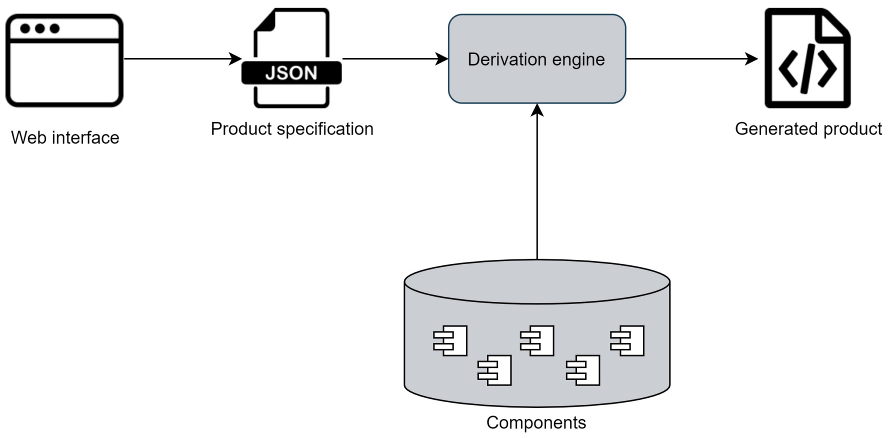


Figure 6.1: System architecture for the Software Product Line

Figure 6.1 provides an overview of how the system architecture is structured.

Firstly, the **web interface** is the component displaying the functionalities offered by the generation tool to the final user. Using the interface the user can select the desired features for the product, can import or export a file with the product specification, can add a title to the product and, last but not least, generate a digital library.

The **components** are reusable elements implementing the functionalities and features defined for the product.

Lastly, the **derivation engine** is the element responsible for the code generation of the product using the components previously implemented and taking the defined specification as a basis for the product, the desired features.

6.1.2 User interface

In this section, the user interface will be described at a high level by discussing the general structure of the application's screens and the navigation structure among them. The interface designed for the generation tool pursued a simple and easy to understand design, very intuitive since the focus for this tool is on the functionality, not the visual appeal.

The main page for the web generator must be a simple interface that allows the user to perform the basic operations such as import or export the specification and export the source code of the product, with a button to call each of these functionalities. The menu bar will be displayed at the top to allow navigation towards the feature selection page. In this case, also, simplicity rules and the page is conformed by a hierarchical list of all the features modelled in figure 4.2 with a unique checkbox besides the name to allow its selection. In case the feature *StaticPagesManagement* is selected, a new button unveils in the menu to access the Static Pages list. The list will be empty in case there is no existing specification that includes static pages loaded, and will allow the creation of a new static page. For that purpose it will be implemented a page with a simple text editor that allows plain HTML input.

6.2 Design

6.2.1 Technological architecture of the system

In this section the goal is to supplement the architecture analysis executed in section 6.1.1, giving a insight of the technologies employed for each component of the SPL. Its architecture is depicted in figure 6.2.

Derivation engine

As it was stated earlier, the final choice for the generation of source code for the products from the SPL was spl-js-engine, a JavaScript derivation engine based on scaffolding. This tool makes use of annotations over the original code of the product to define the snippets belonging to each component, instead of physically splitting the code. This implies an advantage, as it was noted at section 2.1. The template annotations are indicated by comments in the

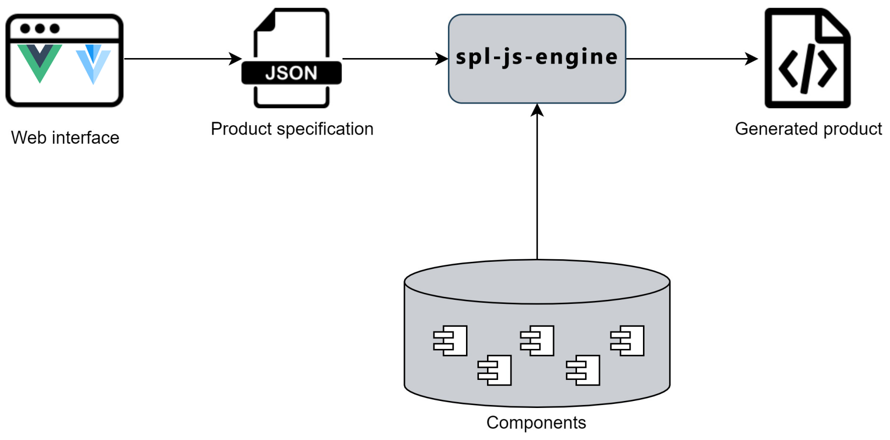


Figure 6.2: Technological architecture of the Software Product Line

programming language of the template and its content is any JavaScript code. Along with to annotations related to features, the template engine allows you to use variables in templates. Finally, the engine can also validate the specification and templates with functionalities like detecting which characteristics are not referenced in any annotation [34].

The derivation engine needs determined input files in order to generate the product. The configuration files for the tool, which will be model.xml, config.json and extra.js.

The file **model.xml** contains the feature model for the SPL, and can be easily generated using the Eclipse plugin FeatureIDE [10], which allows the generation of a XML file from the graphical representation of the feature tree (collapsed version in figure 4.2). The source code for model.xml can be seen at listing A from appendix A. The file **config.json** contains the derivation engine configuration, defining the delimiters for annotation depending of the template file extension, and those elements that must be ignored at generation time. The file **extra.js** contains the JavaScript code utilized to derivate the product.

Additionally, the file **product.json** contains the product specification in JSON format.

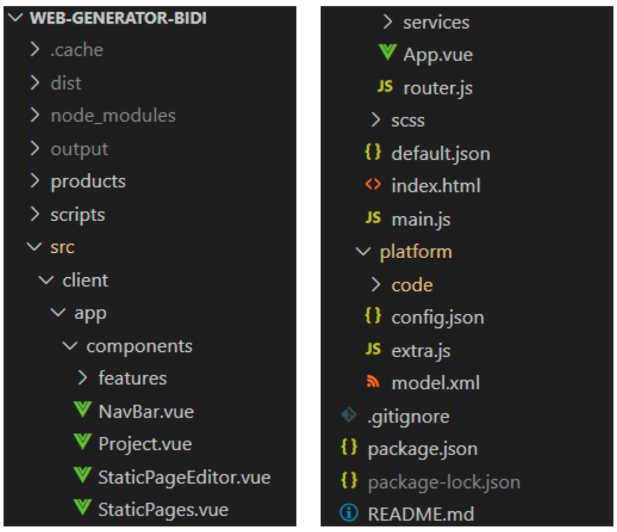
Lastly, the **templates** embrace the source code developed for the product (explained in chapter 5) with the corresponding annotations.

Web interface

The implementation of the web interface was carried out following the steps of the product, with Vue.js and Vuetify, making use of the library vue-router. These technologies were throughly explained in section 5.2.2, so the details will be spared to avoid redundancy.

6.2.2 Application design

This section sets out the internal structure of the web application designed for the generation tool. Figure 6.3 displays the package structure established for said tool. Just like in the case for the product construction, **node_modules** contains the npm libraries referenced, **scripts** contains auxiliary JavaScript files, **products** stores the *product.json* file that defines the specification and **output** will be the containing folder for the generated product after the generation process.



```
∨ WEB-GENERATOR-BIDI
  > .cache
  > dist
  > node_modules
  > output
  > products
  > scripts
  ∨ src
    ∨ client
      ∨ app
        ∨ components
          > features
          NavBar.vue
          Project.vue
          StaticPageEditor.vue
          StaticPages.vue
        > services
        App.vue
        router.js
      > scss
      default.json
      index.html
      main.js
    ∨ platform
      > code
      config.json
      extra.js
      model.xml
  .gitignore
  package.json
  package-lock.json
  README.md
```

Figure 6.3: Generation tool package structure

Moreover, the **src** folder contains the application code and the elements required to generate the products. The internal hierarchy goes as follow:

- **client**, containing the web application code. It hosts the interface configuration files and the *index.html* file, along with a folder to store the styling files. The main components of the application are stored in the **app** folder, that contains the router file and the common aspects of the web, in addition to the following folders:
  - **components**, which contains the pages and components that conform the web interface. Its structure follows the same *Single Page Applications* premise just like the client designed for the product.
  - **services**, containing the JavaScript files in charge of implementing the features and behaviour of the application in combination with the derivation engine.
- **platform**, containing the templates to use as a basis for the generated product. It houses the configuration files detailed earlier in section 6.2.1 and a *code* folder simultaneously housing the code for both the client and the server of the product.

It is ought to be mentioned that, since the generation tool is based on *spl-js-engine*, which was developed by the directors of this thesis, part of the interface was provided together with the engine. Components such as the *FeatureTree* display and selection and some of the JavaScript features to join the behaviour of the interface together with the engine were already implemented and only some modifications were applied by the author of thesis.

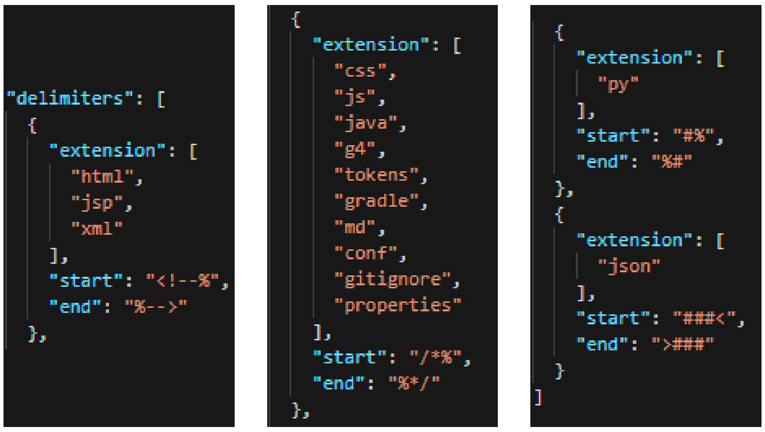
6.3 Implementation and testing

6.3.1 Implementation

In this section, we describe some complex features that implements an important functionality of the application.

Annotations

Based on the previous explanation of the tool, it can be stated that annotations are the key point of the Software Product Line implementation. When it came to annotating the source code it was required to, first of all, define the delimiter comments for each type of template in the *config.json* file. Said configuration can be seen in figure 6.4.



```
"delimiters": [
  {
    "extension": [
      "html",
      "jsp",
      "xml"
    ],
    "start": "<!--%",
    "end": "%-->"
  },
```

```
  {
    "extension": [
      "css",
      "js",
      "java",
      "g4",
      "tokens",
      "gradle",
      "md",
      "conf",
      "gitignore",
      "properties"
    ],
    "start": "/*%",
    "end": "%*/"
  },
```

```
  {
    "extension": [
      "py"
    ],
    "start": "#%",
    "end": "%#"
  },
  {
    "extension": [
      "json"
    ],
    "start": "###<",
    "end": ">###"
  }
]
```

Figure 6.4: Delimiter annotations configuration

The configuration offers the possibility to establish different delimiters for each file type, enabling the integration of the annotation as code comments so that they don't impact on code compilation. Nevertheless, .vue files do not support an universal type of comments valid for each part of the file (*template* in HTML, *script* in JavaScript and *style* in CSS) since the format of comments is different for each of them. Moreover, JSON files do not support comments, son compilation error may arise when compiling.

Once the configuration is settled, the source code annotation can proceed for both the server and the client of the developed product. Considering that the derivation engine takes care of checking dependencies and constraints, it was not necessary to implement them during the annotation.

A small code snippet can be seen in listing 6.1, only to enable the visualization of how these annotations actually work, inserted in the main code and following the delimiter configuration stated above. The annotation process entails wrapping up a slice of code that must be added just in case the feature associated to that piece was selected in the *FeatureTree*. The developer in charge of the annotation must take care of placing the annotation in the correct places, since a comma generated when it should not be there could derive into a compilation error.

```
@Transactional(readOnly = false)
    @PreAuthorize("hasAuthority('ADMIN')")
    public void registerCollaborator(String email, String login, String password/*%  if (feature.UserProfile) { %*/, String name, String surname, String organization, String occupation, String externalURL/*% } %*/ ) throws UserLoginExistsException {
     if(collaboratorDAO.findByEmail(email) !=null) {
       throw new UserLoginExistsException("User email " + email + " already exists");
     }
     if(collaboratorDAO.findByLogin(login) != null) {
       throw new UserLoginExistsException("User login " + login + " is taken");
     }
     Collaborator collaborator = new Collaborator();
     String encryptedPassword = passwordEncoder.encode(password);
     collaborator.setEmail(email);
     collaborator.setLogin(login);
     collaborator.setPassword(encryptedPassword);
     /*% if (feature.UserProfile) { %*/
     collaborator.setName(name);
     collaborator.setSurname(surname);
     collaborator.setOrganization(organization);
     collaborator.setOccupation(occupation);
     collaborator.setExternalURL(externalURL);
     /*% } %*/
     collaboratorDAO.create(collaborator);
    }
```

Listing 6.1: Code snippet from the REST service with annotations

Static pages

The main peculiarity this product has to offer is the possibility of customization even when the premise is that the generated products must be limited to a set of given functionalities. This can be done thanks to the management of static pages, which the final user can design to they liking, and include in the generation tool as a part of the final product. If the feature *StaticPagesManagement* is selected on the FeatureTree, the user can access the *StaticPages* listing and create a new page.

The creation of static pages is implemented using CKEditor, a JavaScript rich text editor with MVC architecture, custom data model and virtual DOM that provides every type of WYSIWYG editing solution imaginable [41, 42]. After creating the page, its raw HTML code gets stored in a list that later on can be exported as part of the specification together with the title of the digital library.

When it comes to the inclusion of the static pages to the generated product, a new annotation had to be implemented in order to create full files from scratch. In this case, the file generation annotation must be at the beginning of the file and must contain the implementation of a function that perceives both the data (title and pages) and the features from the JSON specification file and returns an array of objects with the properties corresponding to the file name and the content of the file. The @ character is used to differentiate the file generation annotation from the rest. The generated files get stored in a dedicated folder and they are accessible from a drop-down menu located on the menu bar of the resulting web application where a new tab gets generated for each page.

6.3.2 Testing

The correct implementation of the Software Product Line was tested through diverse manual tests. With the main objective of checking the correctness of all annotations included on the templates, several products were generated trying all of them. Due to the sizeable amount of elements on the feature model designed, and taking into account that many of the functionalities were not implemented because of that same reason, it was not feasible to test all the possible combinations.

Besides the generation engine, the web interface for the tool was also tested by checking creation, exportation and importation of JSON and ZIP files containing the specification file and the code for the generated application respectively. They use cases any regular user can perform were carried out by the developer team.

Chapter 7

Developed solution

The aim of this chapter is to show the developed application resulting from this project. To this end, a explanation will be provided on the main functionalities of the application.

7.1 Generation tool

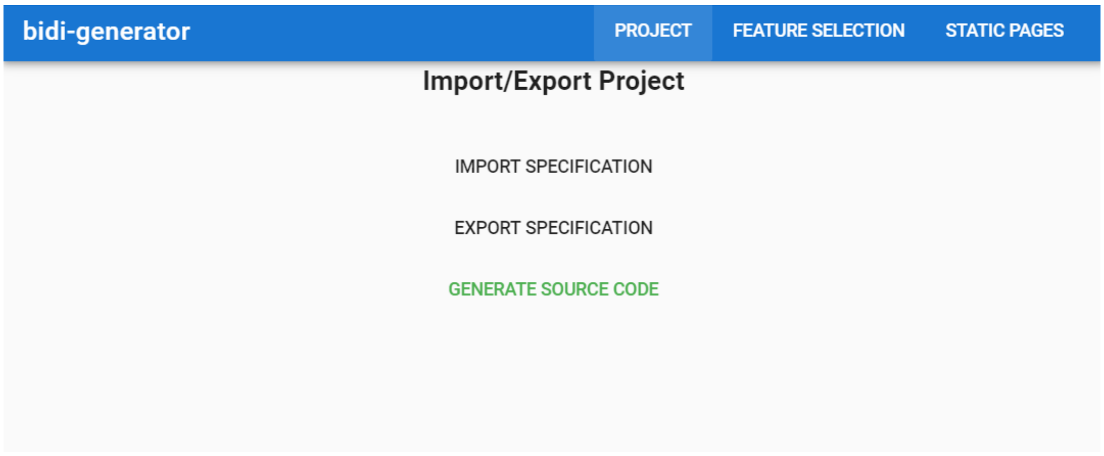
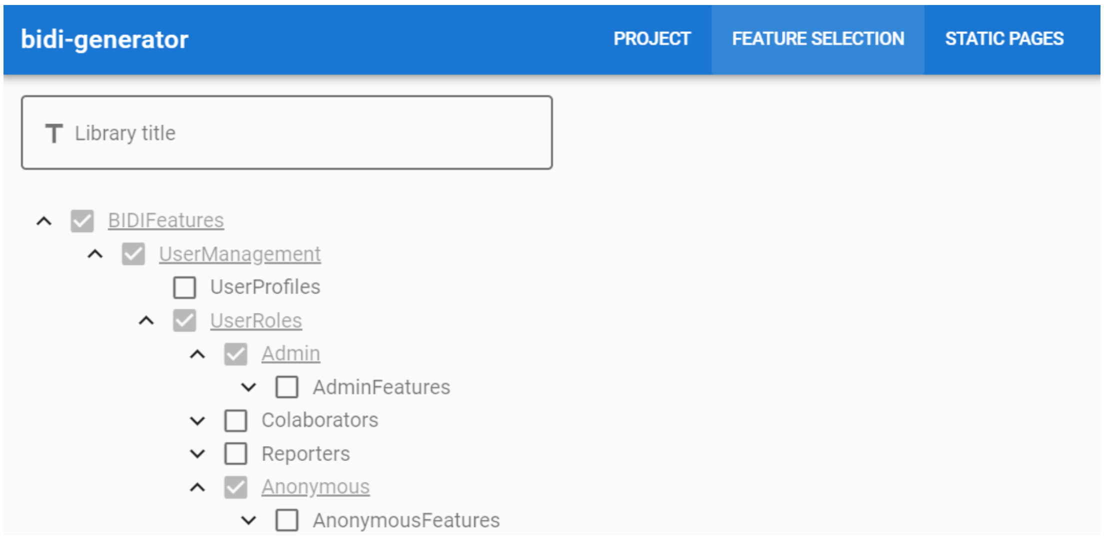


Figure 7.1: Main page of the generation tool

This section illustrates the final behaviour of the generation tool. The main page of the application can be seen in figure 7.1, providing easy access to the key functionalities. When the user opens the generation tool, they can, first of all, import an already existing specification that could have been previously exported in the past, this specification will be shown in the feature selection and the list of static pages. The selection of the features can be done in the *FeatureTree* page, shown in figure 7.2. If a constraint is not fulfilled, a red alert will display at the top of the page informing the user what needs to be corrected. This alert also checks the conditions based on the definition of *OR* and *Alternative* features, avoiding any possible human error jeopardizing the generation of the product.

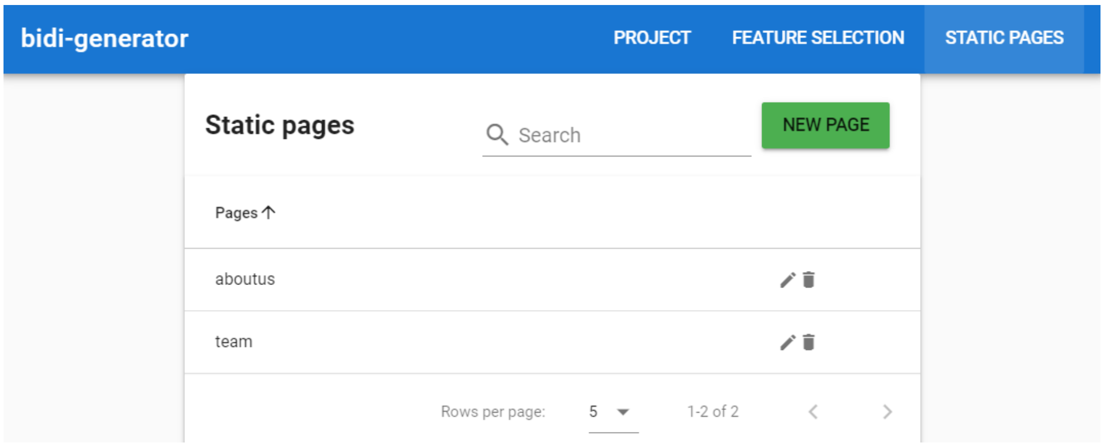


- [✓] BIDIFeatures
  - [✓] UserManagement
    - [ ] UserProfiles
    - [✓] UserRoles
      - [✓] Admin
        - [ ] AdminFeatures
      - [ ] Colaborators
      - [ ] Reporters
      - [✓] Anonymous
        - [ ] AnonymousFeatures

Figure 7.2: Feature tree selection of the generation tool

Lastly, the *StaticPages* listing is represented in figure 7.3. When the user wants to create a New Page, the editor based on CKEditor will be displayed as the center of the page with a field to define the title of the page before saving. The pages can be edited or deleted before generating the product.

The generation of the final product is automatically done when clicking the "Generate Source Code" button from the main page. After clicking the button, a file saver window will pop up, allowing the user to select where to store the zip file with the application, both server and client. Its deployment and execution is really easy and a instructions file can be found inside the zip file so that no technical knowledge should be needed.



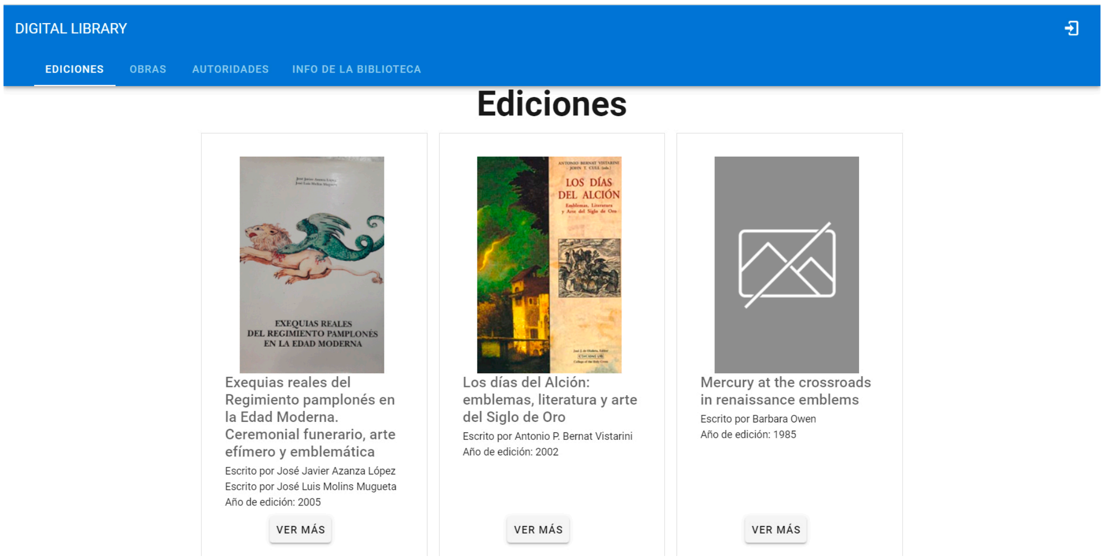
Pages ↑
aboutus
team

Figure 7.3: Static pages management of the generation tool

7.2 Product

This section illustrates the final behaviour of a generated example with the digital library generation tool.

The welcome page for the blog is a part of the variability but in this example it was chosen to show the list of editions published on the digital library. Additionally note that the example is generated in spanish since the product was designed as a prototype for the library BIDISO [1], centered in hispanic literature and part of ARACNE [43], the Digital Humanities and Hispanic Letters Networks. The welcome page can be seen in figure 7.4, showing the cover of the publications when possible, and a default image in case the cover is not available.



Exequias reales del Regimiento pamplonés en la Edad Moderna. Ceremonial funerario, arte efímero y emblemática
Escrito por José Javier Azanza López
Escrito por José Luis Molins Mugueta
Año de edición: 2005

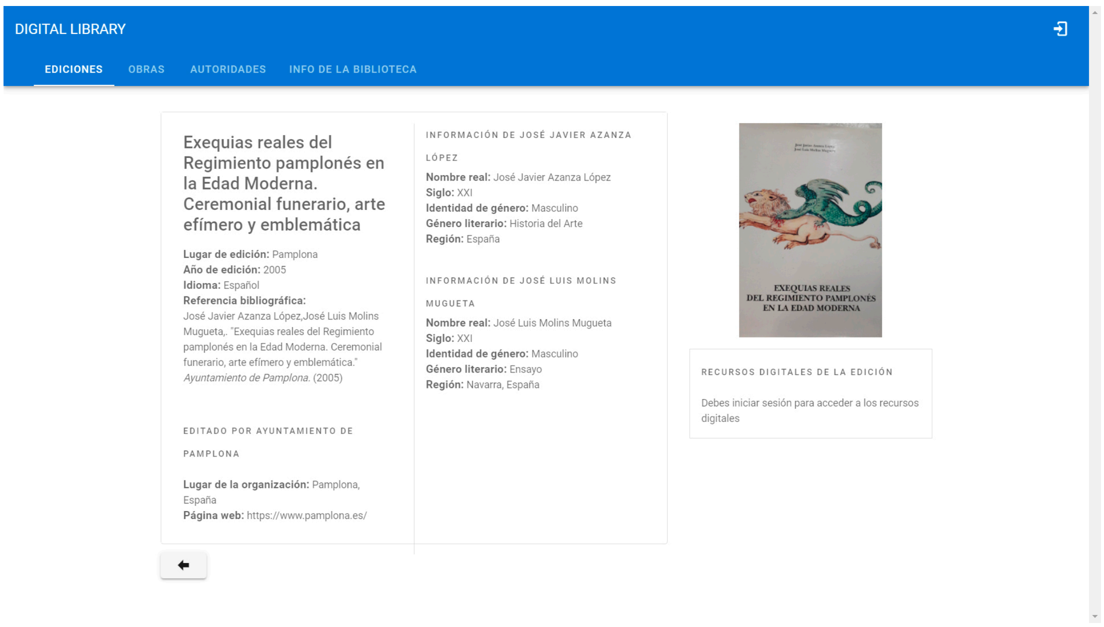
Los días del Alción: emblemas, literatura y arte del Siglo de Oro
Escrito por Antonio P. Bernat Vistarini
Año de edición: 2002

Mercury at the crossroads in renaissance emblems
Escrito por Barbara Owen
Año de edición: 1985

Figure 7.4: List of published editions from the generated product

The horizontal menu at the top bar is provided to navigate the web, allowing the navigation to the list of works, the list of authorities and a drop-down selector to navigate the possible created static pages. In this case, due to the similarity of the interface for both the works list and the edition list, the screenshot will be spared to save up space. When clicking the "*View More*" button below each element, the user gets redirected to the detail page for the selected edition, seen in figure 7.5. In this page the information available about an edition is displayed in a straight-forward manner, trying to get everything in just one sitting. In this case, the digital resources are not listed because the user is not authenticated into the application.

After login into the application with the button at the top right corner, the user can access the files related to a publication once their credentials are validated. The page for the edition details is the same, the only change is that now there is no prohibition message and, instead,


DIGITAL LIBRARY

EDICIONES OBRAS AUTORIDADES INFO DE LA BIBLIOTECA

Exequias reales del Regimiento pamplonés en la Edad Moderna. Ceremonial funerario, arte efímero y emblemática

Lugar de edición: Pamplona
Año de edición: 2005
Idioma: Español
Referencia bibliográfica:
José Javier Azanza López,José Luis Molins Mugueta,. "Exequias reales del Regimiento pamplonés en la Edad Moderna. Ceremonial funerario, arte efímero y emblemática." *Ayuntamiento de Pamplona.* (2005)

EDITADO POR AYUNTAMIENTO DE PAMPLONA

Lugar de la organización: Pamplona, España
Página web: https://www.pamplona.es/

INFORMACIÓN DE JOSÉ JAVIER AZANZA LÓPEZ

Nombre real: José Javier Azanza López
Siglo: XXI
Identidad de género: Masculino
Género literario: Historia del Arte
Región: España

INFORMACIÓN DE JOSÉ LUIS MOLINS MUGUETA

Nombre real: José Luis Molins Mugueta
Siglo: XXI
Identidad de género: Masculino
Género literario: Ensayo
Región: Navarra, España

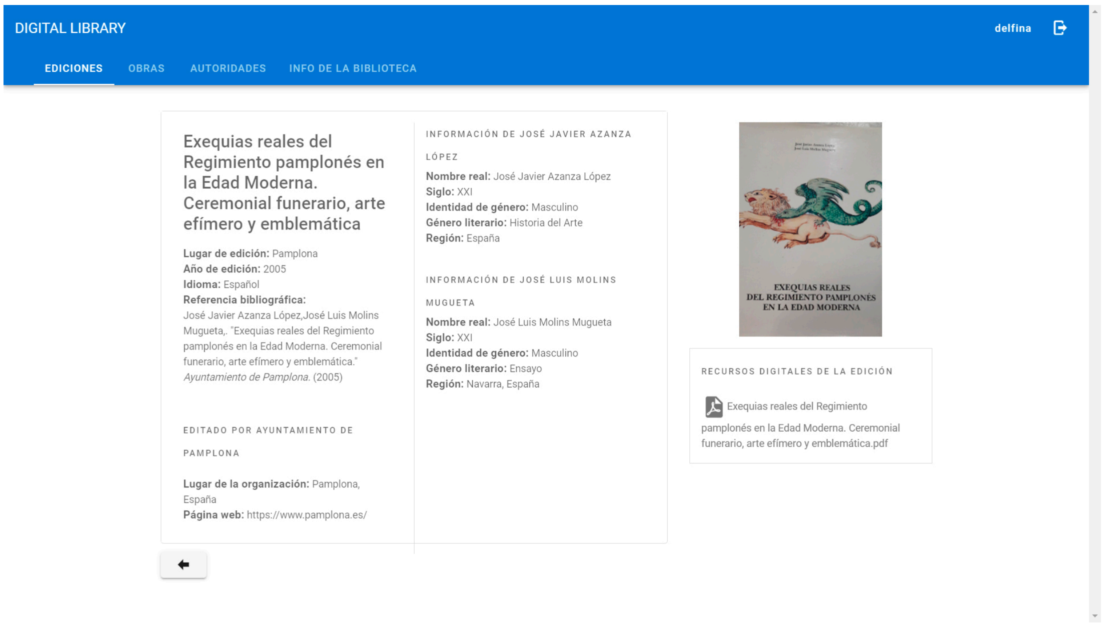
EXEQUIAS REALES DEL REGIMIENTO PAMPLONÉS EN LA EDAD MODERNA

RECURSOS DIGITALES DE LA EDICIÓN

Debes iniciar sesión para acceder a los recursos digitales

Figure 7.5: Detail of an edition without authentication

the icon and link to the file are listed below the cover. If the user clicks the link, the PDF file gets automatically downloaded.


DIGITAL LIBRARY

delfina

EDICIONES OBRAS AUTORIDADES INFO DE LA BIBLIOTECA

Exequias reales del Regimiento pamplonés en la Edad Moderna. Ceremonial funerario, arte efímero y emblemática

Lugar de edición: Pamplona
Año de edición: 2005
Idioma: Español
Referencia bibliográfica:
José Javier Azanza López,José Luis Molins Mugueta,. "Exequias reales del Regimiento pamplonés en la Edad Moderna. Ceremonial funerario, arte efímero y emblemática." *Ayuntamiento de Pamplona.* (2005)

EDITADO POR AYUNTAMIENTO DE PAMPLONA

Lugar de la organización: Pamplona, España
Página web: https://www.pamplona.es/

INFORMACIÓN DE JOSÉ JAVIER AZANZA LÓPEZ

Nombre real: José Javier Azanza López
Siglo: XXI
Identidad de género: Masculino
Género literario: Historia del Arte
Región: España

INFORMACIÓN DE JOSÉ LUIS MOLINS MUGUETA

Nombre real: José Luis Molins Mugueta
Siglo: XXI
Identidad de género: Masculino
Género literario: Ensayo
Región: Navarra, España

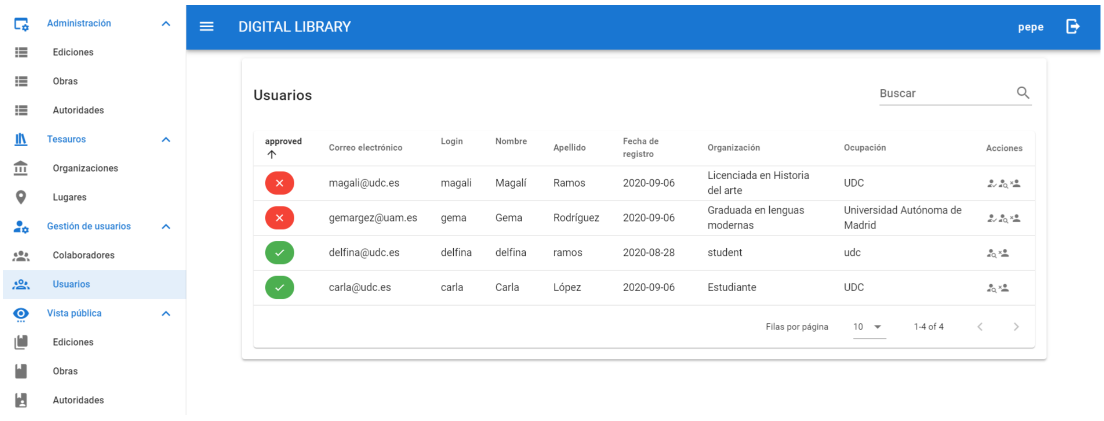
EXEQUIAS REALES DEL REGIMIENTO PAMPLONÉS EN LA EDAD MODERNA

RECURSOS DIGITALES DE LA EDICIÓN

Exequias reales del Regimiento pamplonés en la Edad Moderna. Ceremonial funerario, arte efímero y emblemática.pdf

Figure 7.6: Detail of an edition after authentication

To achieve this, the application verifies that a reporter has been previously approved by an admin since the *AdminUserNeedsToValidate* feature was selected during the generation of



Usuarios

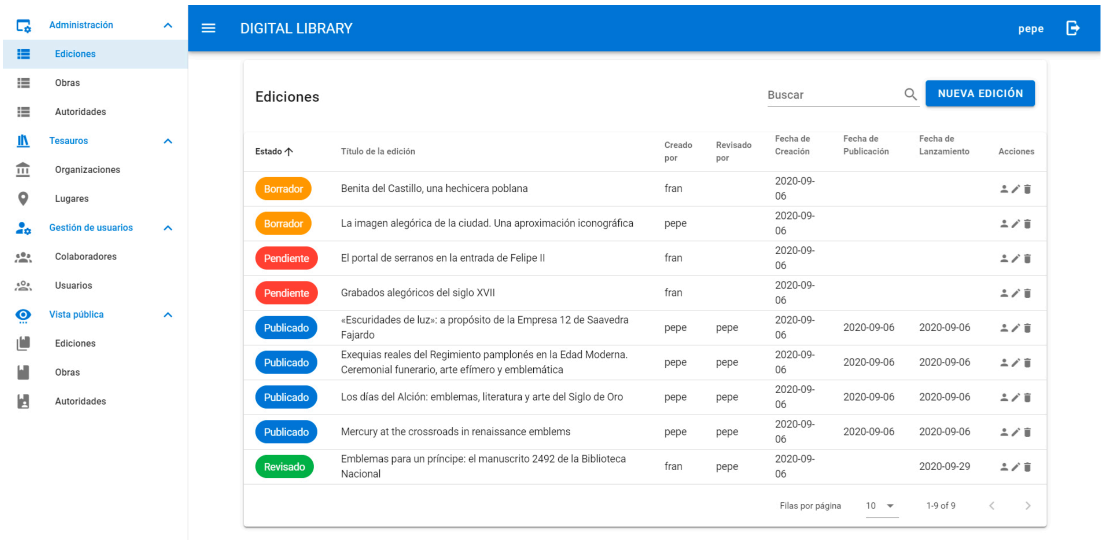
approved ↑	Correo electrónico	Login	Nombre	Apellido	Fecha de registro	Organización	Ocupación	Acciones
✗	magali@udc.es	magali	Magalí	Ramos	2020-09-06	Licenciada en Historia del arte	UDC	
✗	gemargez@uam.es	gema	Gema	Rodríguez	2020-09-06	Graduada en lenguas modernas	Universidad Autónoma de Madrid	
✓	delfina@udc.es	delfina	delfina	ramos	2020-08-28	student	udc	
✓	carla@udc.es	carla	Carla	López	2020-09-06	Estudiante	UDC	

Filas por página 10 1-4 of 4

Figure 7.7: Management of reporters

this product. When an administrator access the management site, they can view a list of registered reporters, signaled in red if they are waiting for approval or in green in case they have already approved. Changing their status is as easy as clicking the check button at the end of the row corresponding to the reporter. On the contrary, if we deny access to that reporter, the account gets deleted.

Note that the structure of the site is different from the public access, the menu bar is now located at the side instead of at the bottom, in addition to the fact that the management team has access to more information, such as the thesaurus for Organizations and Locations. The main page for the management site is also the edition list, as seen in figure 7.8, since the most common task will be to review editions created by other teammates and input new publications.



Ediciones

Estado ↑	Título de la edición	Creado por	Revisado por	Fecha de Creación	Fecha de Publicación	Fecha de Lanzamiento	Acciones
Borrador	Benita del Castillo, una hechicera poblana	fran		2020-09-06			
Borrador	La imagen alegórica de la ciudad. Una aproximación iconográfica	pepe		2020-09-06			
Pendiente	El portal de serranos en la entrada de Felipe II	fran		2020-09-06			
Pendiente	Grabados alegóricos del siglo XVII	fran		2020-09-06			
Publicado	«Escuridades de luz»: a propósito de la Empresa 12 de Saavedra Fajardo	pepe	pepe	2020-09-06	2020-09-06	2020-09-06	
Publicado	Exequias reales del Regimiento pamplonés en la Edad Moderna. Ceremonial funerario, arte efímero y emblemática	pepe	pepe	2020-09-06	2020-09-06	2020-09-06	
Publicado	Los días del Alción: emblemas, literatura y arte del Siglo de Oro	pepe	pepe	2020-09-06	2020-09-06	2020-09-06	
Publicado	Mercury at the crossroads in renaissance emblems	pepe	pepe	2020-09-06	2020-09-06	2020-09-06	
Revisado	Emblemas para un príncipe: el manuscrito 2492 de la Biblioteca Nacional	fran	pepe	2020-09-06		2020-09-29	

Filas por página 10 1-9 of 9

Figure 7.8: Management of editions

In this page, the editions are listed with the most important information from the administration point of view, such as who and when created each element. The focus of attention must be in the status, highlighting those elements that are pending of revision so that they

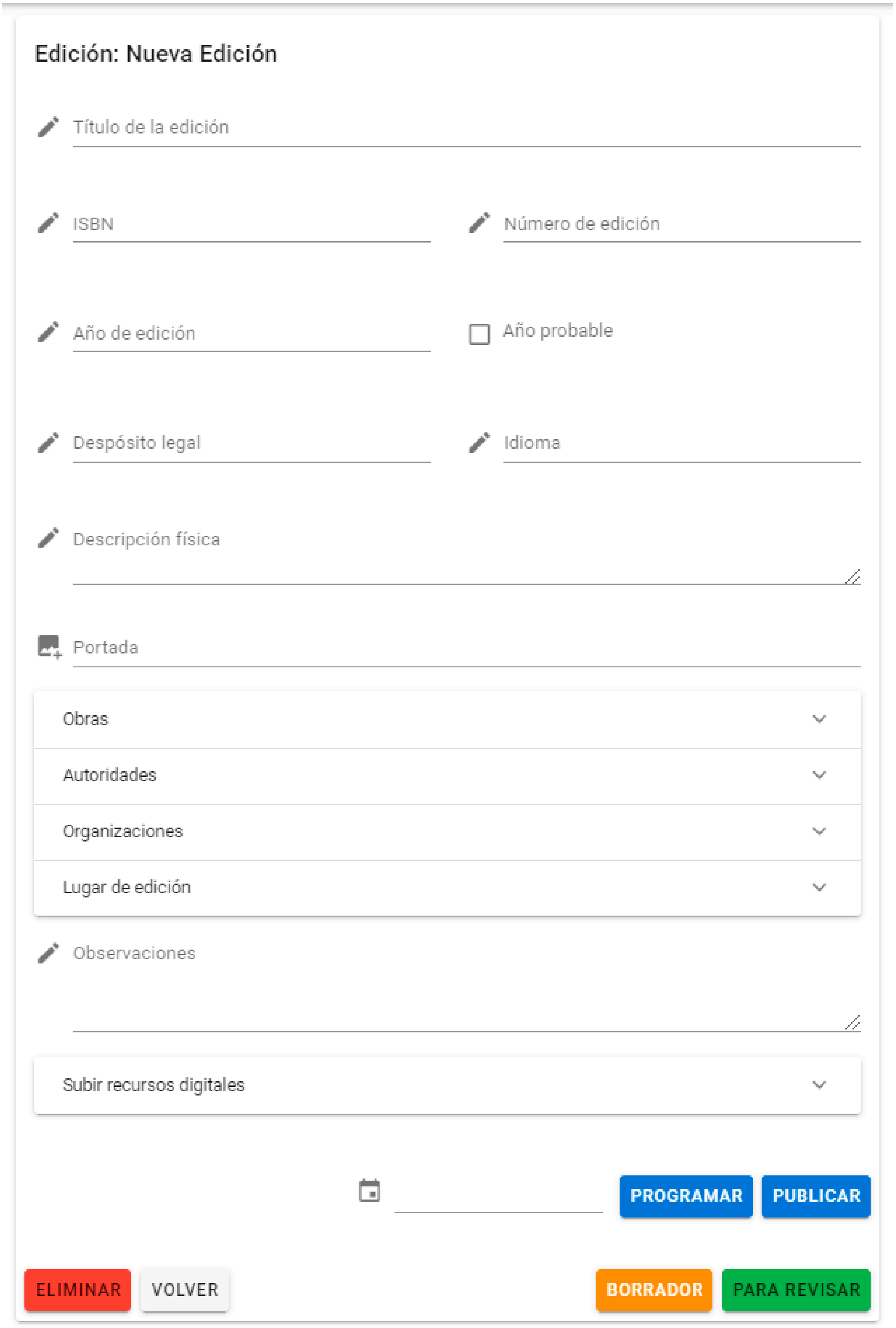


Figure 7.9: Form to create new editions or update existing ones

don't go unnoticed.

When an administration or collaborator wants to create a new edition, the only thing they have to do is to click the "*New Edition*" button at the top of the table and the creation page will display as seen in figure 7.9 (cut to allow the visualization of the full element).

As it was stated in the requirements, the user can create an edition and all of the elements related to it from the same workspace, allowing the possibility to publish at the same moment, schedule the publication, save as draft or mark it as ready to be reviewed. If instead of creating an element, they want to update an existing element, the page would be the same with the variation that the input fields would already be fulfilled with the data gathered form the object.

The screenshots for the work publications will be spared to avoid redundancy due to space issues and its similarity to the edition implementation. Also, the majority of the management site follows a similar distribution and behaviour, adjusting to small derivations such as the requirements that Locations can only be modelled from an edition since they can only exist depending on said edition.

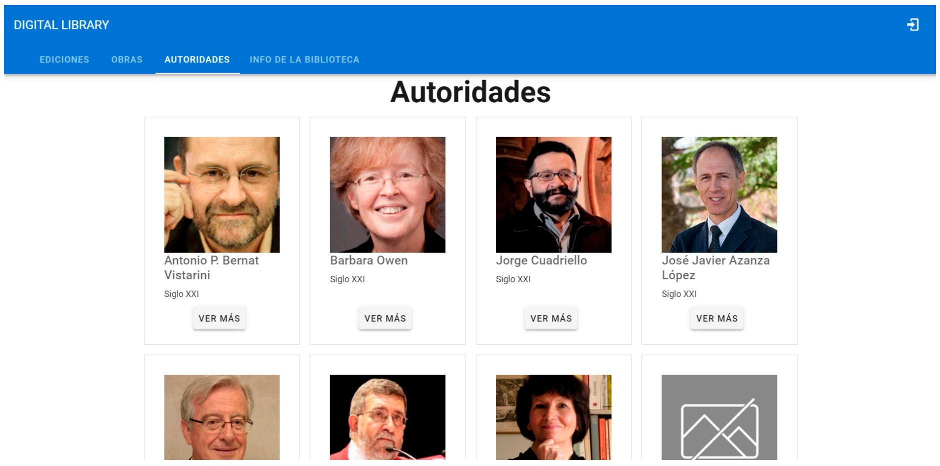


Figure 7.10: List of authorities from public access

Lastly, for the public access the authorities list follows a similar approach to the list of editions but displaying the pictures of the authors as shown in figure 7.10. Due to space issues the detail for an authority will not be shown because it follows the same approach as the detail for an edition, with a list of works and editions the person authored or collaborated.

Chapter 8

Conclusiones y trabajo futuro

This chapter discusses the degree to which the objectives set at the beginning of the project have been achieved, the lessons learned and future implementations that can enhance the functioning of the application.

8.1 Objectives

A tool based on Software Product Lines has been developed that allows the semi-automated generation of web applications to manage a digital library from a set of given features, implemented with a simple interface that facilitates the process of product generation.

The products generated are safe, high quality and easy to use through the execution of several tests, the implementation of a security system and the development of a user-friendly, intuitive interface.

The instructions necessary for the correct configuration and deployment of the generated product is provided to the final users, smoothing the set up process as much as possible.

8.2 Lessons learned

The execution of this thesis gave the author the opportunity to get to know and become aware of the planning and workload of a real project. As well as being able to apply a development methodology in a project, getting to know first-hand the importance of the good use of them, in order to ensure the quality of the final product.

Existing knowledge acquired along the degree about back-end technologies has been improved through their use during the project: Sprint, Hibernate, REST servers... At the same time new knowledge has been gathered related with the technologies and tools employed to develop this thesis, specially about Software Product Lines in combination with *scaffolding* and the work behind a generation tool.

8.3 Future work

As it was stated in section 5, this thesis was centered in the development of an specific product of the SPL that implements a subset features from all functionalities collected during the analysis phase, focusing instead on said analysis and the in-depth analysis of SPL. The future work shall continue with the implementation of those requirements defined but not fulfilled, such as the inclusion of maps into the application and the features related to them, the conversion of the data to make it exportable into different file types, the management of statistics from the management side...

In addition, some future improvements to make the application more suitable for the real world could be automatizing the deployment process for the generated products through the implementation of a cloud application platform, for example, so that the web apps could be accessible from anywhere as long as they have an internet connection.

On top of that, a peculiarity present on the domain of digital libraries is the different styles they all present with very customized interfaces. In upcoming versions of the software it should be considered the possibility of offering a degree of customization of the styles in a easy way, that does not require programming knowledge and that can keep track of the changes on each individual library when changing the version.

Appendices

Appendix A

Feature model for the SPL

Extended representation of the feature model with zoom-in for the set of features that can be more difficult to read.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<featureModel>
  <properties>
    <graphics key="legendautolayout" value="true"/>
    <graphics key="showshortnames" value="false"/>
    <graphics key="layout" value="vertical"/>
    <graphics key="showcollapsedconstraints" value="true"/>
    <graphics key="legendhidden" value="false"/>
    <graphics key="layoutalgorithm" value="4"/>
  </properties>
  <struct>
    <and abstract="true" mandatory="true" name="BIDIFeatures">
      <and mandatory="true" name="UserManagement">
        <graphics key="collapsed" value="true"/>
        <feature name="UserProfiles"/>
        <and abstract="true" mandatory="true" name="UserRoles">
          <and mandatory="true" name="Admin">
            <and abstract="true" name="AdminFeatures">
              <feature name="ReviewingFeatures"/>
              <feature name="StaticPagesManagement"/>
            </and>
          </and>
          <and name="Colaborators">
            <and abstract="true" name="ColaboratorsFeatures">
              <feature name="OnlyCreatorCanModify"/>
              <feature name="ThesauriCRUD"/>
            </and>
          </and>
          <and name="Reporters">
            <or abstract="true" name="ReportersFeatures">
            <feature name="FavoriteElements"/>
            <feature name="FavoriteAuthors"/>
          </or>
        </and>
        <and mandatory="true" name="Anonymous">
          <and abstract="true" name="AnonymousFeatures">
            <feature name="CanAccessElementFiles"/>
          </and>
        </and>
      </and>
      <and name="AnonymousUserCanRegisterInTheApplication">
        <feature name="AdminUserNeedsToValidate"/>
        <or abstract="true" name="SocialRegistration">
          <feature name="Facebook"/>
          <feature name="Twitter"/>
          <feature name="Google"/>
        </or>
      </and>
    </and>
    <and name="DigitalResource">
      <graphics key="collapsed" value="true"/>
      <or abstract="true" mandatory="true" name="FileTypesSupported">
        <feature name="ExternalFiles"/>
        <feature name="EPUBFiles"/>
        <feature name="VideoFiles"/>
        <feature name="AudioFiles"/>
        <feature name="PDFFiles"/>
        <feature name="DigitalizedPages"/>
      </or>
      <feature name="Collections"/>
    </and>
    <and mandatory="true" name="Element">
      <graphics key="collapsed" value="true"/>
      <feature name="SchedulingPublication"/>
      <and abstract="true" mandatory="true" name="ElementTypes">
        <feature mandatory="true" name="WorkElement"/>
        <and mandatory="true" name="EditionElement">
          <and name="LocationManagement">
            <feature name="HistoricalLocation"/>
            <feature name="GeographicalLocation"/>
          </and>
        </and>
        <feature name="PartElement"/>
        <feature name="CollectionElement"/>
      </and>
      </and>
      <and mandatory="true" name="Authority">
        <graphics key="collapsed" value="true"/>
        <feature name="Alias"/>
        <and name="Contacts">
          <feature name="SocialNetwork"/>
        </and>
        <or mandatory="true" name="AuthorityRole">
          <feature name="Author"/>
          <feature name="Editor"/>
          <feature name="Printer"/>
          <feature name="Illustrator"/>
          <feature name="Translator"/>
          <feature name="Prologuist"/>
        </or>
      </and>
      <and name="Organization">
        <graphics key="collapsed" value="true"/>
        <feature name="OrganizationDirector"/>
        <or mandatory="true" name="OrganizationRole">
          <feature name="Library"/>
          <feature name="Editorial"/>
        </or>
      </and>
      <or mandatory="true" name="Language">
        <graphics key="collapsed" value="true"/>
        <feature name="Spanish"/>
        <feature name="English"/>
      </or>
      <and mandatory="true" name="PublicAccess">
        <graphics key="collapsed" value="true"/>
        <and name="Search">
          <graphics key="collapsed" value="true"/>
          <and name="SearchResult">
            <and name="SearchResultEdition">
              <and name="SearchFormat">
                <feature name="SearchResultAsList"/>
                <feature name="SearchResultAsMap"/>
              </and>
            </and>
            <feature name="SearchResultWorks"/>
            <feature name="SearchResultAuthority"/>
          </and>
          <and name="AdvancedSearch">
            <feature name="SearchByTitle"/>
            <and name="SearchByAuthority">
```

```xml
              <feature name="SearchByAuthor"/>
              <feature name="SearchByEditor"/>
              <feature name="SearchByCustom"/>
            </and>
            <feature name="SearchByLocation"/>
            <feature name="SearchByLanguage"/>
            <and name="SearchByDate">
              <feature name="SearchByDateRange"/>
            </and>
            <feature name="SearchByIdentifier"/>
            <feature name="SearchByISBN"/>
          </and>
          <or name="SimpleSearch">
            <feature name="SimpleSearchByTitle"/>
            <feature name="SimpleSearchByAuthor"/>
            <feature name="SimpleSearchByYear"/>
            <feature name="SimpleSearchByLocation"/>
          </or>
        </and>
        <and name="Navigation">
          <or name="HomePage">
            <feature name="EditionsList"/>
            <feature name="WorksList"/>
            <feature name="AuthoritiesList"/>
            <feature name="PresentationPage"/>
            <feature name="LatestReleasesPage"/>
          </or>
          <and name="Visualization">
            <or name="EditionsVisualization">
              <feature name="EditionsAsList"/>
              <feature name="EditionsAsImages"/>
            </or>
            <or name="WorksVisualization">
              <feature name="WorksAsList"/>
              <feature name="WorksAsImages"/>
            </or>
            <or name="AuthoritiesVisualization">
              <feature name="AuthoritiesAsList"/>
              <feature name="AuthoritiesAsImages"/>
            </or>
          </and>
          <and name="Pagination">
            <or name="EditionsListPagination">
              <feature name="EditionsInfiniteScroll"/>
              <feature name="EditionsPaginationByRange"/>
            </or>
            <or name="WorksListPagination">
              <feature name="WorksInfiniteScroll"/>
              <feature name="WorksPaginationByRange"/>
            </or>
            <or name="AuthoritiesListPagination">
              <feature name="AuthoritiesInfiniteScroll"/>
              <feature name="AuthoritiesPaginationByRange"/>
            </or>
          </and>
        </and>
        <feature name="Maps"/>
      </and>
      <and mandatory="true" name="ManagementAccess">
        <graphics key="collapsed" value="true"/>
        <and name="ManagementSearch">
          <and name="AdvancedManagementSearch">
            <and name="SearchByElement">
              <feature name="SearchByElementTitle"/>
              <and name="SearchByElementAuthority">
                <feature name="NoAuthority"/>
              </and>
              <feature name="SearchByDraft"/>
              <feature name="SearchByRevised"/>
              <feature name="SearchByElementId"/>
              <feature name="SearchByElementISBN"/>
              <and name="SearchByElementYear">
                <feature name="NoYear"/>
              </and>
              <feature name="SearchByGenre"/>
              <and name="SearchByEditionLocation">
                <feature name="NoEditionLocation"/>
              </and>
            </and>
          </and>
          <and name="Filter">
            <feature name="FilterCollaborators"/>
            <feature name="FilterReporters"/>
            <feature name="FilterElements"/>
          </and>
          <feature name="MapSearch"/>
          <feature name="AlphabeticalPagination"/>
        </and>
        <feature name="Statistics"/>
      </and>
      <and name="ExportFeatures">
        <graphics key="collapsed" value="true"/>
        <and name="ExportType">
          <feature name="PDF"/>
          <feature name="TXT"/>
          <feature name="JSON"/>
          <feature name="EXCEL"/>
          <feature name="CSV"/>
        </and>
        <and name="ExportPossibilities">
          <feature name="SingleItem"/>
          <feature name="SelectionOfItems"/>
        </and>
        <and name="ExportableElements">
          <feature name="ExportAuthority"/>
          <feature name="ExportCollection"/>
          <feature name="ExportEdition"/>
          <feature name="ExportWork"/>
          <feature name="ExportPart"/>
          <feature name="ExportOrganization"/>
          <feature name="ExportBibliographicReference"/>
        </and>
      </and>
    </and>
  </struct>
  <constraints>
    <rule>
      <imp>
        <var>DigitalizedPages</var>
        <var>Collections</var>
      </imp>
    </rule>
    <rule>
      <imp>
        <var>SearchByAuthor</var>
        <var>Author</var>
      </imp>
    </rule>
    <rule>
      <imp>
        <var>SearchByEditor</var>
        <var>Editor</var>
      </imp>
    </rule>
    <rule>
      <imp>
        <var>Maps</var>
        <var>GeographicalLocation</var>
      </imp>
    </rule>
    <rule>
      <imp>
        <var>SearchResultAsMap</var>
        <var>GeographicalLocation</var>
      </imp>
    </rule>
    <rule>
      <imp>
        <var>Collections</var>
        <disj>
          <var>DigitalizedPages</var>
          <disj>
            <var>AudioFiles</var>
            <var>VideoFiles</var>
          </disj>
        </disj>
      </imp>
    </rule>
    <rule>
      <imp>
        <var>SimpleSearchByAuthor</var>
        <var>Author</var>
      </imp>
    </rule>
    <rule>
      <imp>
        <var>MapSearch</var>
        <var>GeographicalLocation</var>
      </imp>
    </rule>
    <rule>
      <imp>
        <var>ExportCollection</var>
        <var>CollectionElement</var>
      </imp>
    </rule>
    <rule>
      <imp>
        <var>ExportOrganization</var>
        <var>Organization</var>
      </imp>
    </rule>
    <rule>
      <imp>
```

```
          <var>ExportPart</var>
          <var>PartElement</var>
        </imp>
      </rule>
    </constraints>
</featureModel>
```

Listing A.1: Source XML code corresponding to the feature model

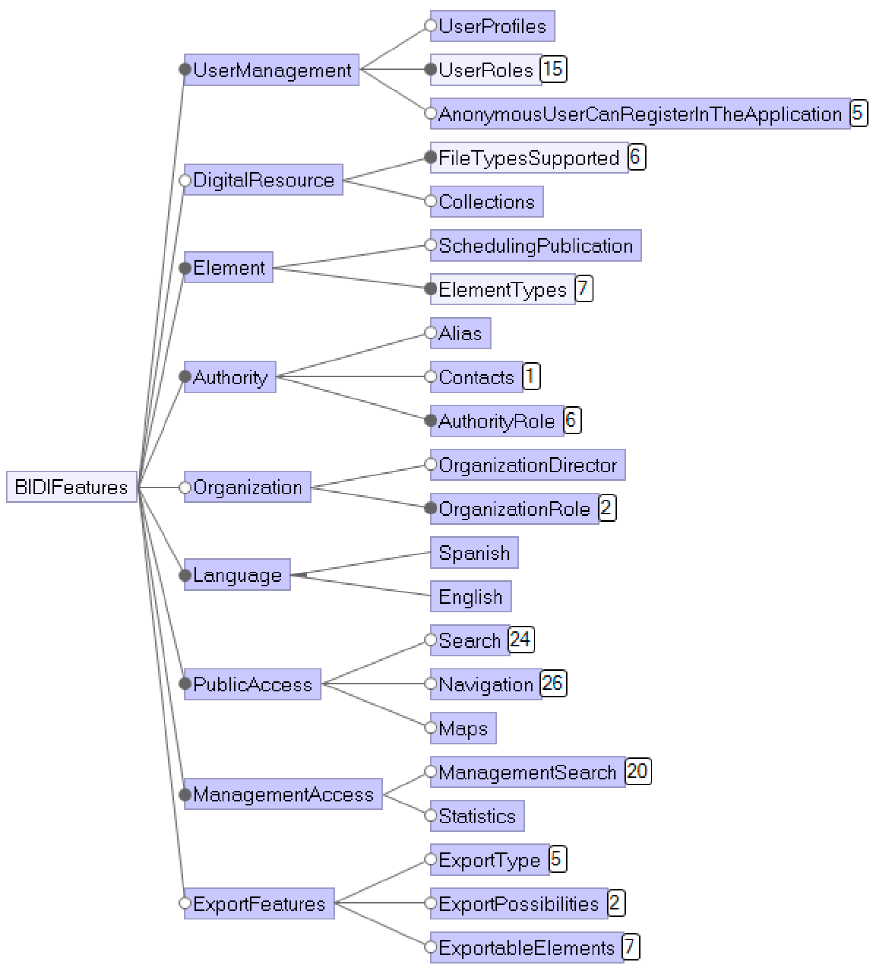


Figure A.1: Collapsed feature model

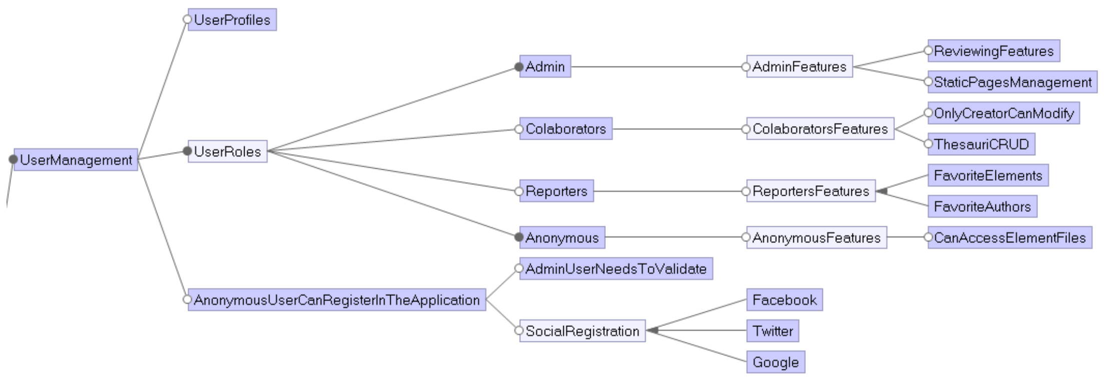


Figure A.2: User management features

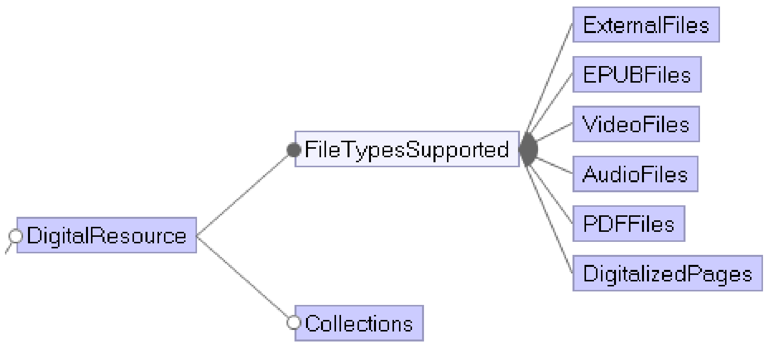


Figure A.3: Digital resources features

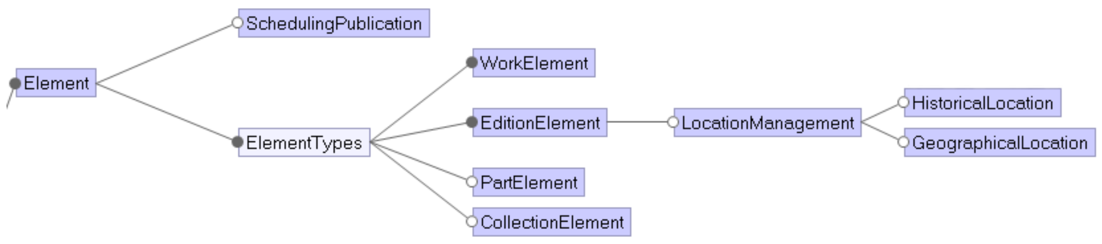


Figure A.4: Element features

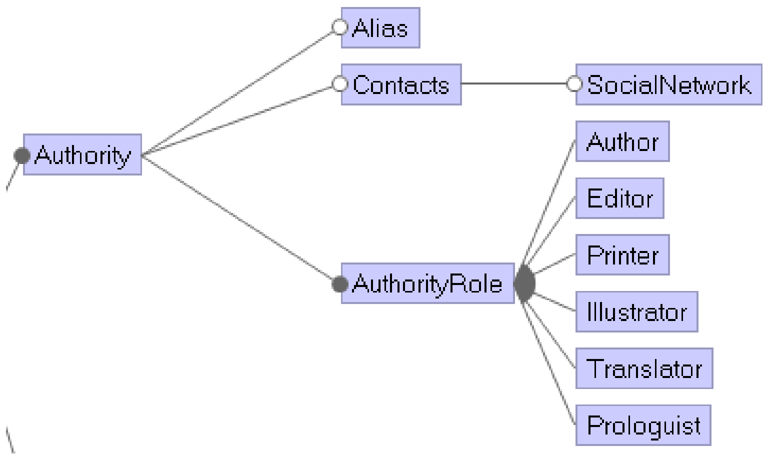


Figure A.5: Authority features

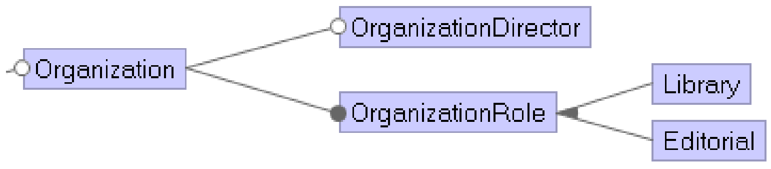


Figure A.6: Organization features

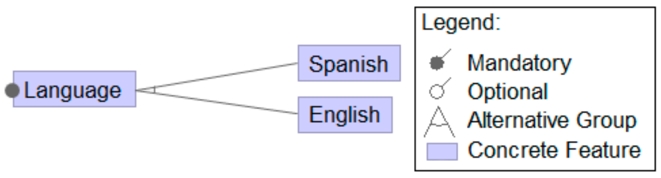


Figure A.7: Language features

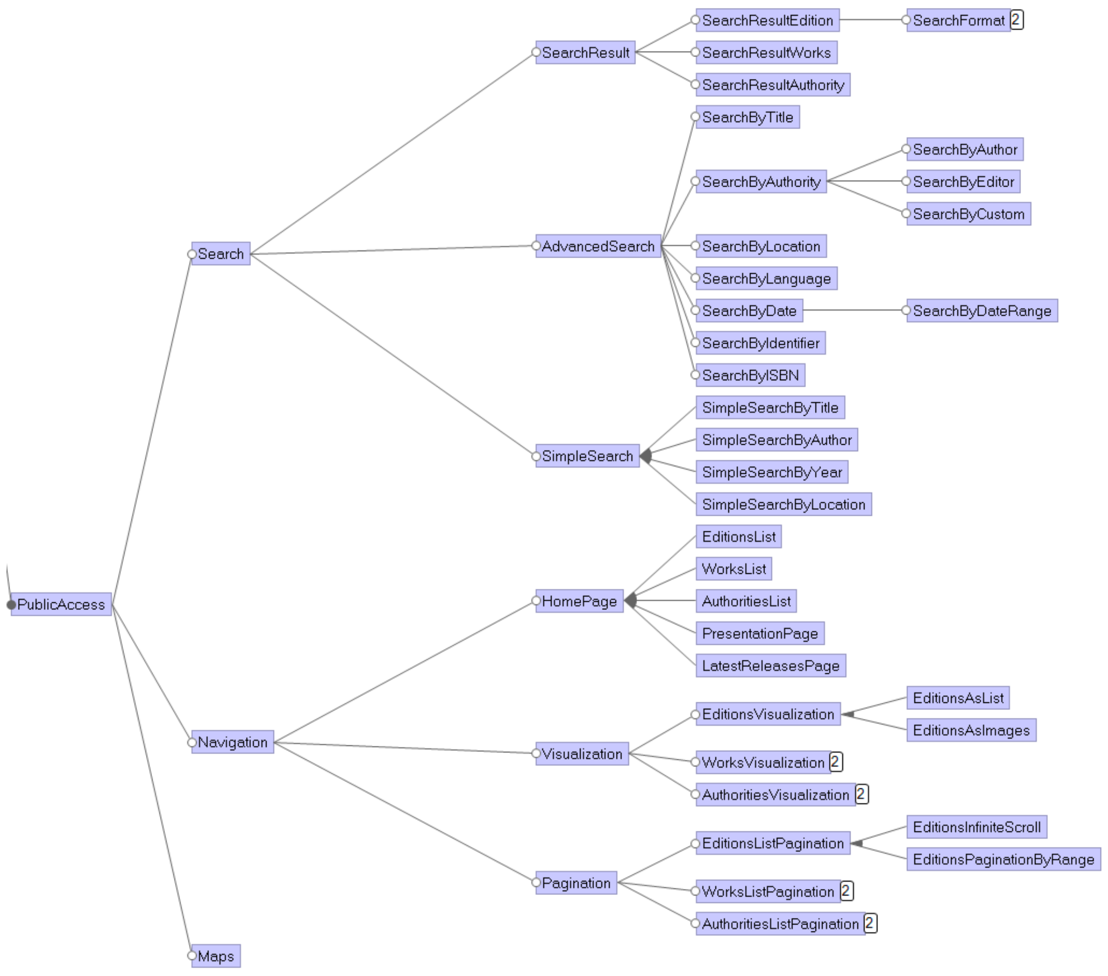


Figure A.8: Public access features

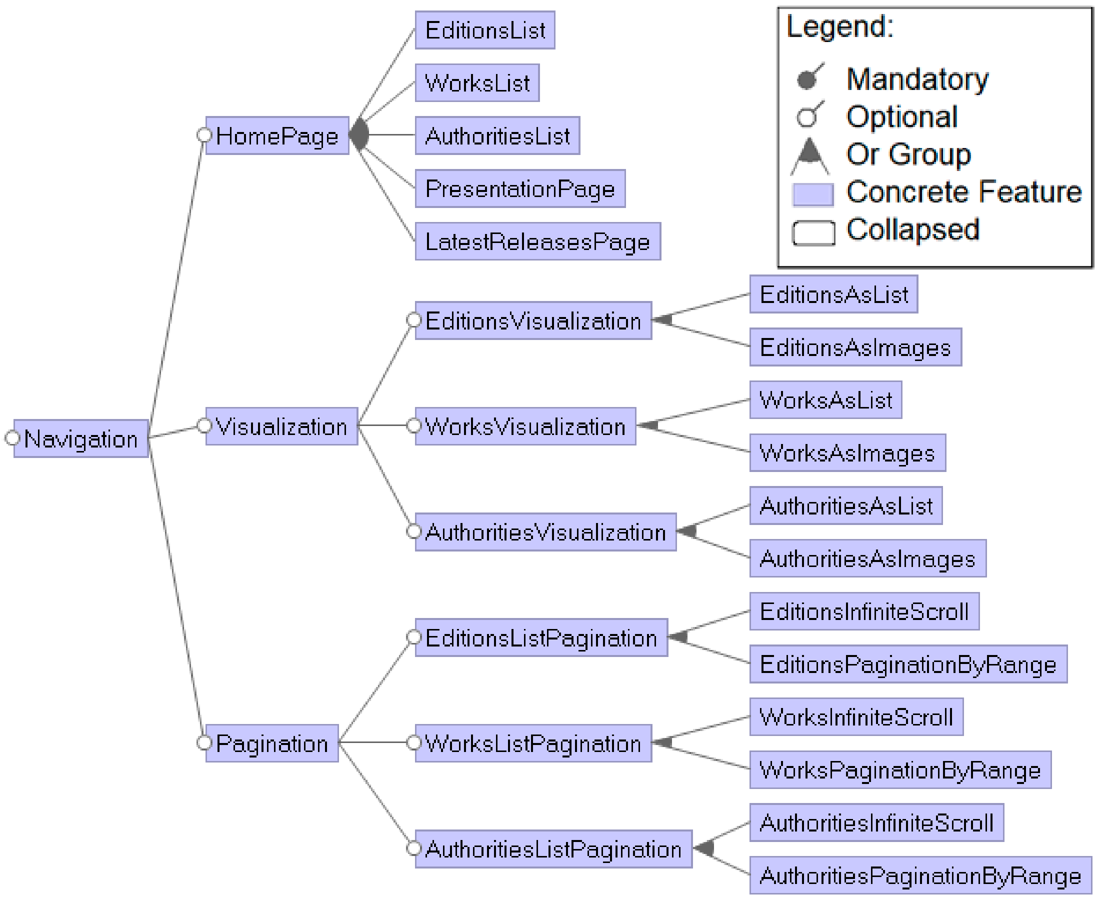


Figure A.9: Navigation features

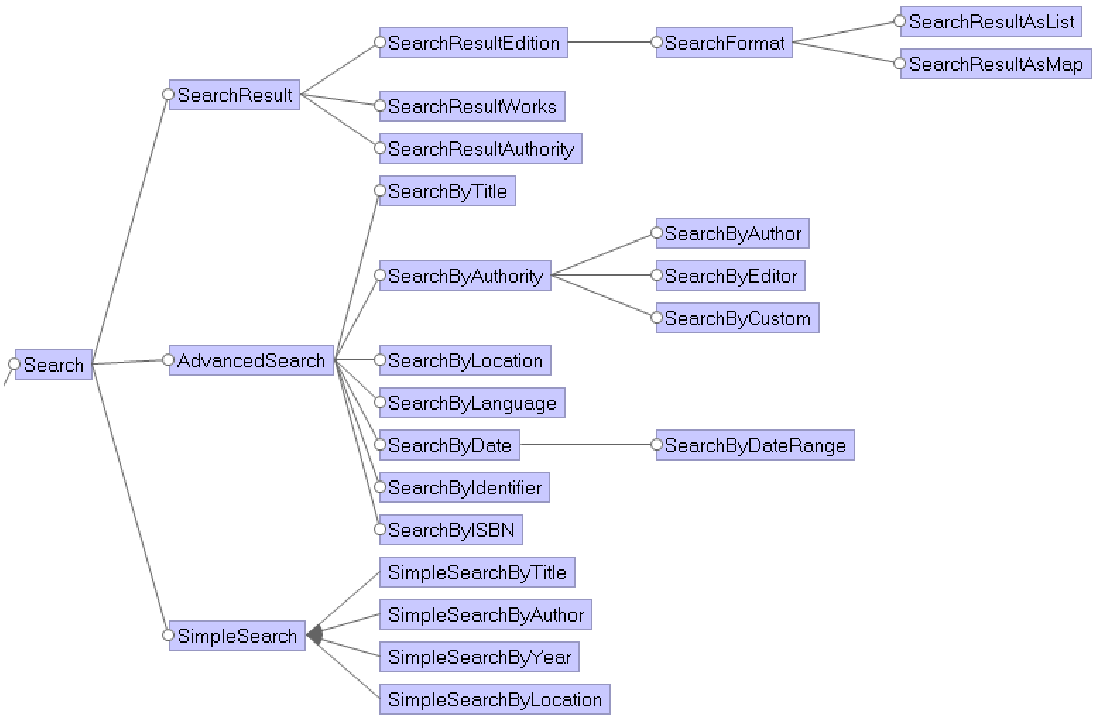


Figure A.10: Public search features

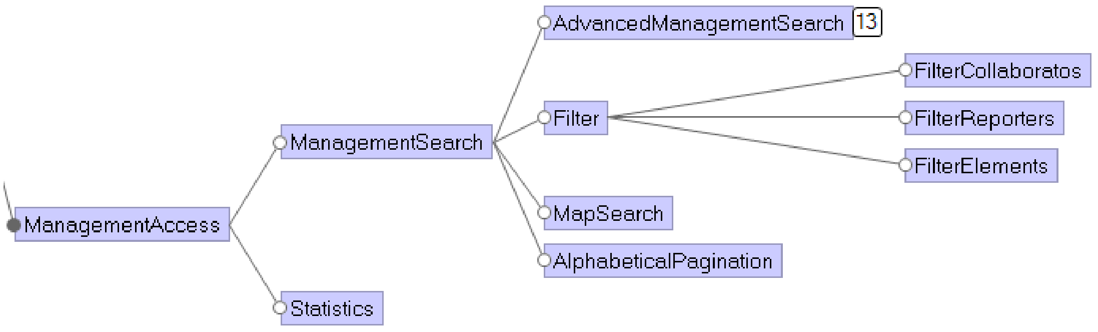


Figure A.11: Management features collapsed

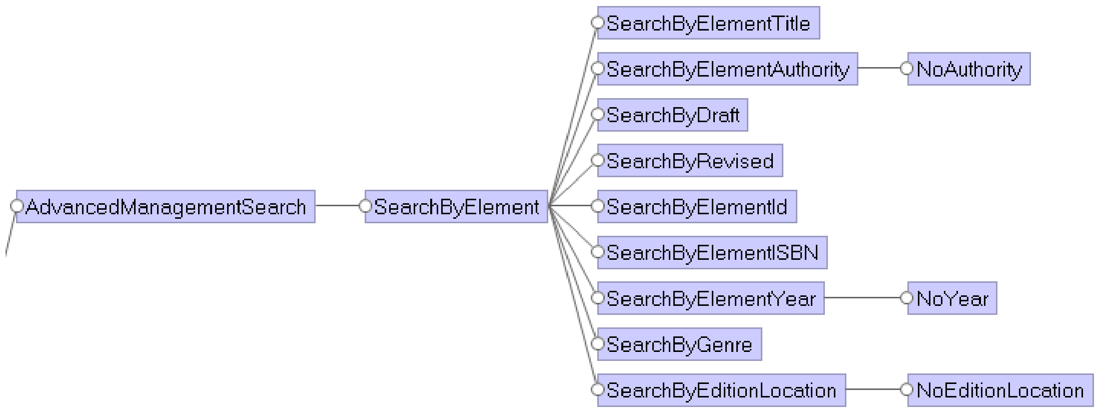


Figure A.12: Management search

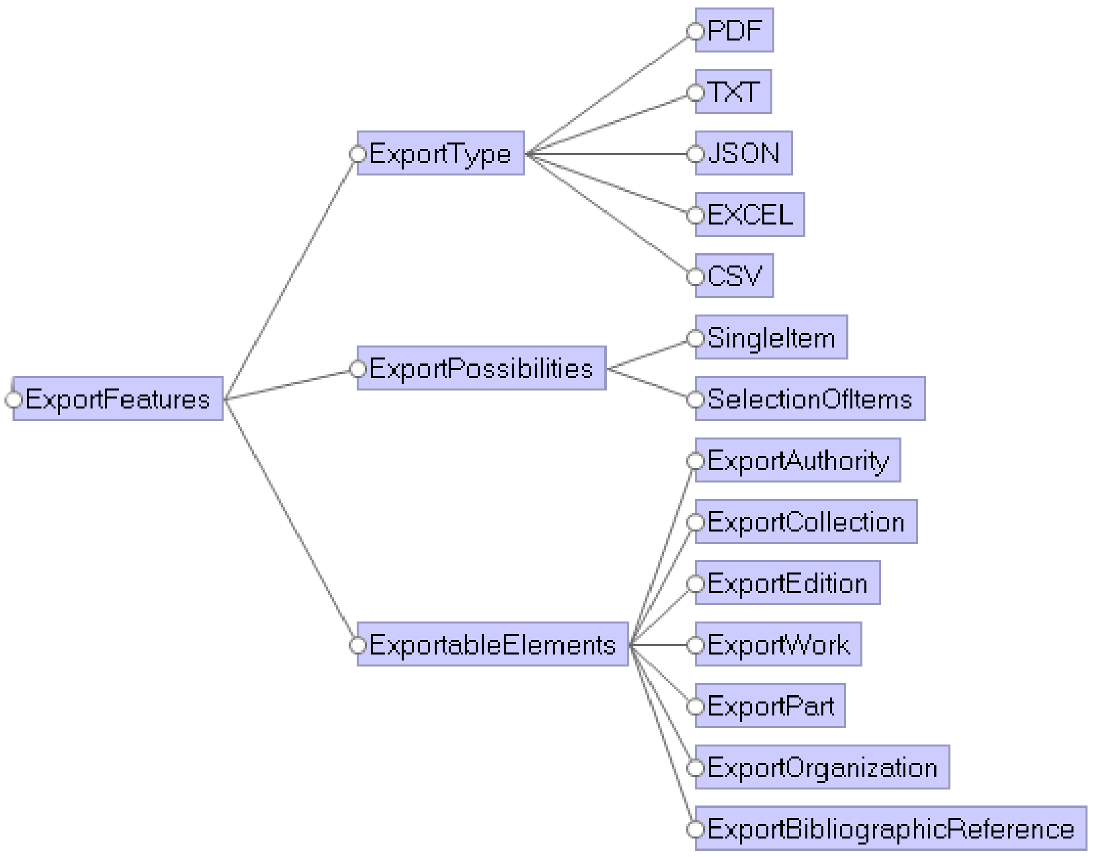
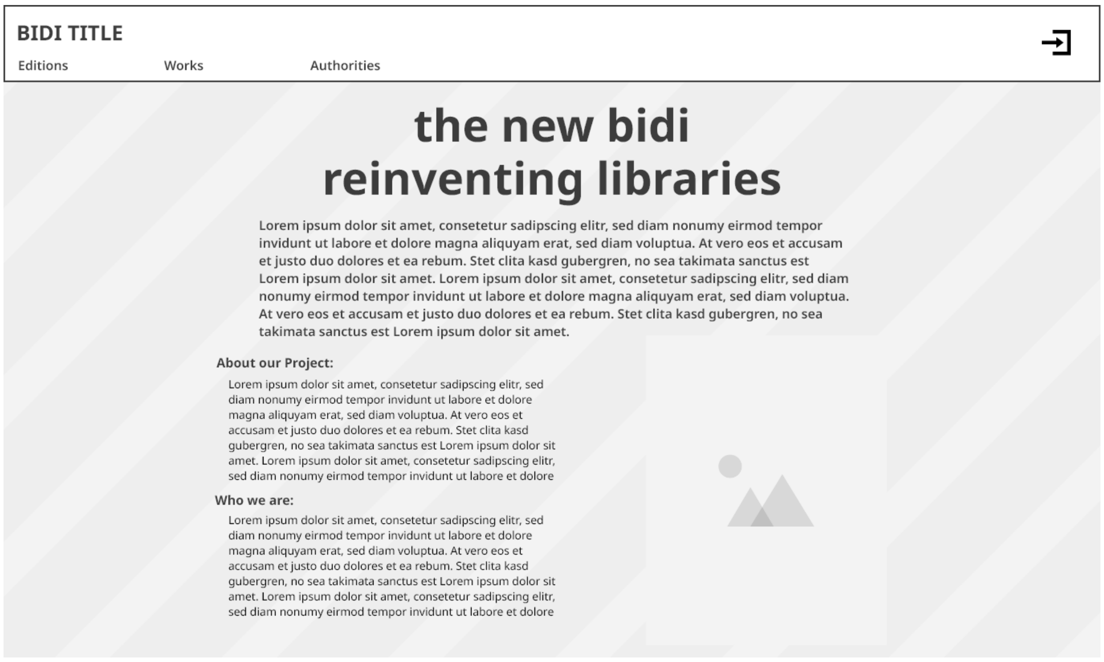


Figure A.13: Export features

Appendix B

User interface mockups

EXTENDED version of the user interface analysis to include those mock-up that could not fit the space dedicated for it in the section 5.1.2.



the new bidi
reinventing libraries

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

About our Project:

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore

Who we are:

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore

Figure B.1: Home for Static Pages

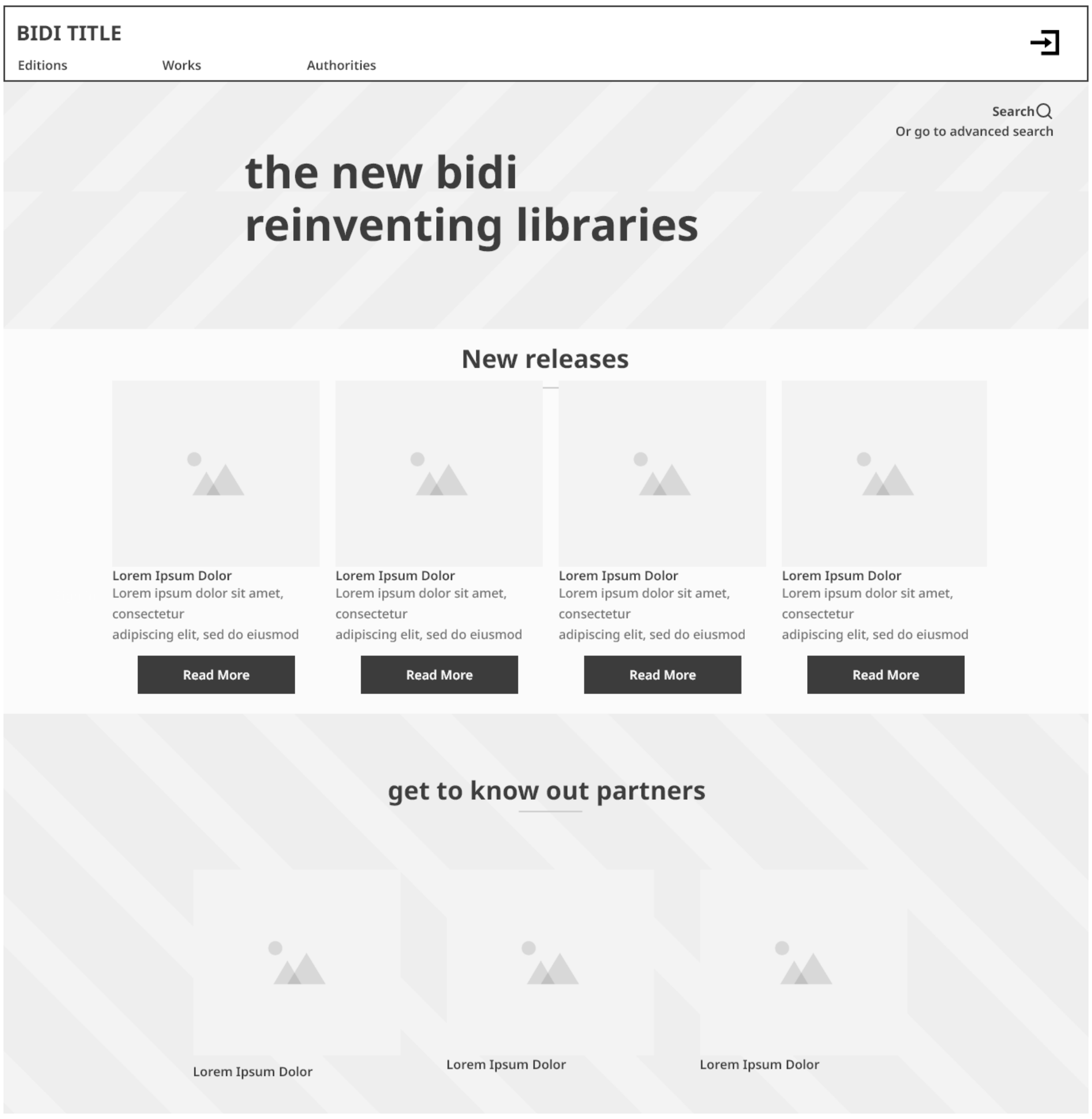
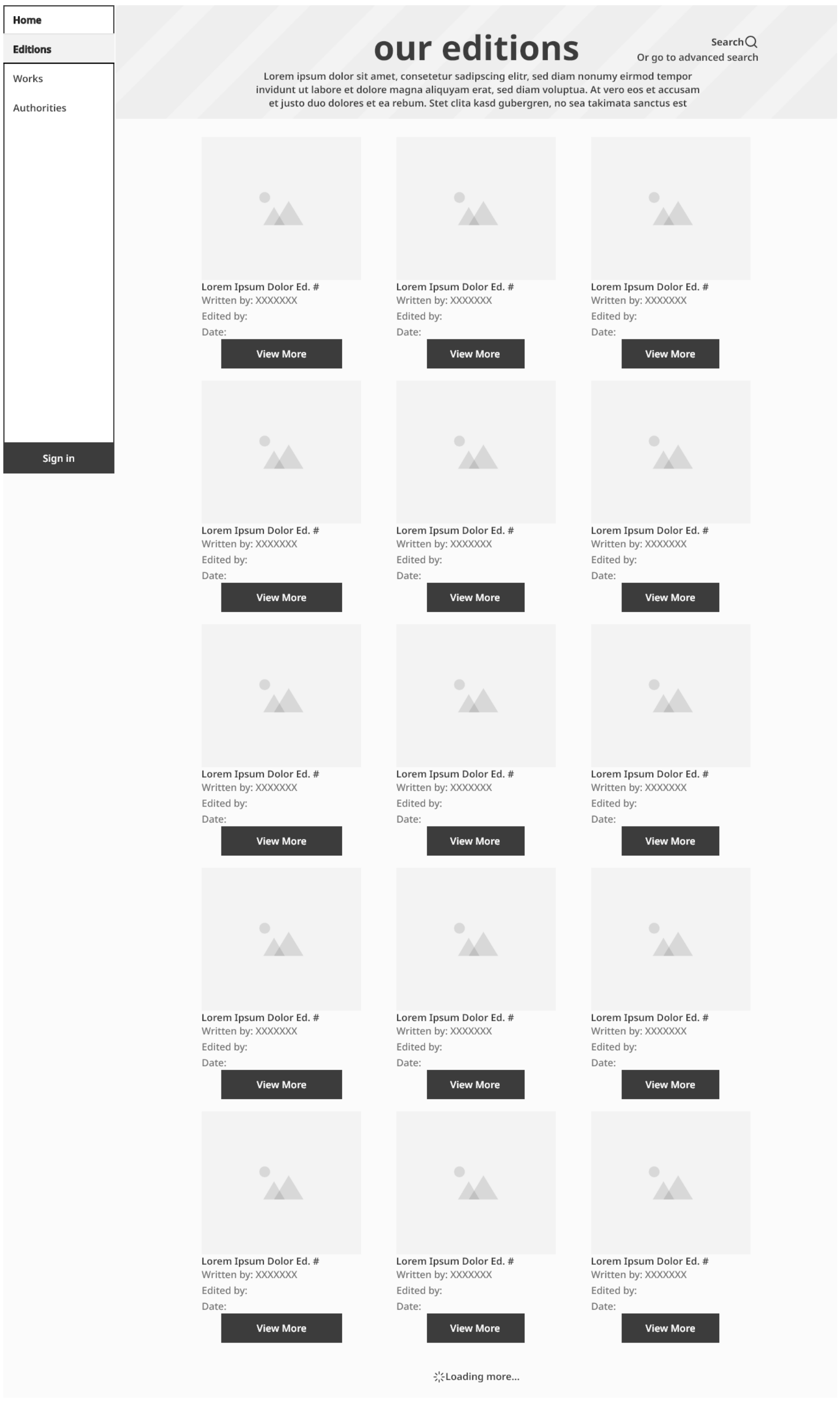


Figure B.2: Home for New releases



our editions

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est

Lorem Ipsum Dolor Ed. #
Written by: XXXXXXX
Edited by:
Date:

Lorem Ipsum Dolor Ed. #
Written by: XXXXXXX
Edited by:
Date:

Lorem Ipsum Dolor Ed. #
Written by: XXXXXXX
Edited by:
Date:

Lorem Ipsum Dolor Ed. #
Written by: XXXXXXX
Edited by:
Date:

Lorem Ipsum Dolor Ed. #
Written by: XXXXXXX
Edited by:
Date:

Lorem Ipsum Dolor Ed. #
Written by: XXXXXXX
Edited by:
Date:

Lorem Ipsum Dolor Ed. #
Written by: XXXXXXX
Edited by:
Date:

Lorem Ipsum Dolor Ed. #
Written by: XXXXXXX
Edited by:
Date:

Lorem Ipsum Dolor Ed. #
Written by: XXXXXXX
Edited by:
Date:

Lorem Ipsum Dolor Ed. #
Written by: XXXXXXX
Edited by:
Date:

Lorem Ipsum Dolor Ed. #
Written by: XXXXXXX
Edited by:
Date:

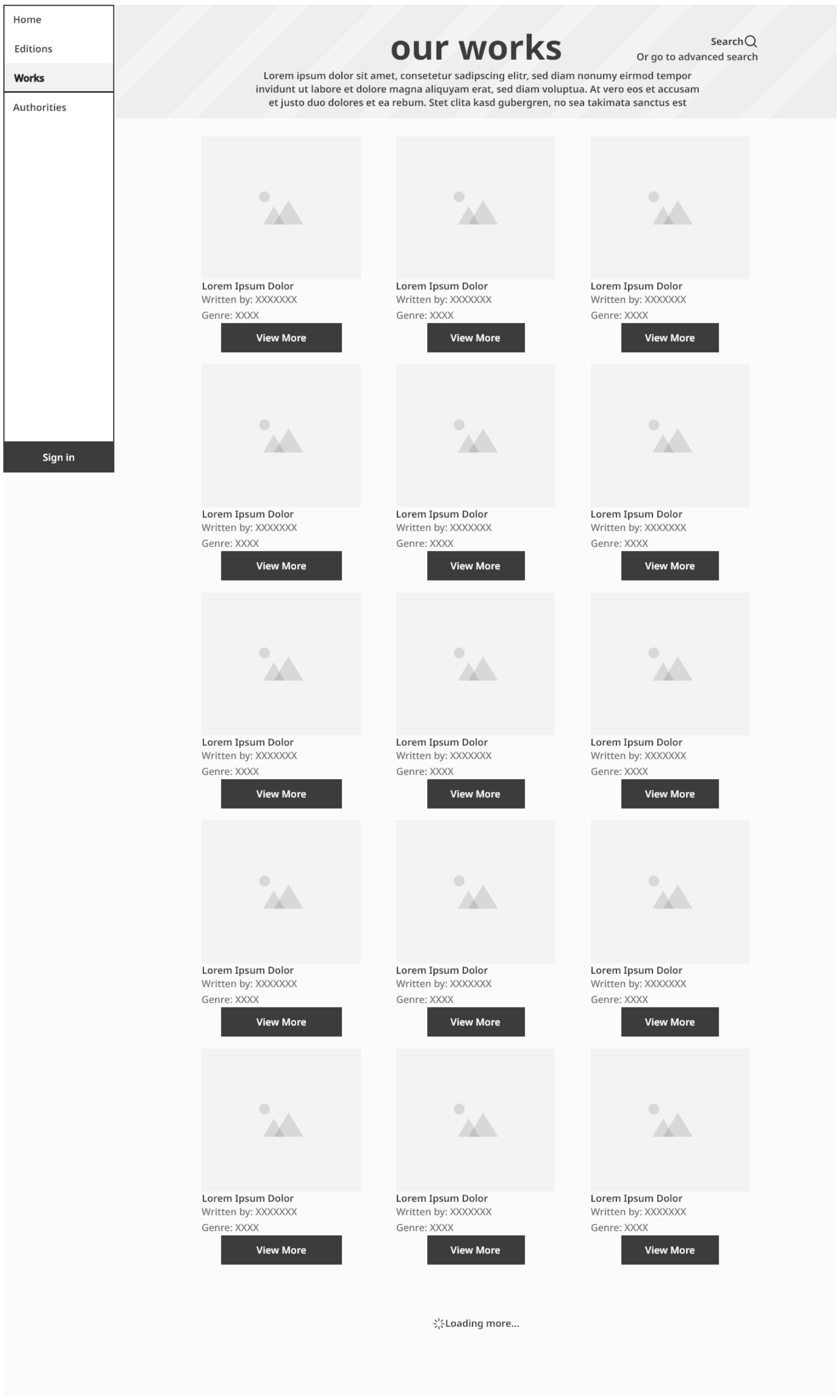
Lorem Ipsum Dolor Ed. #
Written by: XXXXXXX
Edited by:
Date:

Lorem Ipsum Dolor Ed. #
Written by: XXXXXXX
Edited by:
Date:

Lorem Ipsum Dolor Ed. #
Written by: XXXXXXX
Edited by:
Date:

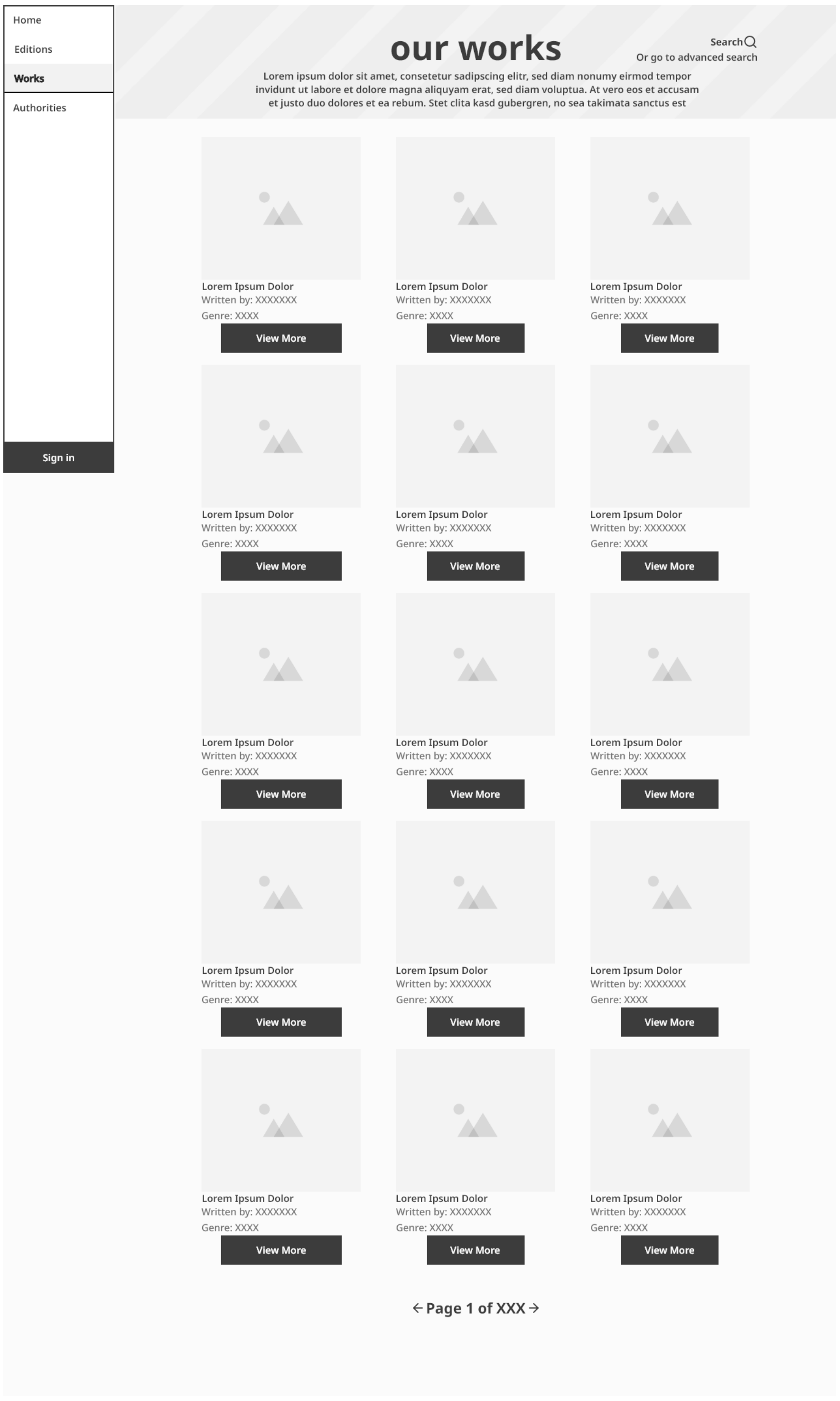
Lorem Ipsum Dolor Ed. #
Written by: XXXXXXX
Edited by:
Date:

Figure B.3: Home for Editions with Infinite scroll



Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est

Figure B.4: Home for Works with infinite scroll



Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est

Figure B.5: Home for works with pagination

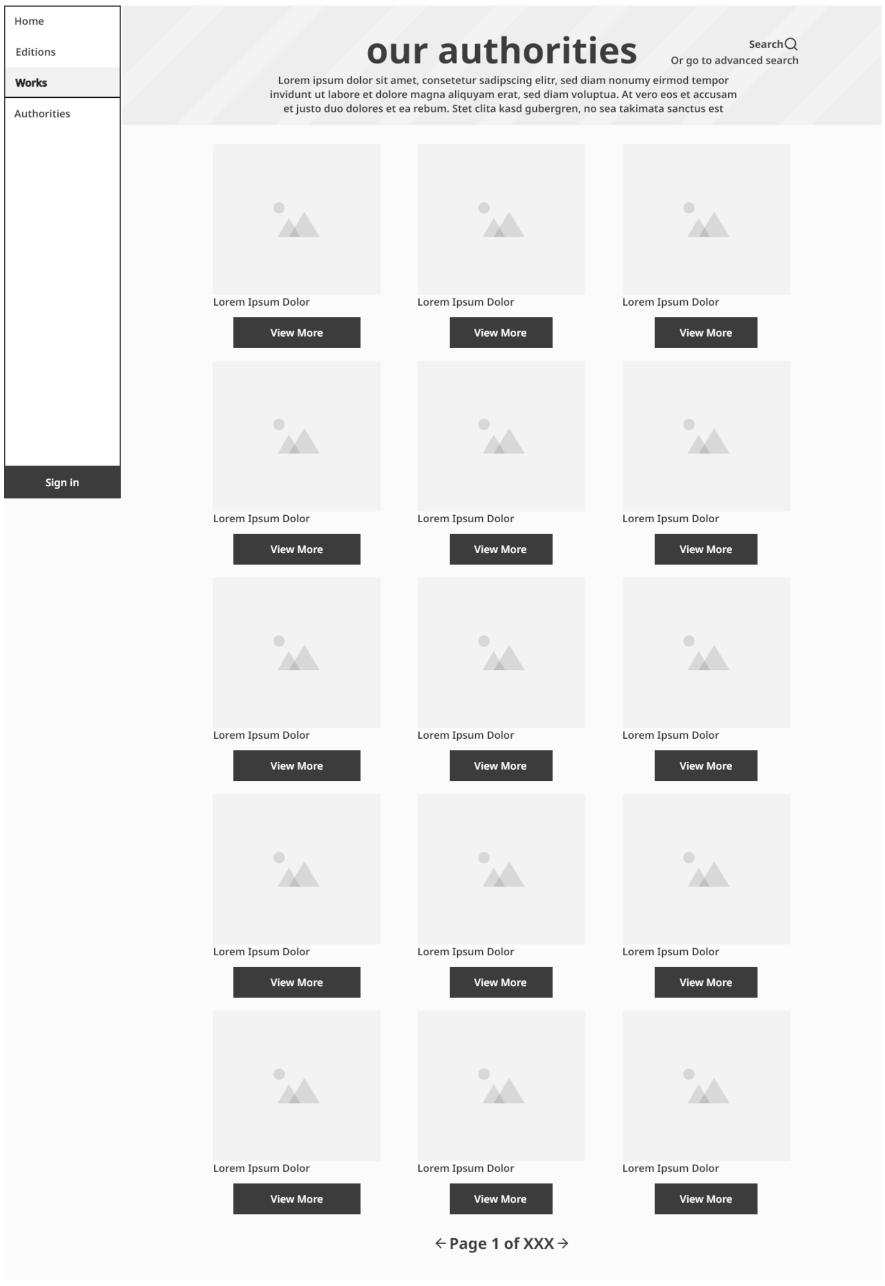

Home
Editions
Works
Authorities
Sign in
our authorities
Search
Or go to advanced search
Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est
Lorem Ipsum Dolor
View More
Page 1 of XXX

Figure B.6: Home for authorities with pagination

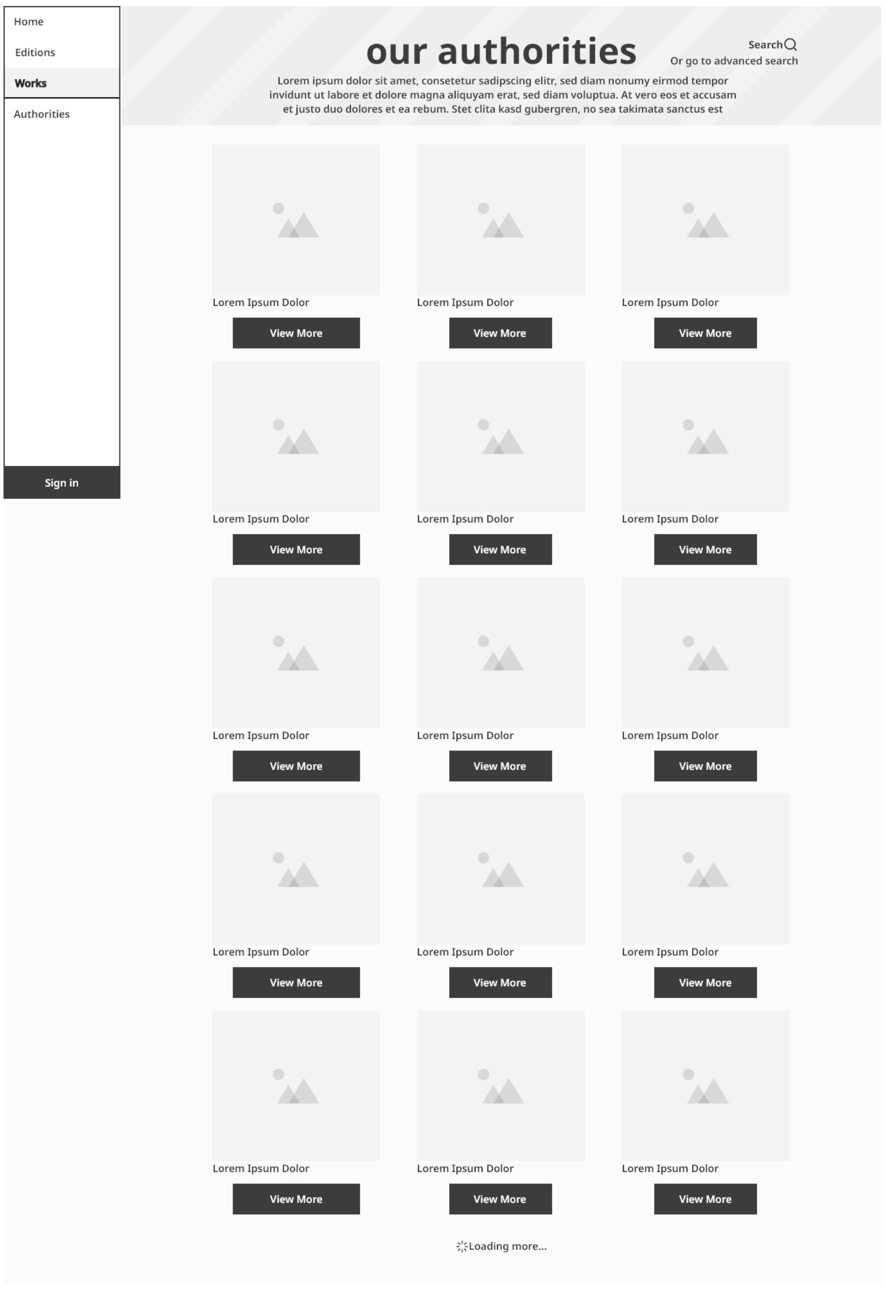

Home
Editions
Works
Authorities
Sign in
our authorities
Search
Or go to advanced search
Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est
Lorem Ipsum Dolor
View More
Loading more...

Figure B.7: Home for authorities with infinite scroll

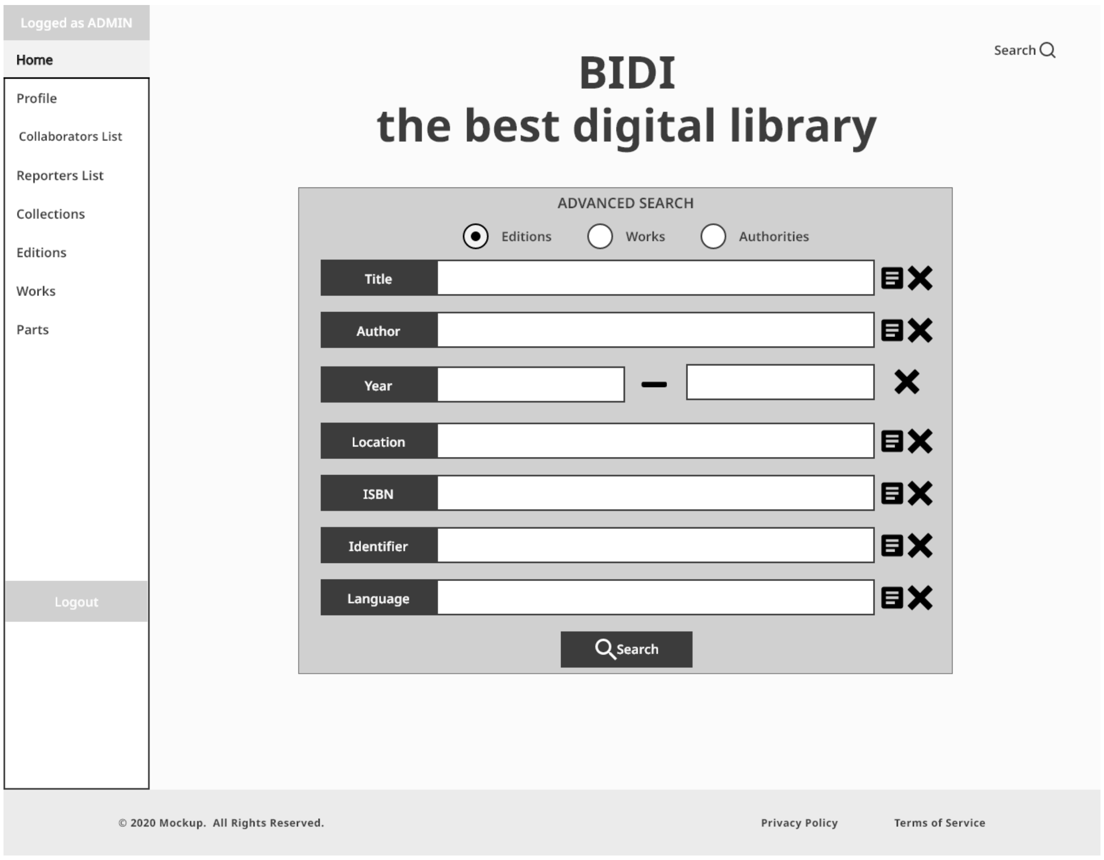


Figure B.8: Advanced search logged as Admin

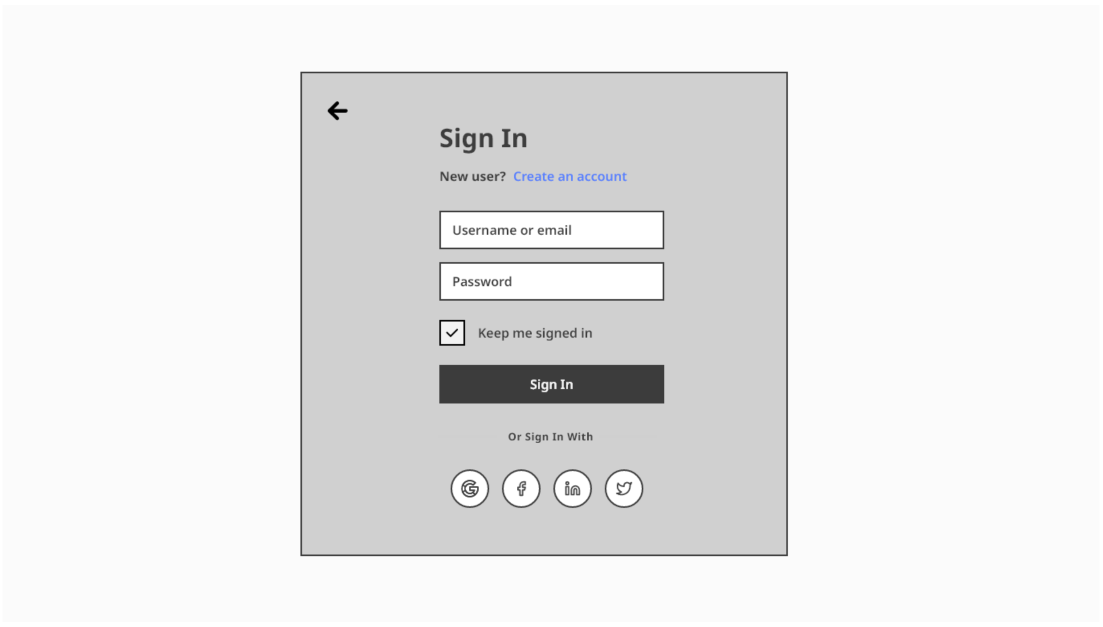


Figure B.9: Sign in

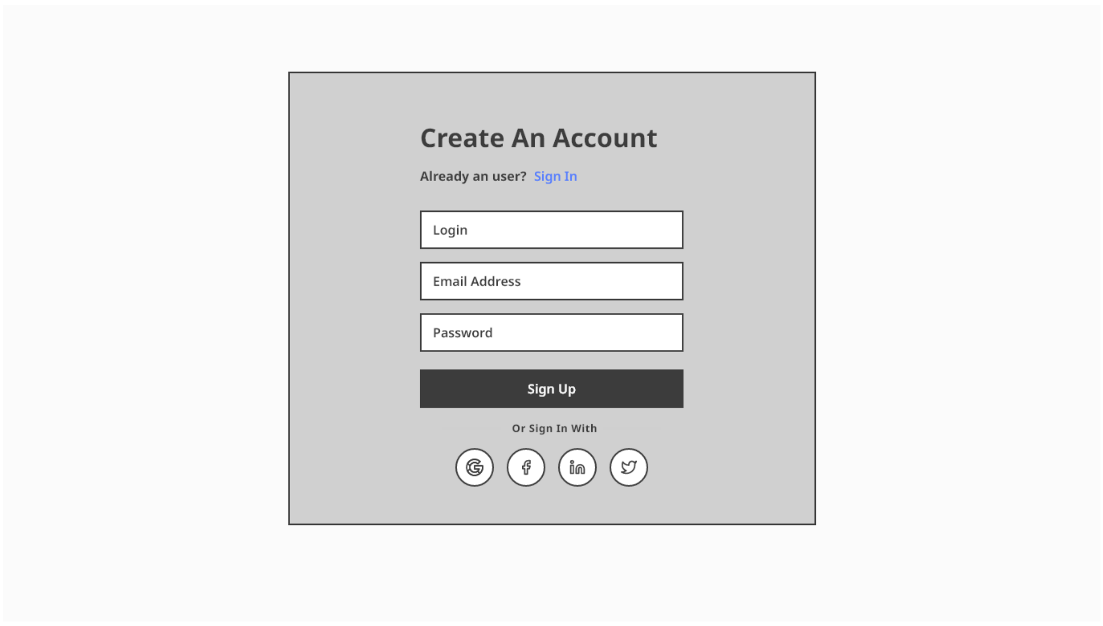


Figure B.10: Sign up simple

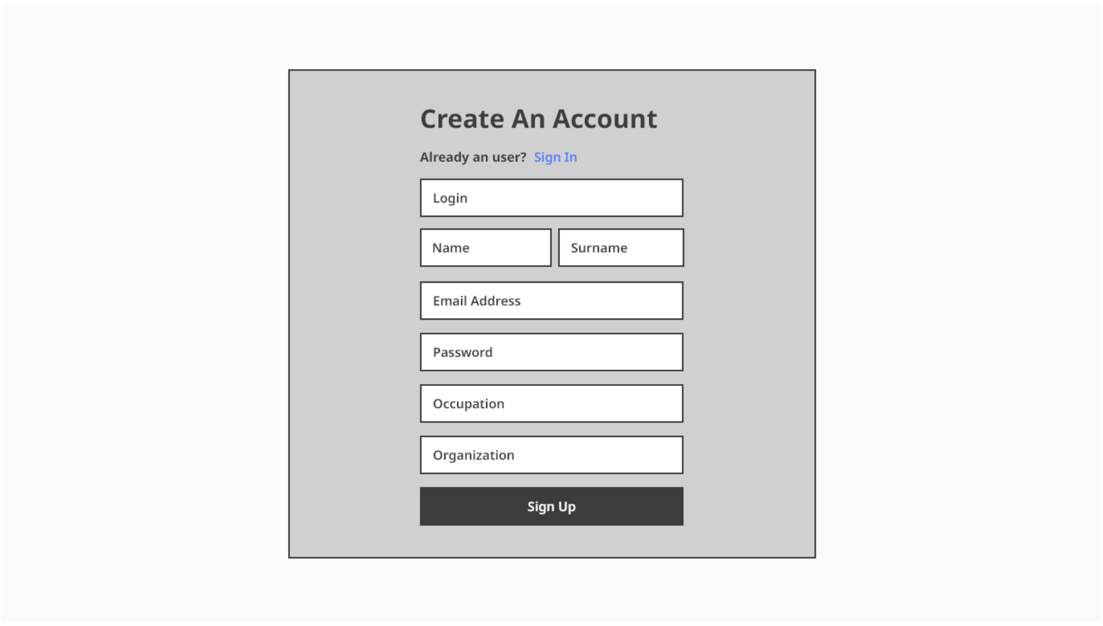
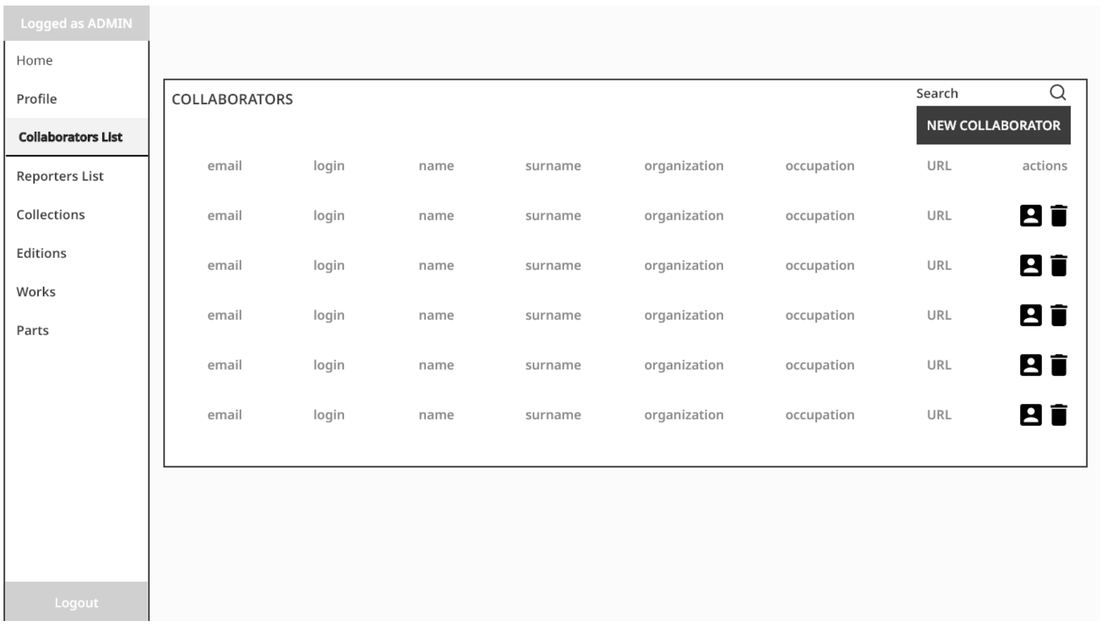


Figure B.11: Sign up with extended user profiles



email	login	name	surname	organization	occupation	URL	actions
email	login	name	surname	organization	occupation	URL	👤 🗑
email	login	name	surname	organization	occupation	URL	👤 🗑
email	login	name	surname	organization	occupation	URL	👤 🗑
email	login	name	surname	organization	occupation	URL	👤 🗑
email	login	name	surname	organization	occupation	URL	👤 🗑

Figure B.12: Admin management - Collaborators list

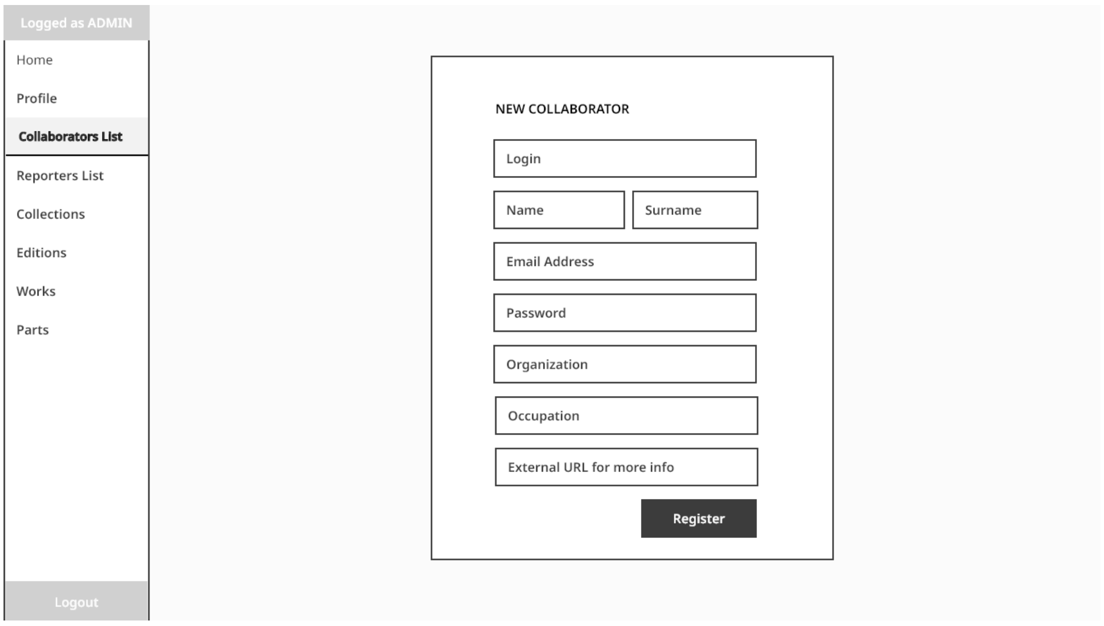
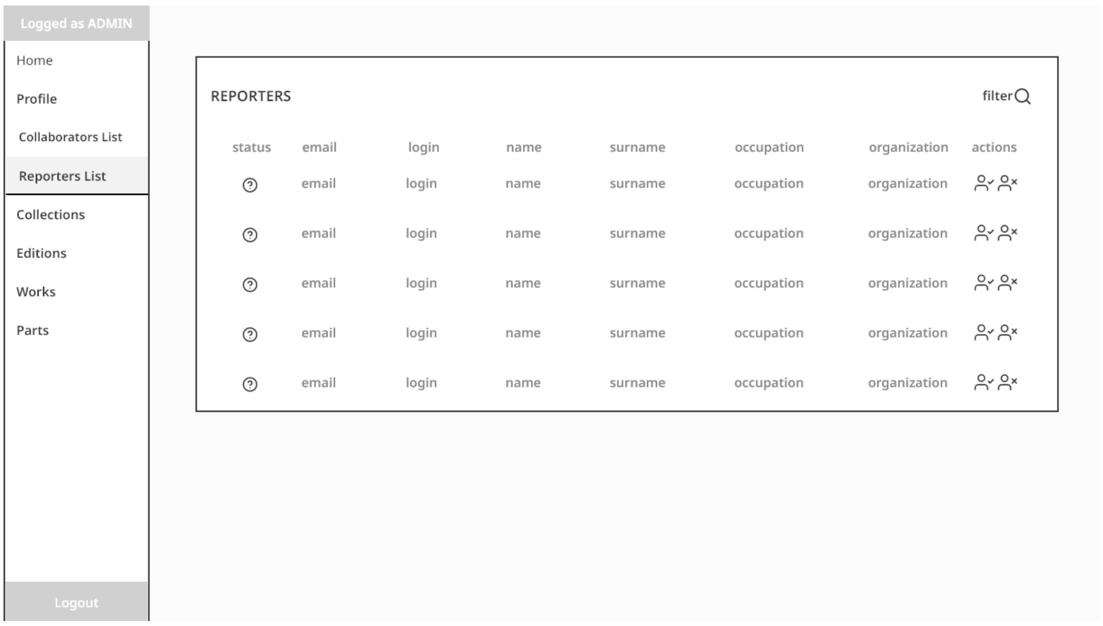


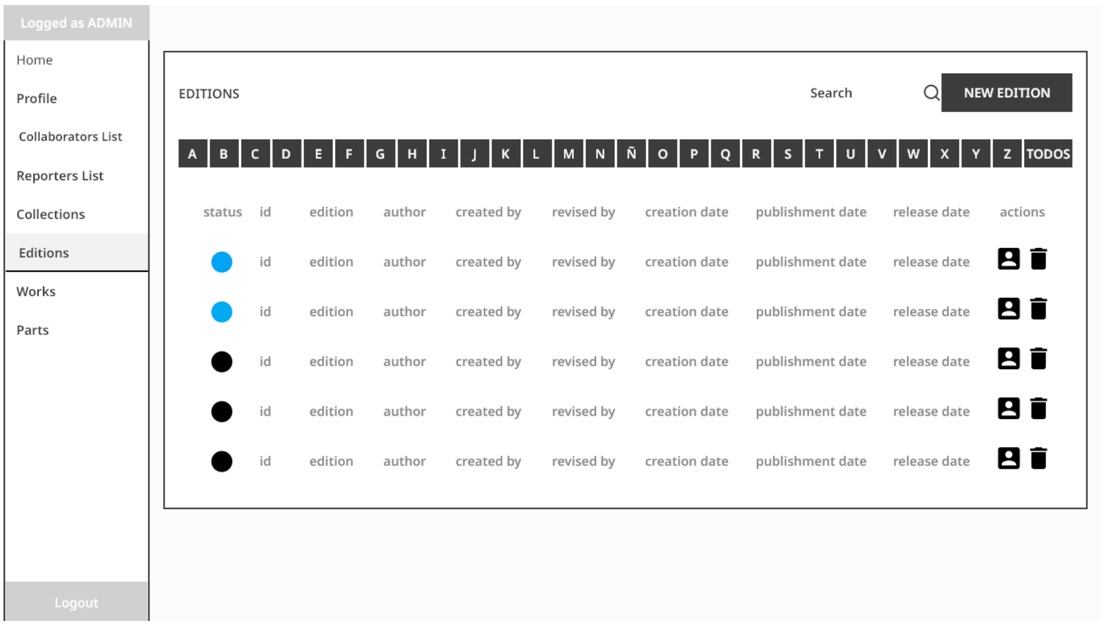
Figure B.13: Admin management - Register collaborator



REPORTERS

status	email	login	name	surname	occupation	organization	actions
?	email	login	name	surname	occupation	organization	✓ ✗
?	email	login	name	surname	occupation	organization	✓ ✗
?	email	login	name	surname	occupation	organization	✓ ✗
?	email	login	name	surname	occupation	organization	✓ ✗
?	email	login	name	surname	occupation	organization	✓ ✗

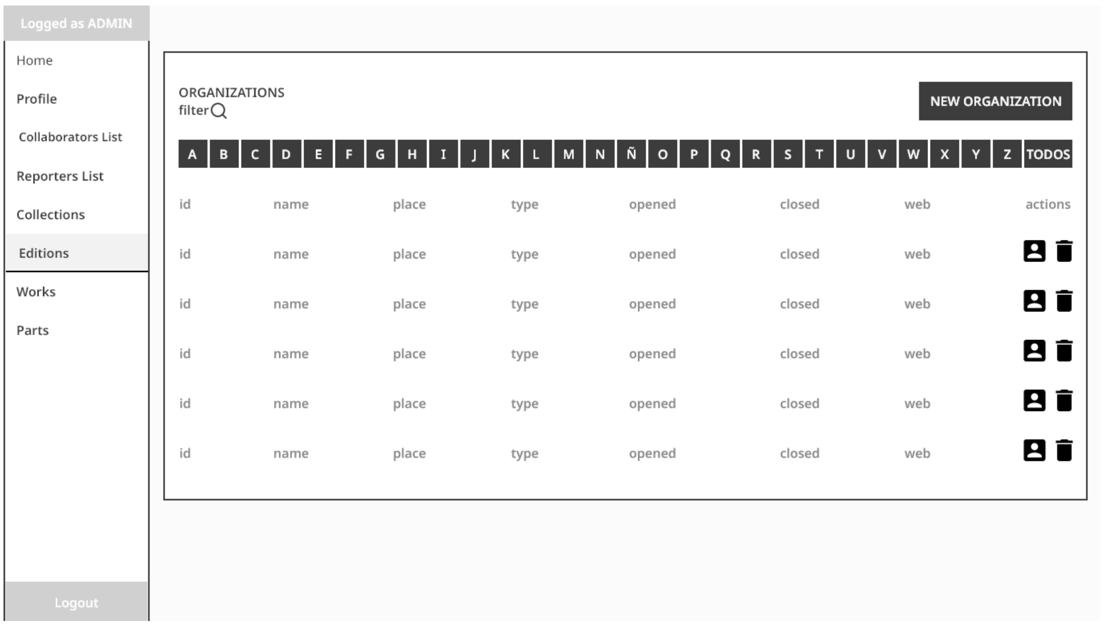
Figure B.14: Admin management - Reporters list



EDITIONS

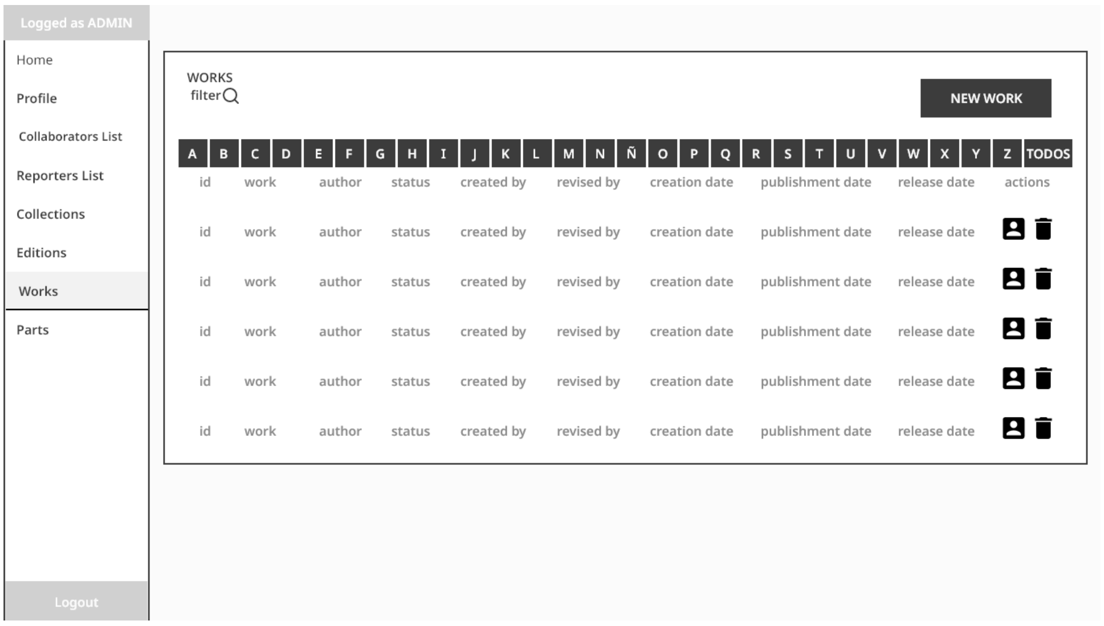
status	id	edition	author	created by	revised by	creation date	publishment date	release date	actions
🔵	id	edition	author	created by	revised by	creation date	publishment date	release date	
🔵	id	edition	author	created by	revised by	creation date	publishment date	release date	
●	id	edition	author	created by	revised by	creation date	publishment date	release date	
●	id	edition	author	created by	revised by	creation date	publishment date	release date	
●	id	edition	author	created by	revised by	creation date	publishment date	release date	

Figure B.15: Admin management - Edition list



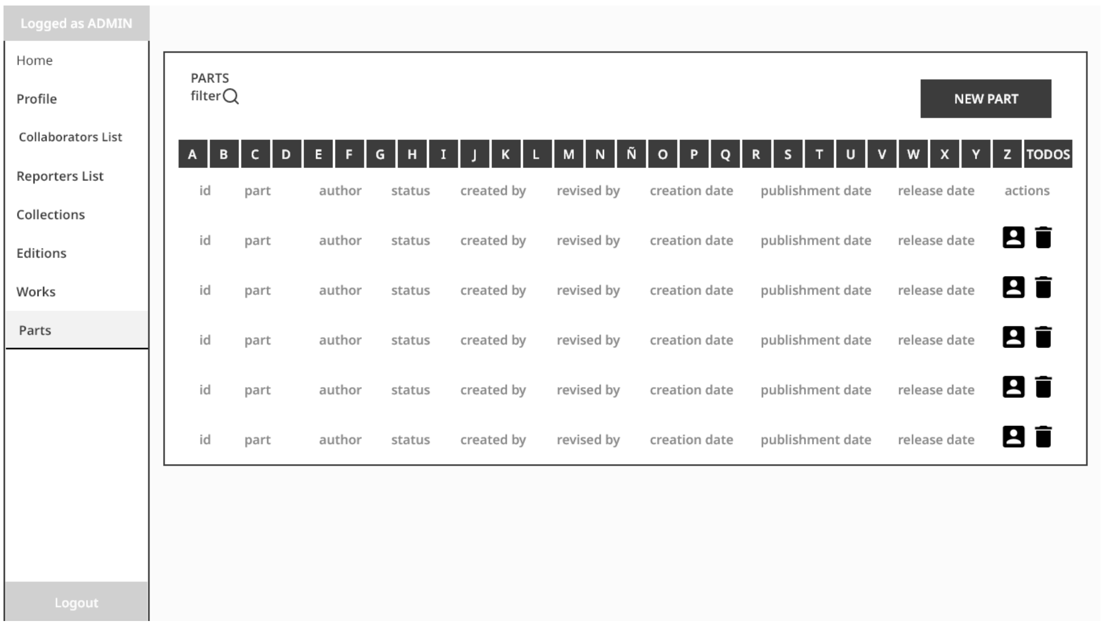
id	name	place	type	opened	closed	web	actions
id	name	place	type	opened	closed	web	
id	name	place	type	opened	closed	web	
id	name	place	type	opened	closed	web	
id	name	place	type	opened	closed	web	
id	name	place	type	opened	closed	web	

Figure B.16: Admin management - Organizations list



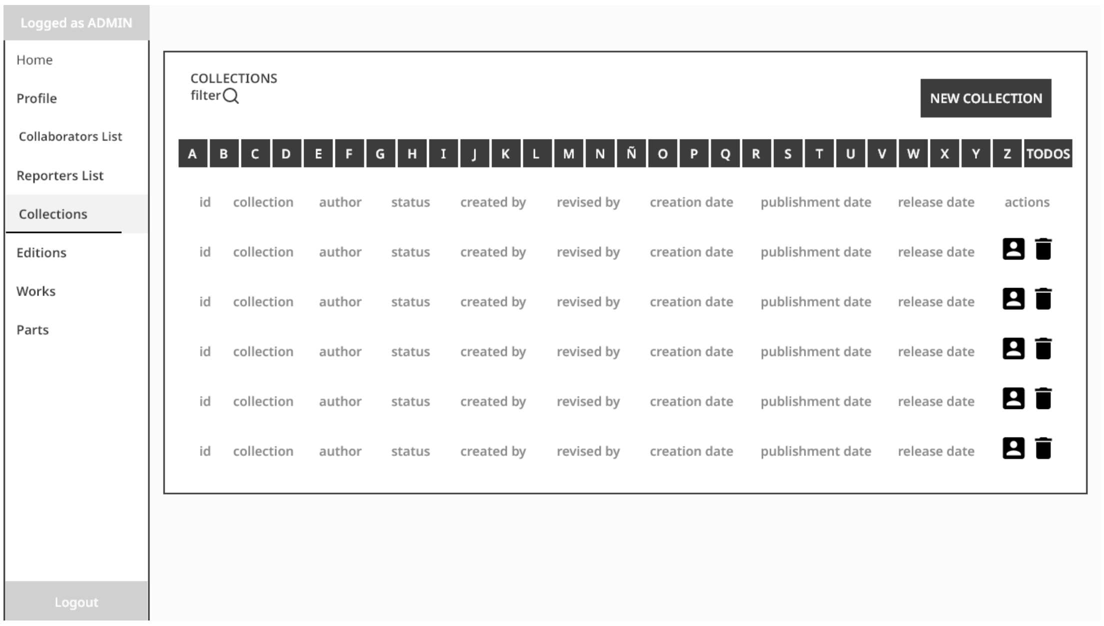
id	work	author	status	created by	revised by	creation date	publishment date	release date	actions
id	work	author	status	created by	revised by	creation date	publishment date	release date	
id	work	author	status	created by	revised by	creation date	publishment date	release date	
id	work	author	status	created by	revised by	creation date	publishment date	release date	
id	work	author	status	created by	revised by	creation date	publishment date	release date	
id	work	author	status	created by	revised by	creation date	publishment date	release date	

Figure B.17: Admin management - Works list



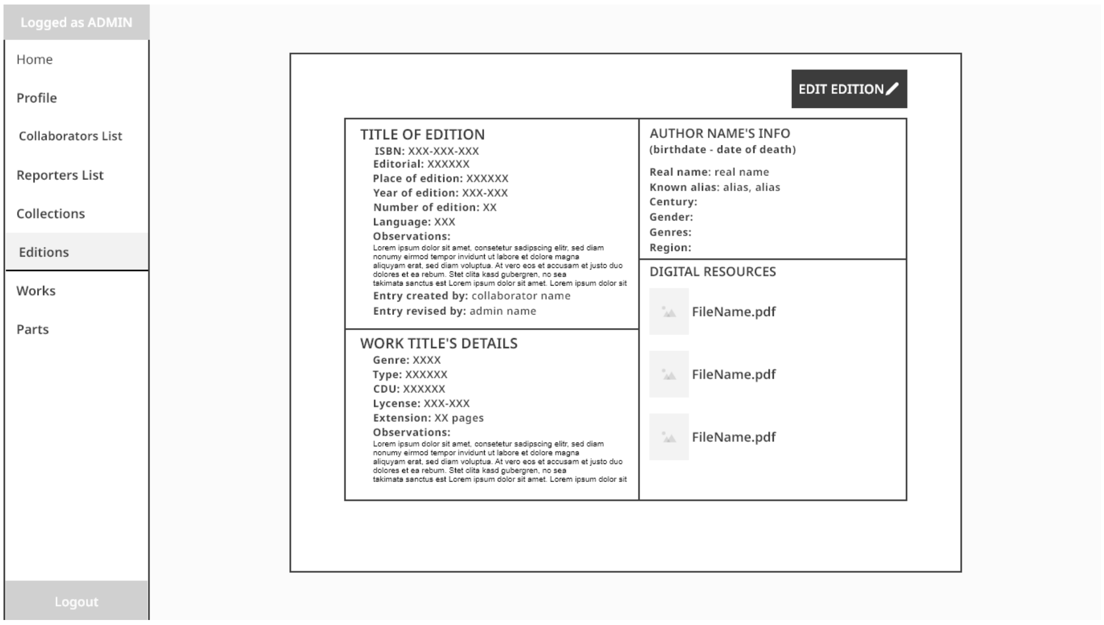
id	part	author	status	created by	revised by	creation date	publishment date	release date	actions
id	part	author	status	created by	revised by	creation date	publishment date	release date	
id	part	author	status	created by	revised by	creation date	publishment date	release date	
id	part	author	status	created by	revised by	creation date	publishment date	release date	
id	part	author	status	created by	revised by	creation date	publishment date	release date	
id	part	author	status	created by	revised by	creation date	publishment date	release date	

Figure B.18: Admin management - Parts list



id	collection	author	status	created by	revised by	creation date	publishment date	release date	actions
id	collection	author	status	created by	revised by	creation date	publishment date	release date	
id	collection	author	status	created by	revised by	creation date	publishment date	release date	
id	collection	author	status	created by	revised by	creation date	publishment date	release date	
id	collection	author	status	created by	revised by	creation date	publishment date	release date	
id	collection	author	status	created by	revised by	creation date	publishment date	release date	

Figure B.19: Admin management - Collection list



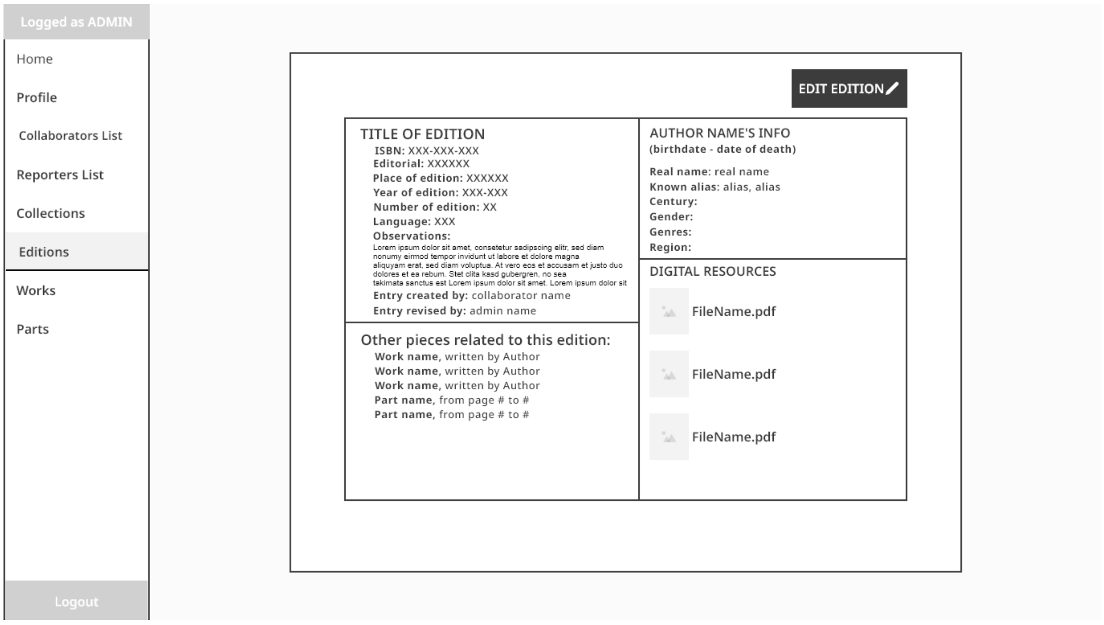
TITLE OF EDITION
ISBN: XXX-XXX-XXX
Editorial: XXXXXX
Place of edition: XXXXXX
Year of edition: XXX-XXX
Number of edition: XX
Language: XXX
Observations:
Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit
Entry created by: collaborator name
Entry revised by: admin name

WORK TITLE'S DETAILS
Genre: XXXX
Type: XXXXXX
CDU: XXXXXX
Lycense: XXX-XXX
Extension: XX pages
Observations:
Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit

AUTHOR NAME'S INFO
(birthdate - date of death)
Real name: real name
Known alias: alias, alias
Century:
Gender:
Genres:
Region:

DIGITAL RESOURCES
FileName.pdf
FileName.pdf
FileName.pdf

Figure B.20: Admin management - Edition detail



TITLE OF EDITION
ISBN: XXX-XXX-XXX
Editorial: XXXXXX
Place of edition: XXXXXX
Year of edition: XXX-XXX
Number of edition: XX
Language: XXX
Observations:
Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit
Entry created by: collaborator name
Entry revised by: admin name

Other pieces related to this edition:
Work name, written by Author
Work name, written by Author
Work name, written by Author
Part name, from page # to #
Part name, from page # to #

AUTHOR NAME'S INFO
(birthdate - date of death)
Real name: real name
Known alias: alias, alias
Century:
Gender:
Genres:
Region:

DIGITAL RESOURCES
FileName.pdf
FileName.pdf
FileName.pdf

Figure B.21: Admin management - Edition with several works related

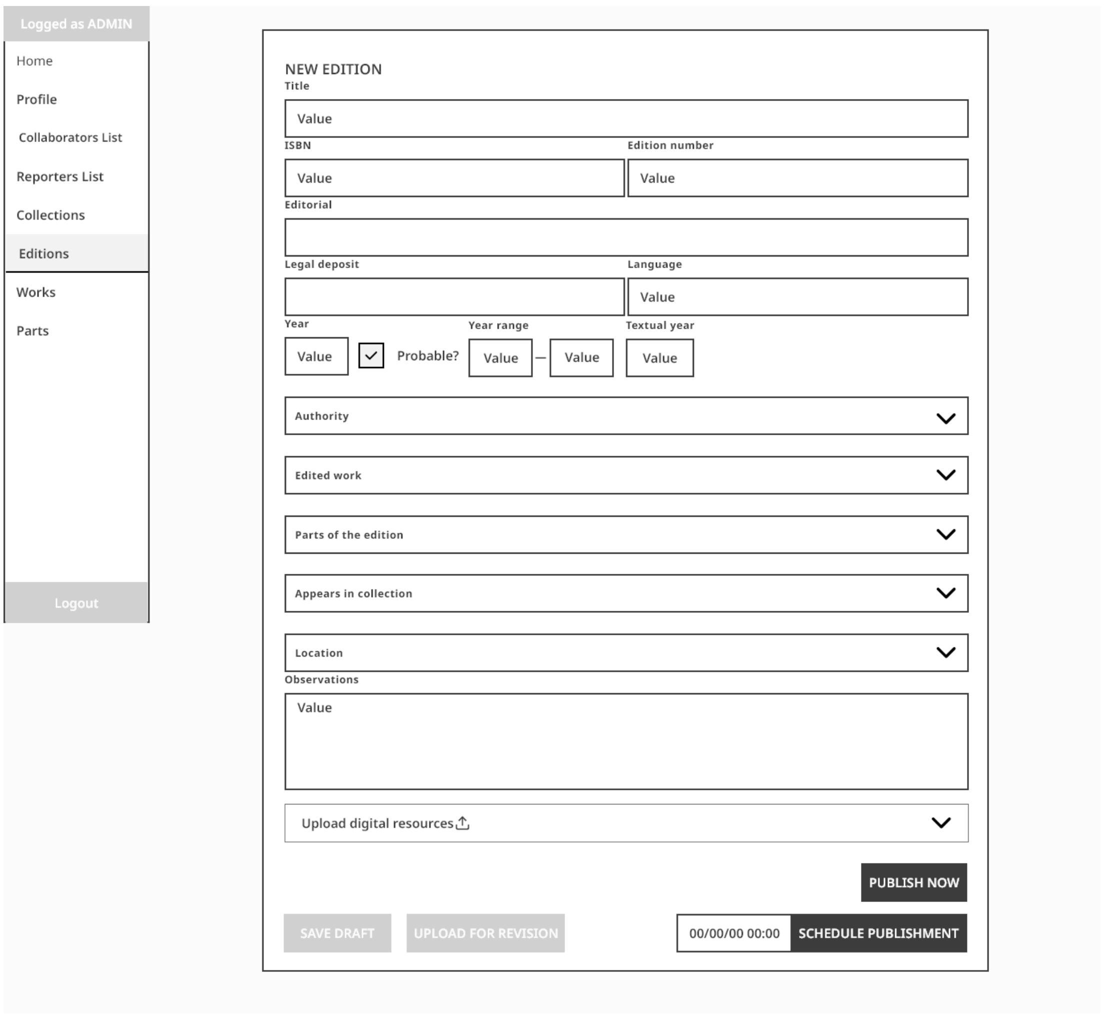
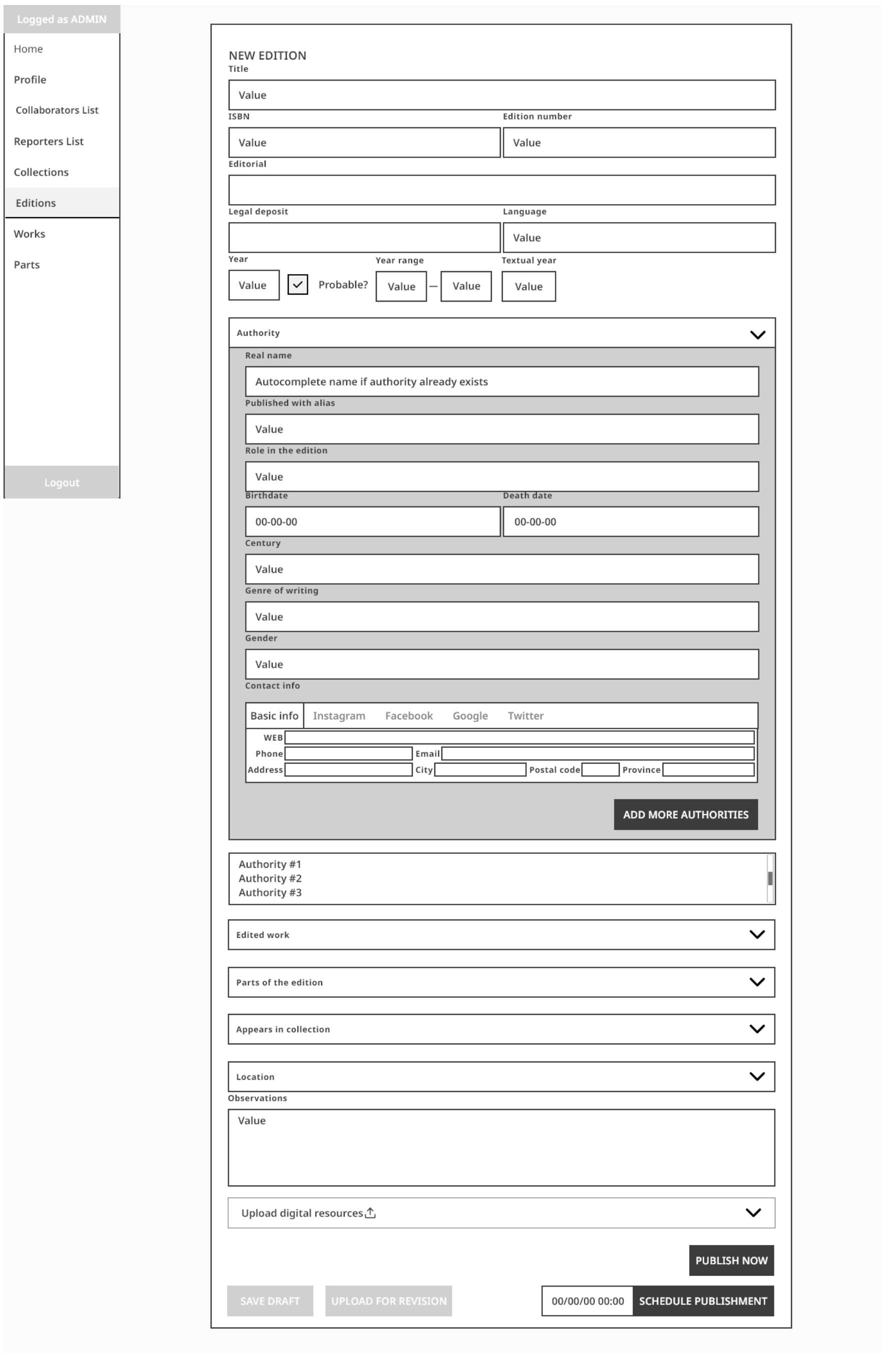


Figure B.22: Admin management - Edition form


Logged as ADMIN

Home
Profile
Collaborators List
Reporters List
Collections
Editions
Works
Parts

Logout

NEW EDITION

Title	Value
ISBN	Value
Edition number	Value
Editorial	
Legal deposit	
Language	Value
Year	Value ✓ Probable?
Year range	Value — Value
Textual year	Value

Authority

Real name	Autocomplete name if authority already exists
Published with alias	Value
Role in the edition	Value
Birthdate	00-00-00
Death date	00-00-00
Century	Value
Genre of writing	Value
Gender	Value

Contact info

Basic info | Instagram | Facebook | Google | Twitter

WEB
Phone
Email
Address
City
Postal code
Province

ADD MORE AUTHORITIES

Authority #1
Authority #2
Authority #3

Edited work

Parts of the edition

Appears in collection

Location

Observations

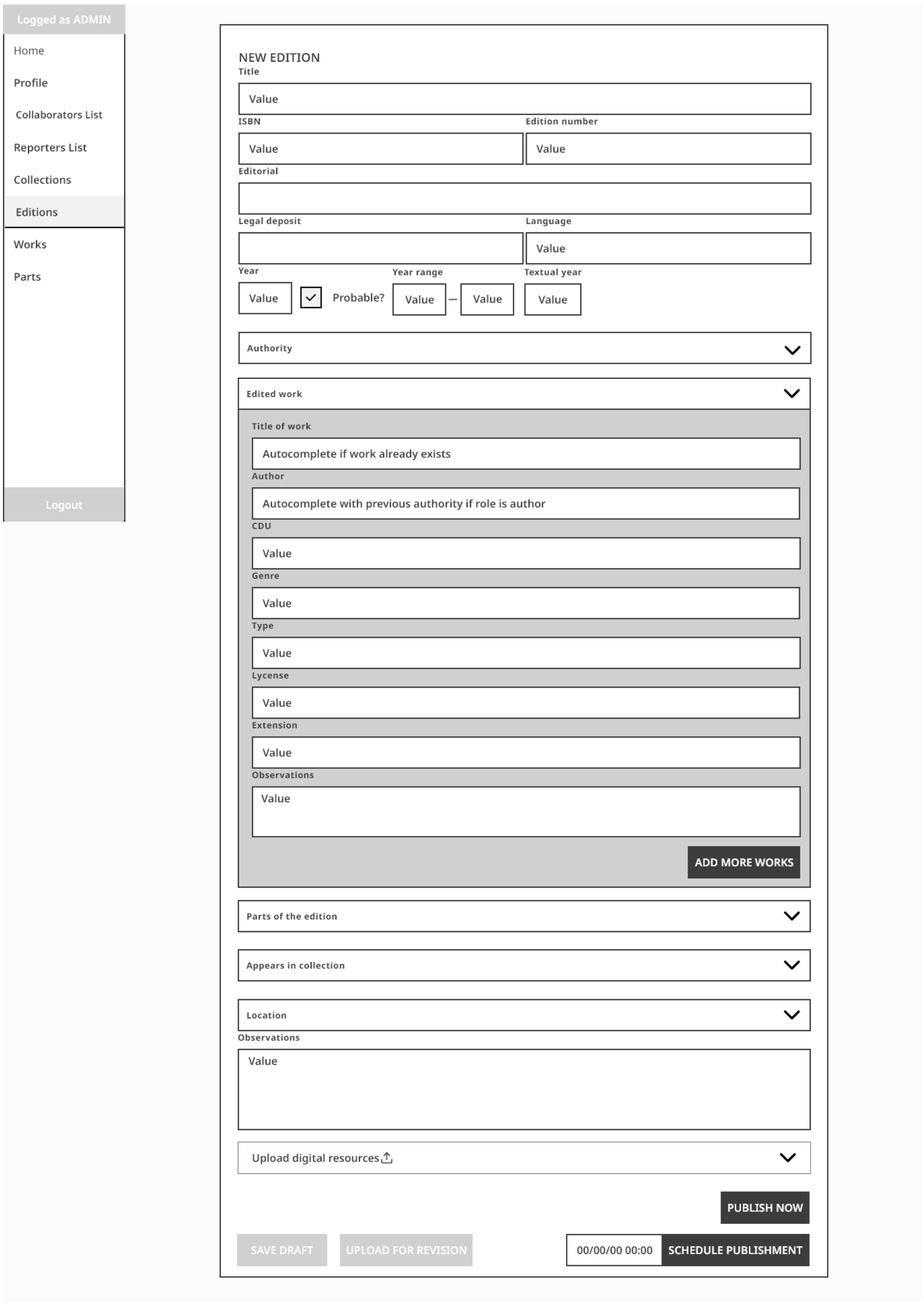
Value

Upload digital resources

PUBLISH NOW

SAVE DRAFT | UPLOAD FOR REVISION | 00/00/00 00:00 | SCHEDULE PUBLISHMENT

Figure B.23: Admin management - Edition form with authority accordion



Logged as ADMIN

- Home
- Profile
- Collaborators List
- Reporters List
- Collections
- Editions
- Works
- Parts

Logout

NEW EDITION

Title	Value
ISBN	Value
Edition number	Value
Editorial	
Legal deposit	
Language	Value
Year	Value ✓ Probable?
Year range	Value — Value
Textual year	Value

Authority

Edited work

Title of work	Autocomplete if work already exists
Author	Autocomplete with previous authority if role is author
CDU	Value
Genre	Value
Type	Value
Lycense	Value
Extension	Value
Observations	Value

ADD MORE WORKS

Parts of the edition

Appears in collection

Location

Observations	Value

Upload digital resources

PUBLISH NOW

SAVE DRAFT UPLOAD FOR REVISION 00/00/00 00:00 SCHEDULE PUBLISHMENT

Figure B.24: Admin management - Edition form with works accordion

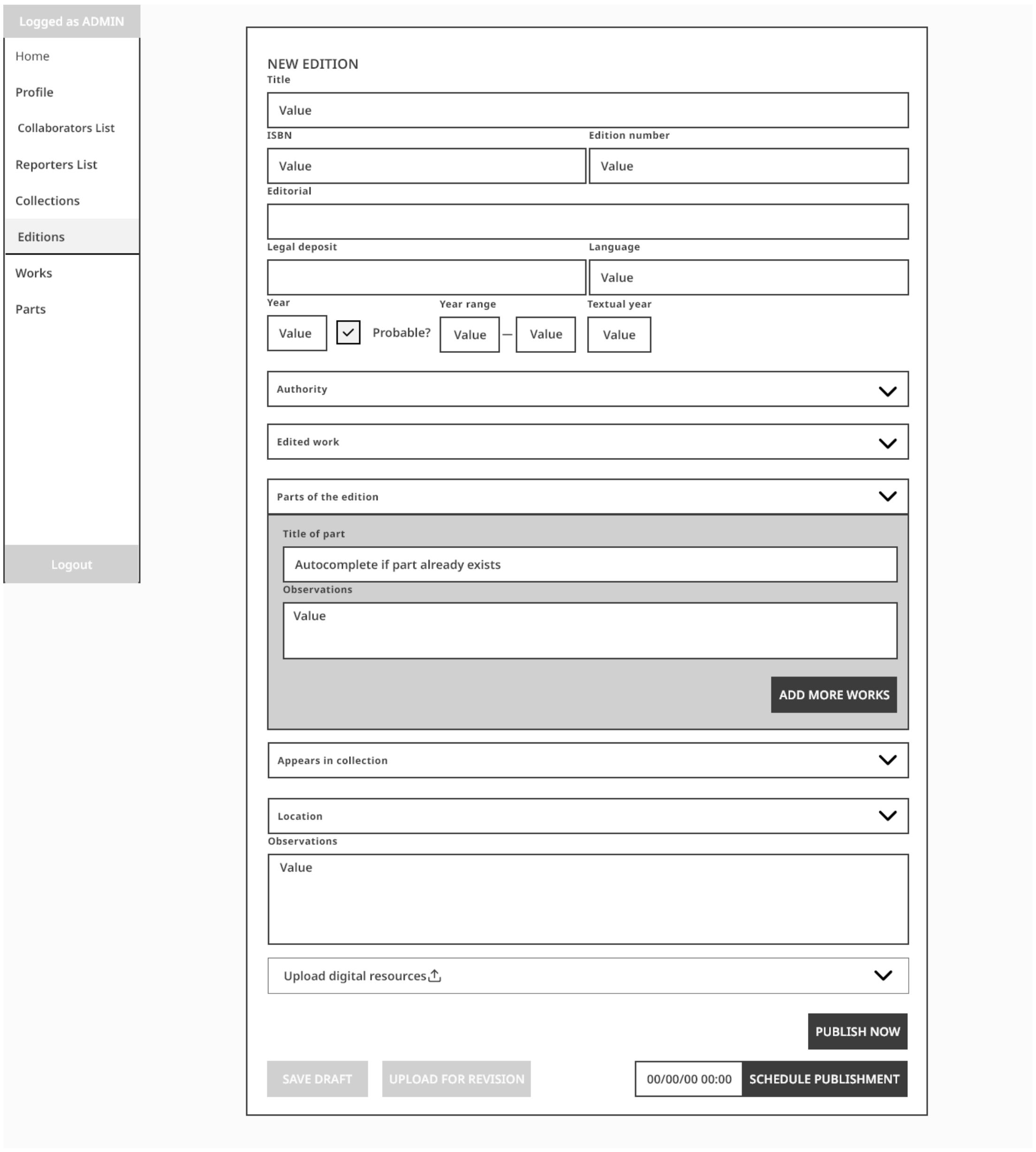
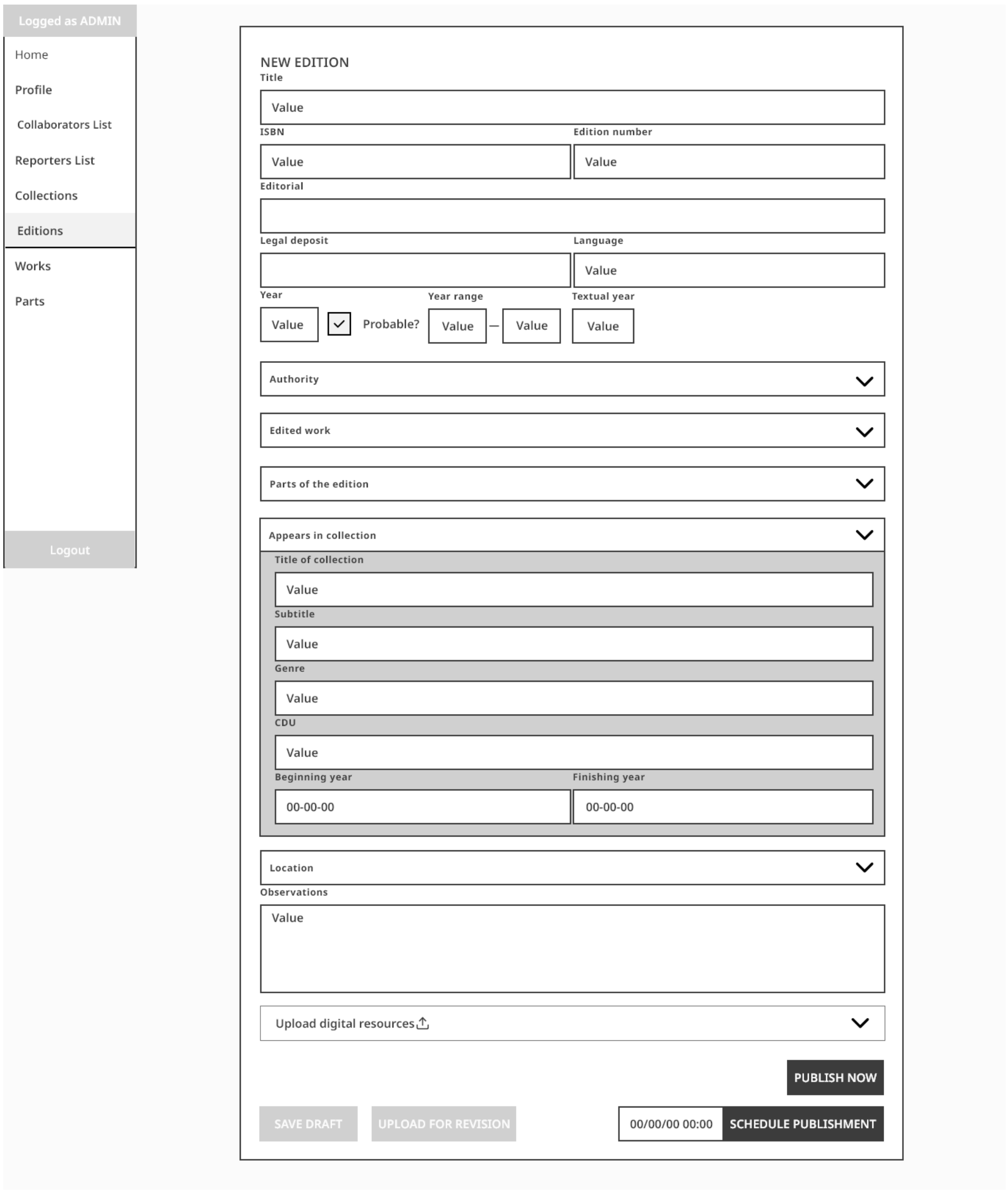


Figure B.25: Admin management - Edition form with parts accordion



NEW EDITION

Title: Value

ISBN: Value

Edition number: Value

Editorial:

Legal deposit:

Language: Value

Year: Value ✓ Probable?

Year range: Value — Value

Textual year: Value

Authority

Edited work

Parts of the edition

Appears in collection

Title of collection: Value

Subtitle: Value

Genre: Value

CDU: Value

Beginning year: 00-00-00

Finishing year: 00-00-00

Location

Observations: Value

Upload digital resources

PUBLISH NOW

SAVE DRAFT | UPLOAD FOR REVISION | 00/00/00 00:00 | SCHEDULE PUBLISHMENT

Figure B.26: Admin management - Edition form with collection accordion

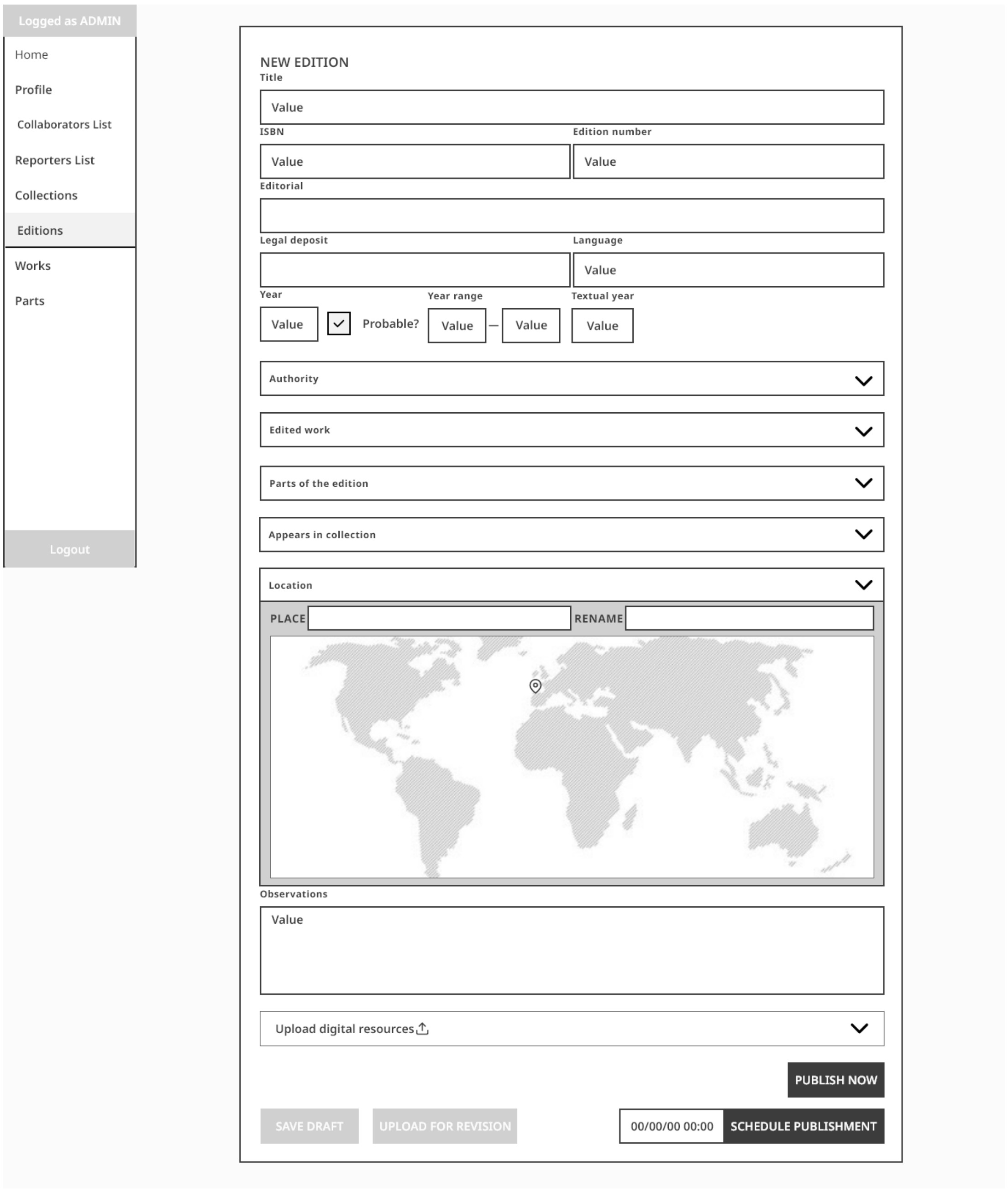


Figure B.27: Admin management - Edition form with location accordion

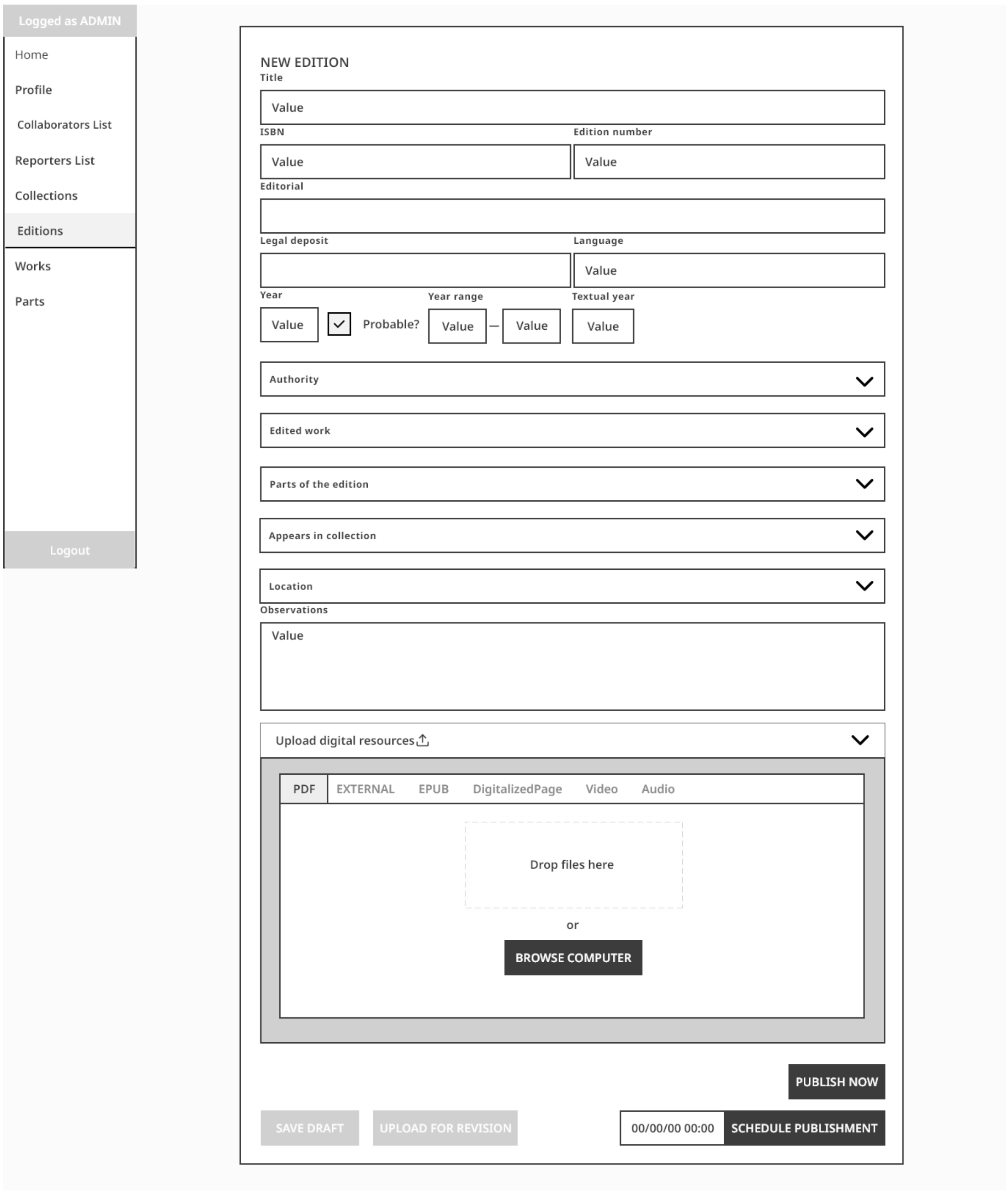


Figure B.28: Admin management - Edition form with digital resources accordion, PDF

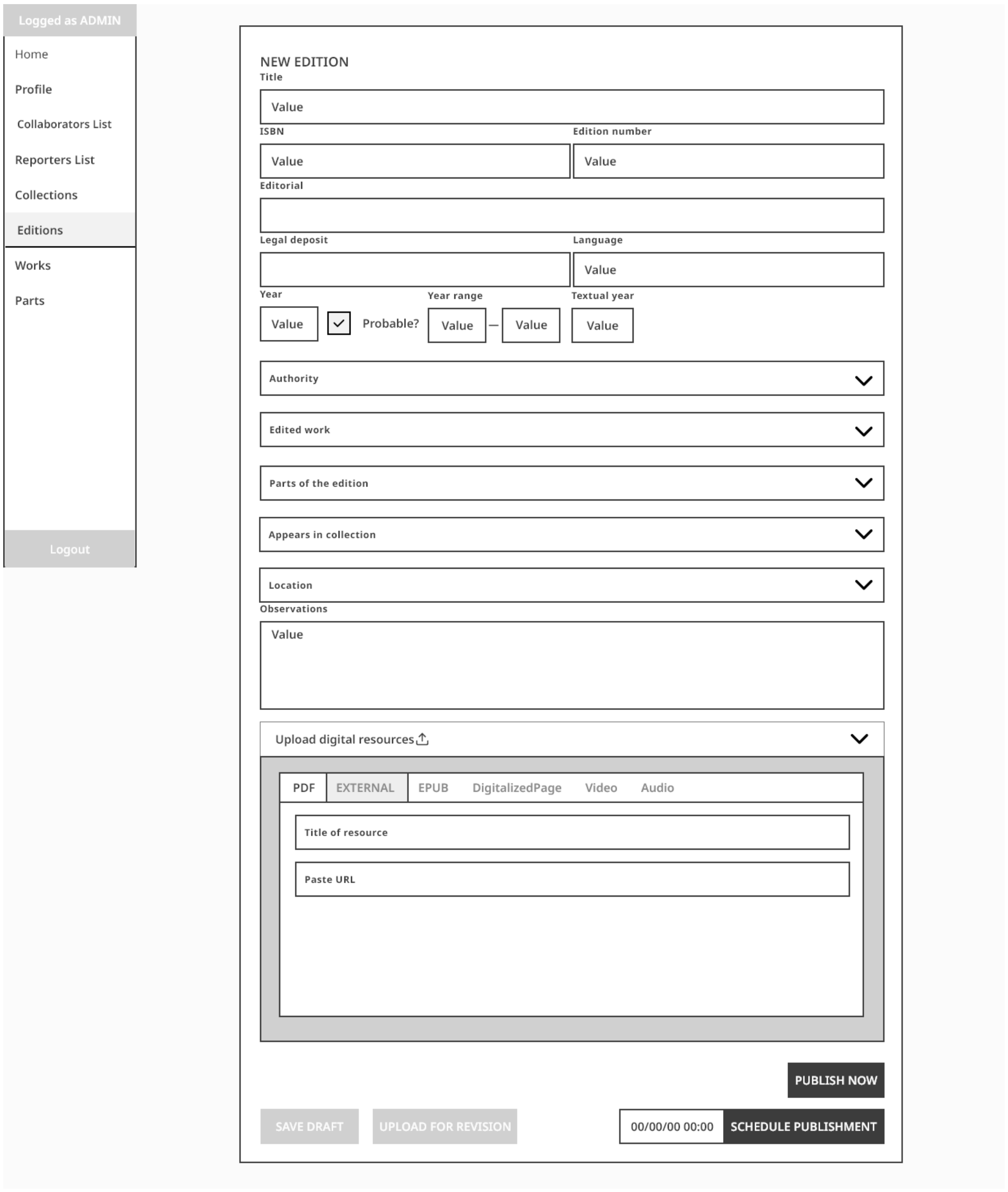


Figure B.29: Admin management - Edition form with digital resources accordion, external files

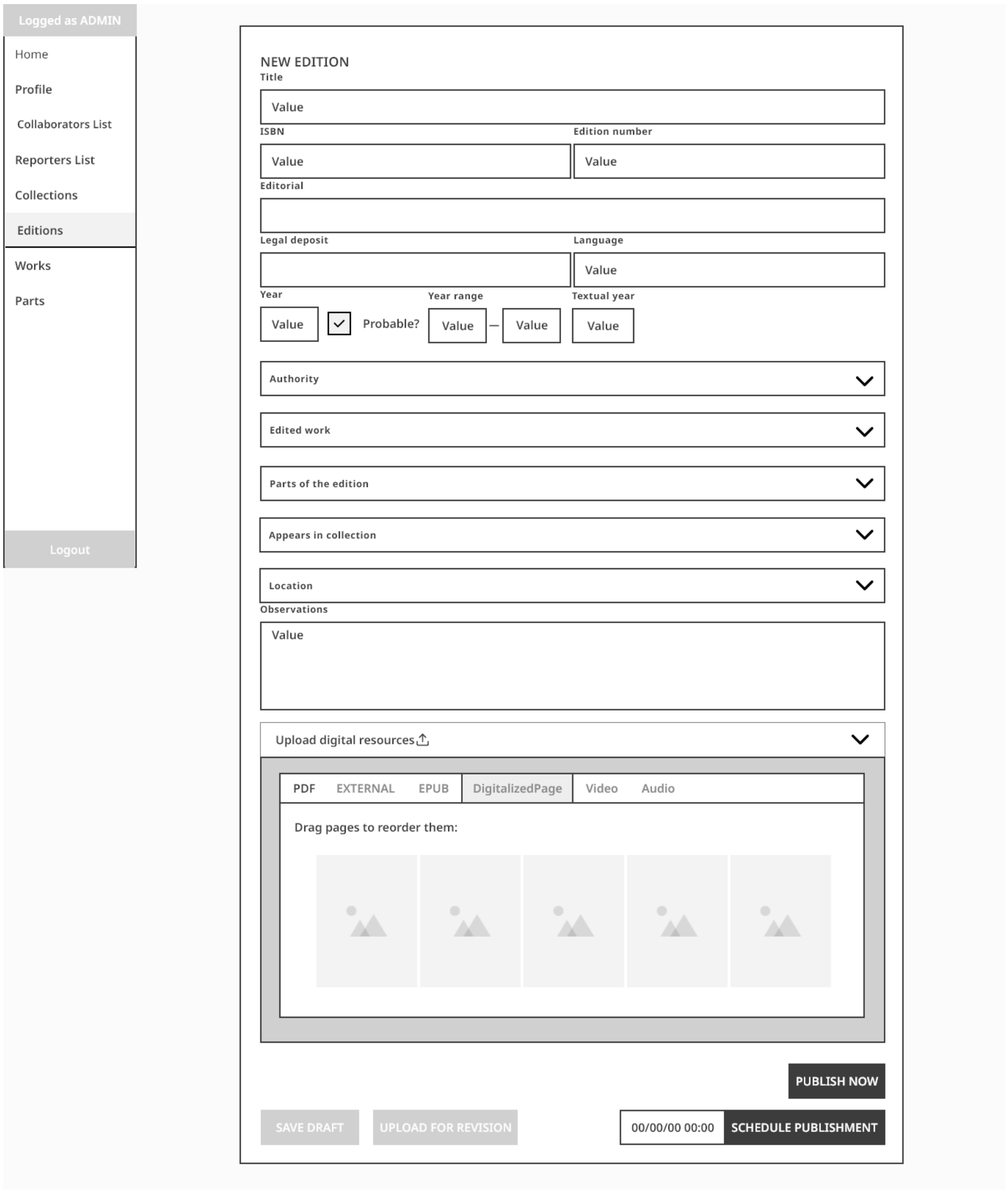
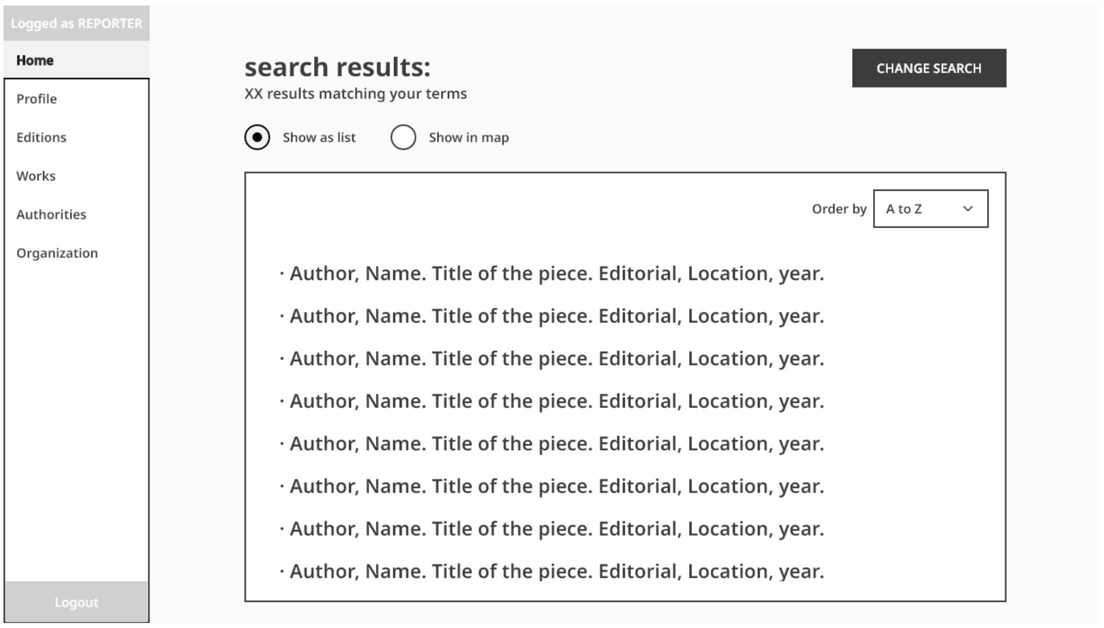


Figure B.30: Admin management - Edition form with digital resources accordion, collection of digitalized pages



· Author, Name. Title of the piece. Editorial, Location, year.
· Author, Name. Title of the piece. Editorial, Location, year.
· Author, Name. Title of the piece. Editorial, Location, year.
· Author, Name. Title of the piece. Editorial, Location, year.
· Author, Name. Title of the piece. Editorial, Location, year.
· Author, Name. Title of the piece. Editorial, Location, year.
· Author, Name. Title of the piece. Editorial, Location, year.
· Author, Name. Title of the piece. Editorial, Location, year.

Figure B.31: Search result as text list

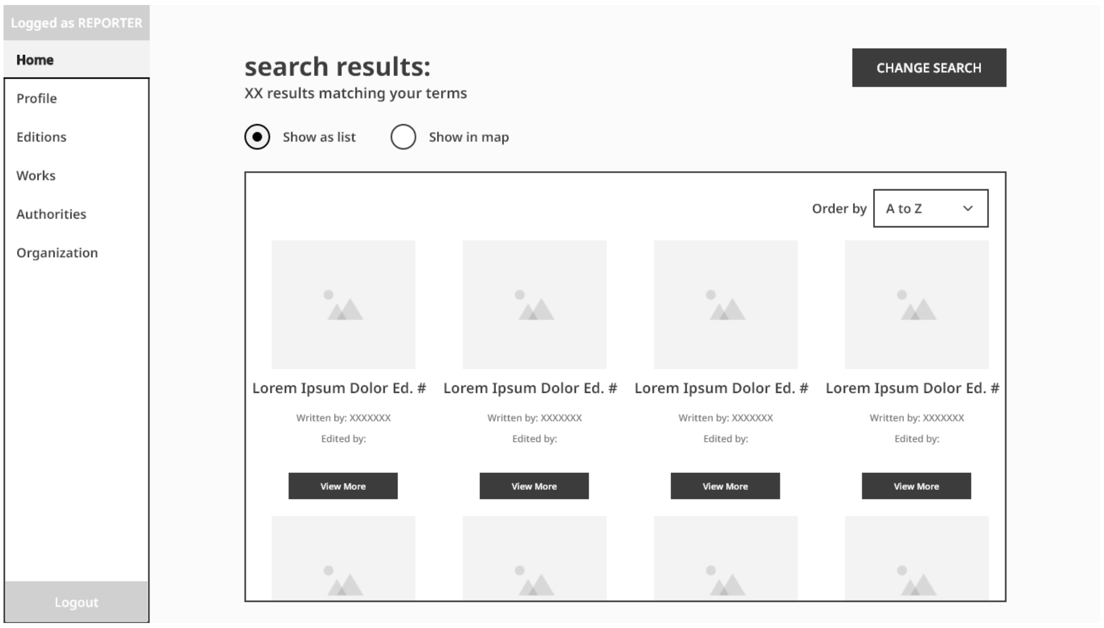


Figure B.32: Search result with images

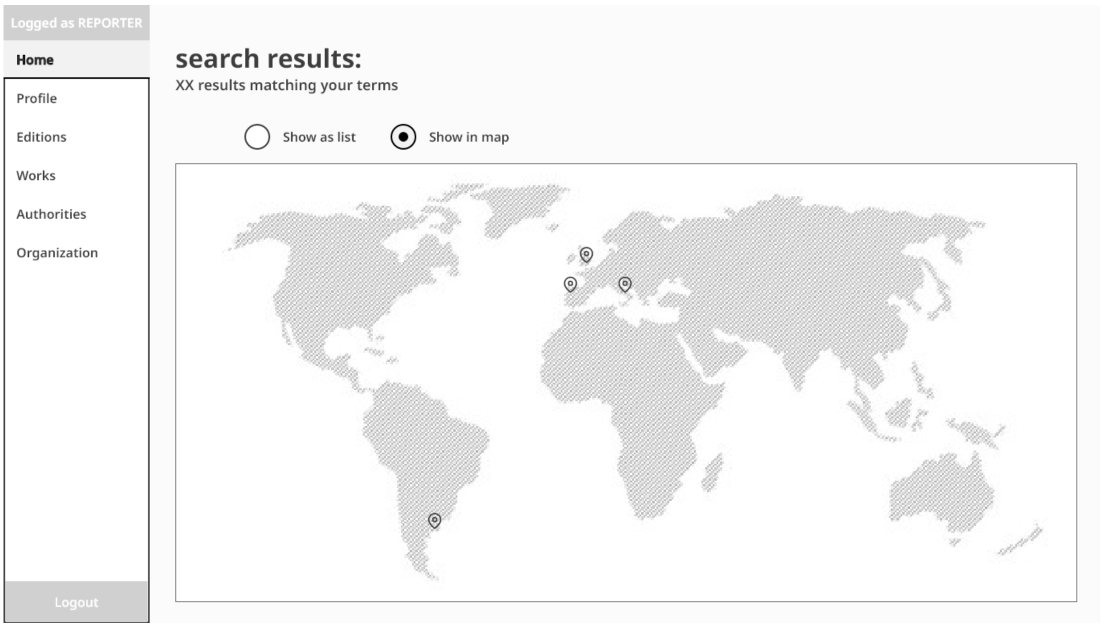
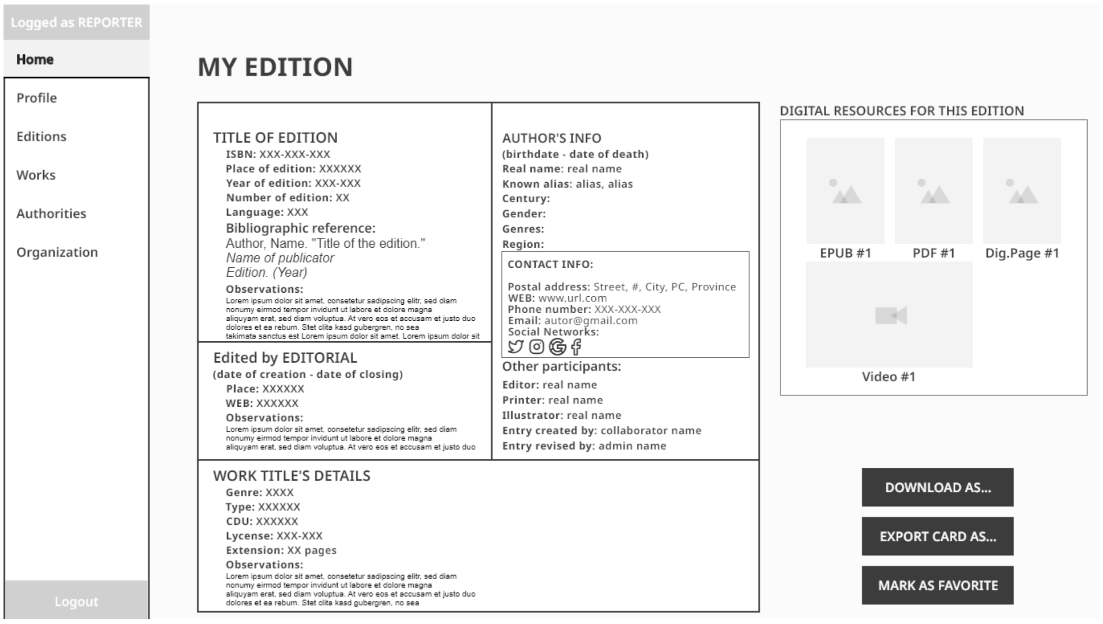


Figure B.33: Search result as map



MY EDITION

TITLE OF EDITION

ISBN: XXX-XXX-XXX
Place of edition: XXXXXX
Year of edition: XXX-XXX
Number of edition: XX
Language: XXX
Bibliographic reference:
Author, Name. "Title of the edition."
Name of publicator
Edition. (Year)
Observations:
Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit

Edited by EDITORIAL

(date of creation - date of closing)
Place: XXXXXX
WEB: XXXXXX
Observations:
Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo

AUTHOR'S INFO

(birthdate - date of death)
Real name: real name
Known alias: alias, alias
Century:
Gender:
Genres:
Region:

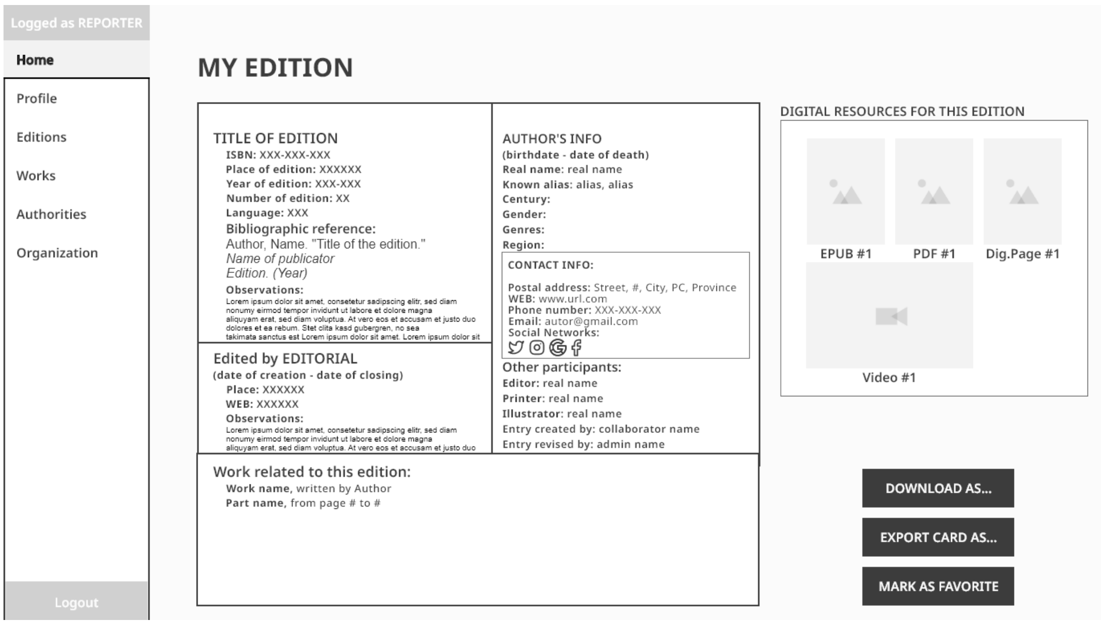
CONTACT INFO:
Postal address: Street, #, City, PC, Province
WEB: www.url.com
Phone number: XXX-XXX-XXX
Email: autor@gmail.com
Social Networks:

Other participants:
Editor: real name
Printer: real name
Illustrator: real name
Entry created by: collaborator name
Entry revised by: admin name

WORK TITLE'S DETAILS

Genre: XXXX
Type: XXXXXX
CDU: XXXXXX
Lycense: XXX-XXX
Extension: XX pages
Observations:
Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea

Figure B.34: Public edition detail with one wokr



MY EDITION

TITLE OF EDITION
ISBN: XXX-XXX-XXX
Place of edition: XXXXXX
Year of edition: XXX-XXX
Number of edition: XX
Language: XXX
Bibliographic reference:
Author, Name. "Title of the edition."
Name of publicator
Edition. (Year)
Observations:
Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit

Edited by EDITORIAL
(date of creation - date of closing)
Place: XXXXXX
WEB: XXXXXX
Observations:
Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo

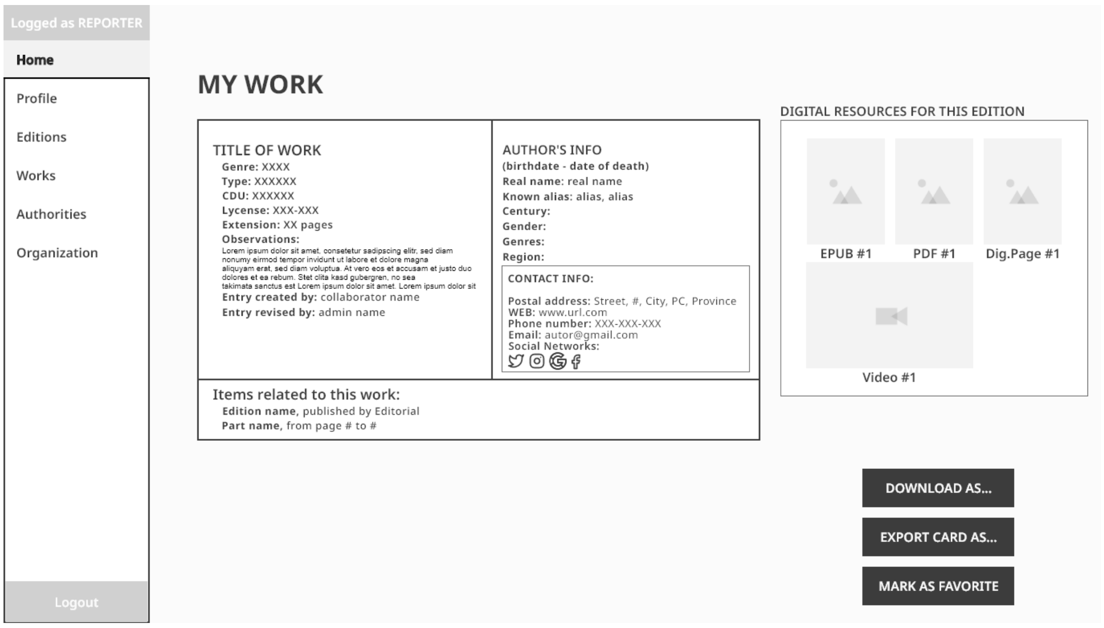
AUTHOR'S INFO
(birthdate - date of death)
Real name: real name
Known alias: alias, alias
Century:
Gender:
Genres:
Region:

CONTACT INFO:
Postal address: Street, #, City, PC, Province
WEB: www.url.com
Phone number: XXX-XXX-XXX
Email: autor@gmail.com
Social Networks:

Other participants:
Editor: real name
Printer: real name
Illustrator: real name
Entry created by: collaborator name
Entry revised by: admin name

Work related to this edition:
Work name, written by Author
Part name, from page # to #

Figure B.35: Public edition detail with several works



MY WORK

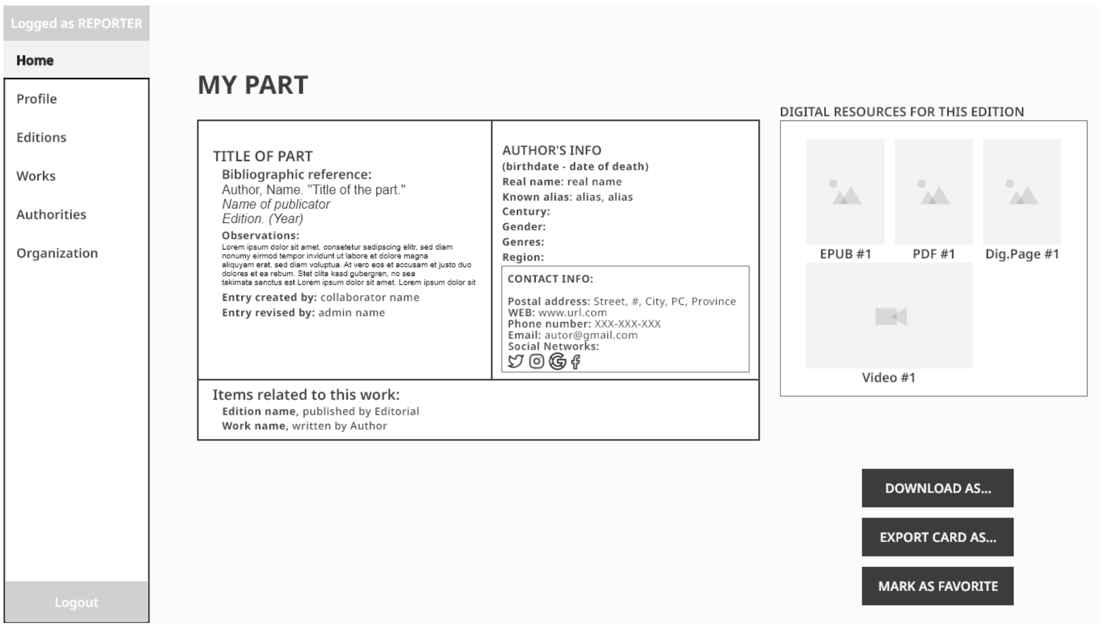
TITLE OF WORK
Genre: XXXX
Type: XXXXXX
CDU: XXXXXX
Lycense: XXX-XXX
Extension: XX pages
Observations:
Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit
Entry created by: collaborator name
Entry revised by: admin name

AUTHOR'S INFO
(birthdate - date of death)
Real name: real name
Known alias: alias, alias
Century:
Gender:
Genres:
Region:

CONTACT INFO:
Postal address: Street, #, City, PC, Province
WEB: www.url.com
Phone number: XXX-XXX-XXX
Email: autor@gmail.com
Social Networks:

Items related to this work:
Edition name, published by Editorial
Part name, from page # to #

Figure B.36: Public work detail with several publications related



Logged as REPORTER

Home

Profile

Editions

Works

Authorities

Organization

Logout

MY PART

TITLE OF PART

Bibliographic reference:
Author, Name. "Title of the part."
Name of publicator
Edition. (Year)

Observations:
Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit

Entry created by: collaborator name
Entry revised by: admin name

AUTHOR'S INFO
(birthdate - date of death)
Real name: real name
Known alias: alias, alias
Century:
Gender:
Genres:
Region:

CONTACT INFO:
Postal address: Street, #, City, PC, Province
WEB: www.url.com
Phone number: XXX-XXX-XXX
Email: autor@gmail.com
Social Networks:

Items related to this work:
Edition name, published by Editorial
Work name, written by Author

DIGITAL RESOURCES FOR THIS EDITION

EPUB #1 PDF #1 Dig.Page #1

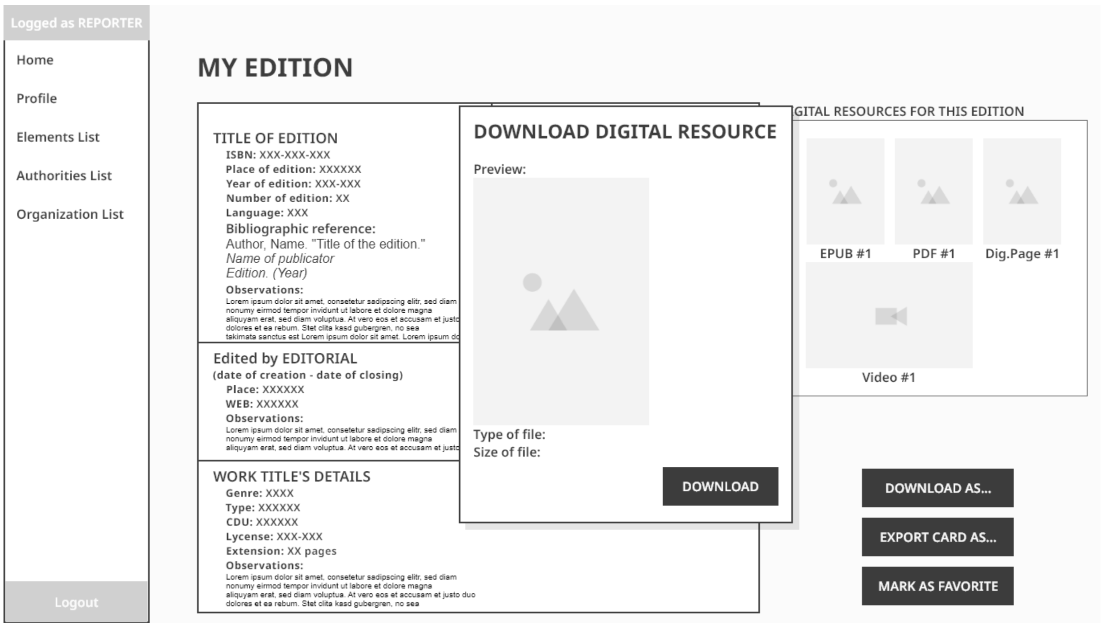
Video #1

DOWNLOAD AS...

EXPORT CARD AS...

MARK AS FAVORITE

Figure B.37: Public part detail



Logged as REPORTER

Home

Profile

Elements List

Authorities List

Organization List

Logout

MY EDITION

TITLE OF EDITION
ISBN: XXX-XXX-XXX
Place of edition: XXXXXX
Year of edition: XXX-XXX
Number of edition: XX
Language: XXX

Bibliographic reference:
Author, Name. "Title of the edition."
Name of publicator
Edition. (Year)

Observations:
Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum do

Edited by EDITORIAL
(date of creation - date of closing)
Place: XXXXXX
WEB: XXXXXX
Observations:
Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo

WORK TITLE'S DETAILS
Genre: XXXX
Type: XXXXXX
CDU: XXXXXX
Lycense: XXX-XXX
Extension: XX pages
Observations:
Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea

DOWNLOAD DIGITAL RESOURCE

Preview:

Type of file:
Size of file:

DOWNLOAD

GITAL RESOURCES FOR THIS EDITION

EPUB #1 PDF #1 Dig.Page #1

Video #1

DOWNLOAD AS...

EXPORT CARD AS...

MARK AS FAVORITE

Figure B.38: Public edition download resource

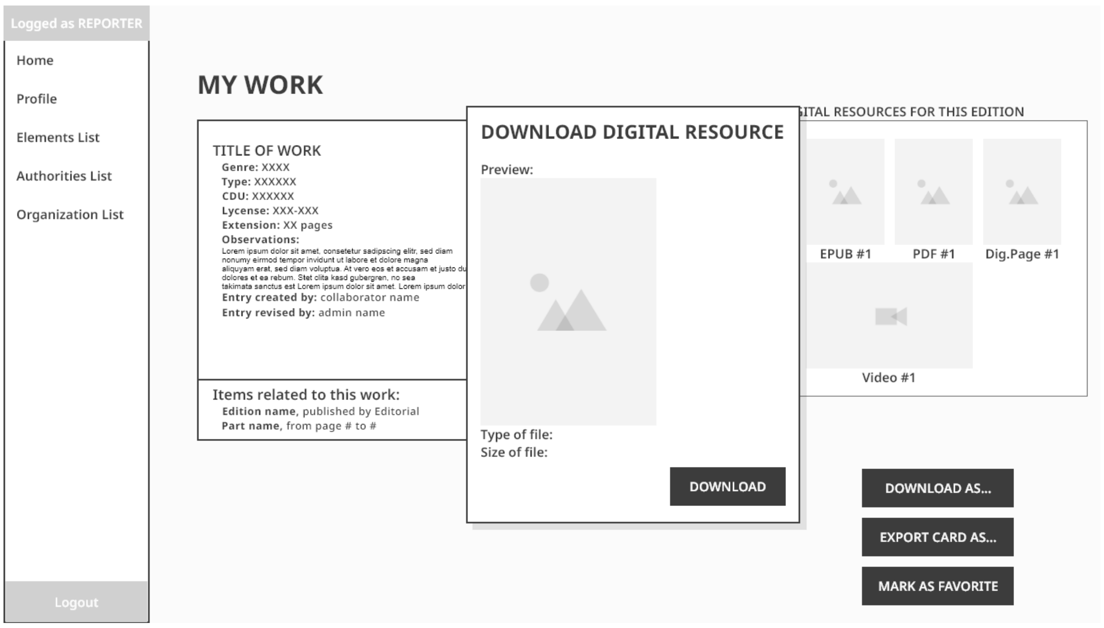


Figure B.39: Public work download resource

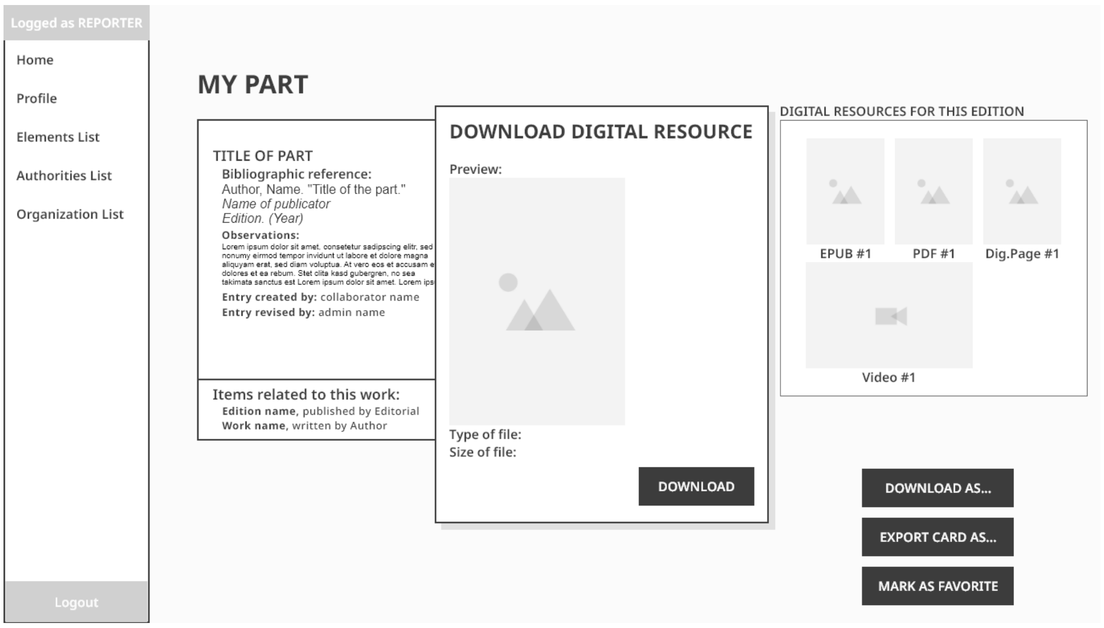
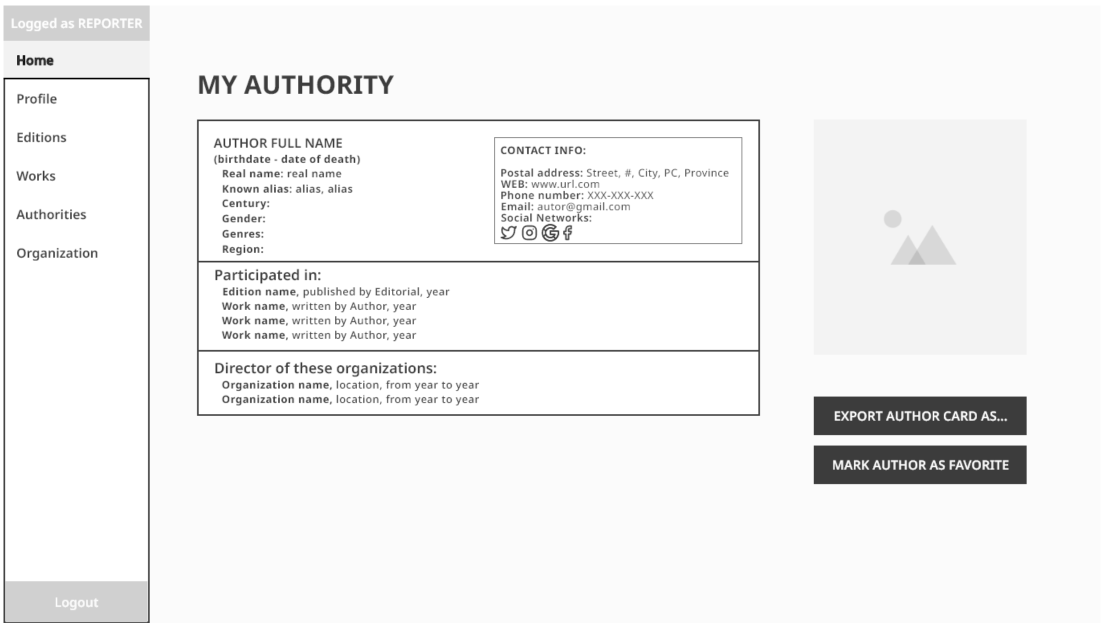


Figure B.40: Public part download resource



MY AUTHORITY

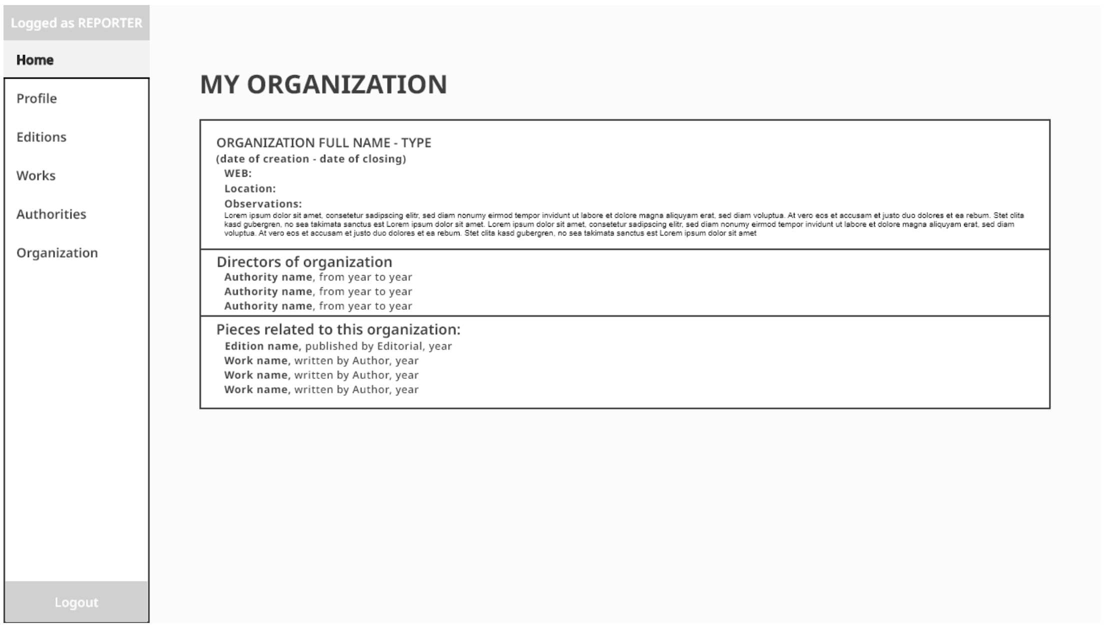
AUTHOR FULL NAME
(birthdate - date of death)
Real name: real name
Known alias: alias, alias
Century:
Gender:
Genres:
Region:

CONTACT INFO:
Postal address: Street, #, City, PC, Province
WEB: www.url.com
Phone number: XXX-XXX-XXX
Email: autor@gmail.com
Social Networks:

Participated in:
Edition name, published by Editorial, year
Work name, written by Author, year
Work name, written by Author, year
Work name, written by Author, year

Director of these organizations:
Organization name, location, from year to year
Organization name, location, from year to year

Figure B.41: Public authority detail



MY ORGANIZATION

ORGANIZATION FULL NAME - TYPE
(date of creation - date of closing)
WEB:
Location:
Observations:
Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet

Directors of organization
Authority name, from year to year
Authority name, from year to year
Authority name, from year to year

Pieces related to this organization:
Edition name, published by Editorial, year
Work name, written by Author, year
Work name, written by Author, year
Work name, written by Author, year

Figure B.42: Public organization detail

Appendix C

Data model for the product

Extended representation of the data model with a close-up for the Digital Resources classes that were not explained in detail.

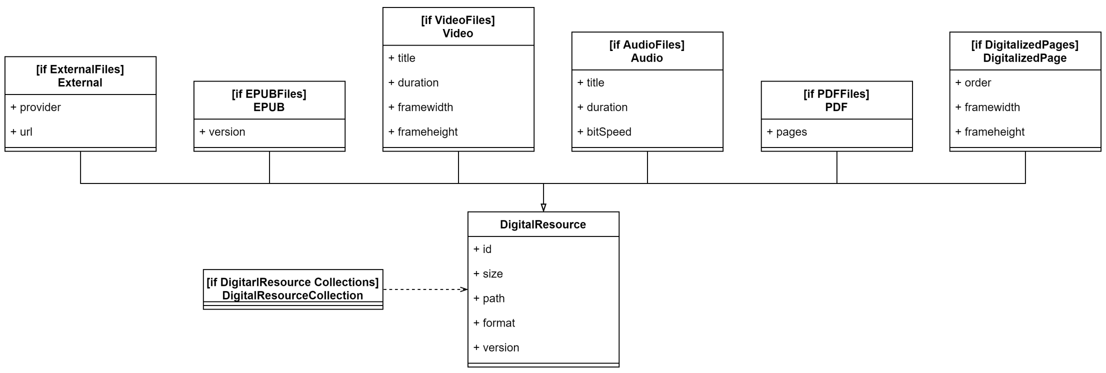


Figure C.1: Digital resources classes

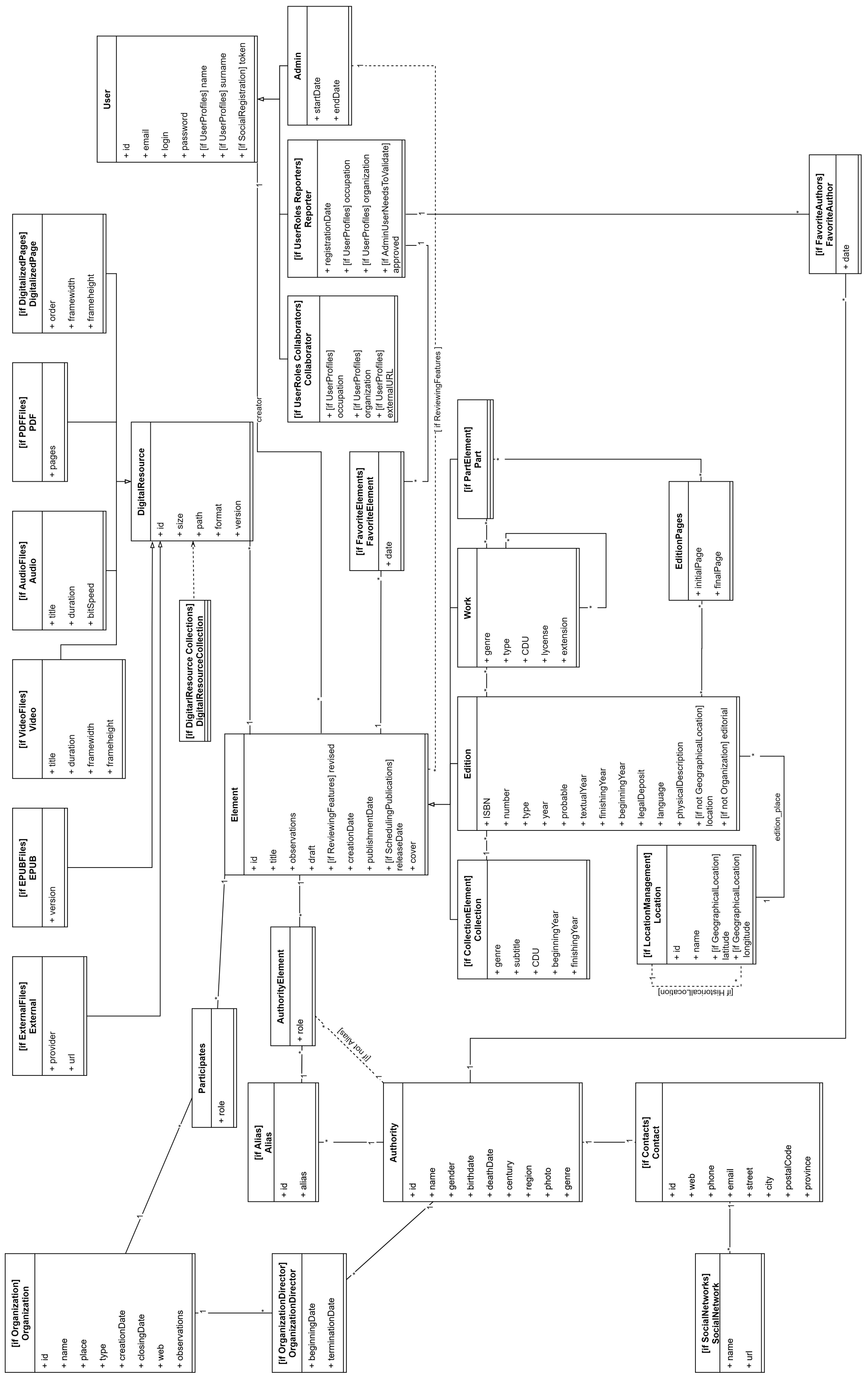


Figure C.2: Digital resources classes

Appendix D

Glosary of Acronyms

AOP *Aspect Oriented Programming.*

API *Application Programming Interface.*

CLI *Command Line Interface.*

CRUD *Create, Read, Update and Delete.*

CSS *Cascading StyleSheets.*

DAO *Data Access Object.*

DBMS *DataBase Management System.*

DOM *Document Object Model.*

DTO *Data Transfer Object.*

FOP *Feature Oriented Programming.*

HTML *HyperText Markup Language.*

HTTP *HyperText Transfer Protocol.*

IDE *Integrated Development Environment.*

JDK *Java Development Kit.*

JPA *Java Persistence API.*

JSON *JavaScript Object Notation.*

JWT *JSON Web Token.*

MVCC	*Multi Version Concurrency Control.*
NPM	*Node Package Manager.*
ORM	*Object Relational Mapping.*
PDF	*Portable Document Format.*
RDBMS	*Relational DataBase Management System.*
REST	*Representational State Transfer.*
SPA	*Single Page Application.*
SPL	*Software Product Line.*
SQL	*Structured Query Language.*
UML	*Unified Modeling Language.*
URL	*Uniform Resource Locator.*
WYSIWYG	*What You See Is What You Get.*
XML	*Extensible Markup language.*

Appendix E

Glossary of terms

Back-end Part of a computer system that is not directly accessed by the user and is responsible for storing and manipulating system data.

Bug Error or issue in a computer program.

Framework Reusable software environment that provides particular functionality to facilitate the development of a computer application.

Front-end Part of a computer system with which the user interacts directly.

Issue Unit of work to complete an improvement in a system.

Plugin Software component that adds an specific feature to a computer program.

Scaffolding Technique consisting on code generation starting from predefined templates and an specification given by the developer.

Scrum Set of practices used in agile project management that emphasize daily communication and the flexible reassessment of plans that are carried out in short, iterative phases of work.

Sprint In Scrum it is a prefixed interval during which a ”Done or Finished” product increment is created.

Bibliography

[1] Biblioteca digital siglo de oro. [Online]. Available: https://www.bidiso.es/index.htm;jsessionid=A5109F67AF0FF912C9573C5B72E0AABE

[2] Symbola, divisas o empresas históricas. [Online]. Available: https://www.bidiso.es/Symbola/

[3] Cbdrs catálogo y biblioteca digital de relaciones de sucesos. [Online]. Available: https://www.bidiso.es/CBDRS/

[4] E. R. L. Carme Fernández Pérez-Sanjulián, Mª Antonia Pérez Rodríguez and Ángeles Saavedra Places, "Recursos para a clasificación da produción editorial na galiza durante a etapa franquista: deseño e alimentación da base de datos," 2014. [Online]. Available: https://www.janusdigital.es/anexos/contribucion.htm?id=15

[5] Biblioteca virtual galega. [Online]. Available: http://bvg.udc.es/

[6] Ahead tool suite. [Online]. Available: https://www.cs.utexas.edu/~schwartz/ATS/fopdocs/

[7] Featurehouse: Language-independent, automated software composition. [Online]. Available: https://www.infosun.fim.uni-passau.de/spl/apel/fh/

[8] The aspectj project. [Online]. Available: https://www.eclipse.org/aspectj/

[9] Antenna: An ant-to-end solution for wireless java. [Online]. Available: http://antenna.sourceforge.net/wtkpreprocess.php

[10] J. M. T. S. B. L. Saake, *Mastering Software Variability with FeatureIDE.* Springer, 2017.

[11] A. C. Álvarez. Github repository for spl-js-engine derivation engine. [Online]. Available: https://github.com/AlexCortinas/spl-js-engine

[12] D. L. from University of A Coruña, "Sistemas de Código Aberto para a creación, mantemento e aproveitamento de Bibliotecas Dixitais: Compresión e Indexación. (Spanish) [Open source system for the creation, maintenance and exploitation of digital libraries]," 2006. [Online]. Available: https://lbd.udc.es/ProjectInformation.do?lang=en_US&slug=TIN2006-15071-C03-03

[13] ——, "Herramientas avanzadas para la implementación de Bibliotecas Digitales. (Spanish) [Advanced tools for the implementation of digital libraries]," 2006. [Online]. Available: https://lbd.udc.es/ProjectInformation.do?lang=en_US&slug=TIN2006-15071-C03-03

[14] Delos network of excellence on digital libraries. [Online]. Available: http://delosw.isti.cnr.it/

[15] Postgresql official website. [Online]. Available: https://www.postgresql.org/

[16] Vue.js official guide. [Online]. Available: https://vuejs.org/v2/guide/

[17] Maven official website. [Online]. Available: https://maven.apache.org/

[18] Node.js official website. [Online]. Available: https://nodejs.org/es/docs/guides/getting-started-guide/

[19] Npm official website. [Online]. Available: https://www.npmjs.com/about

[20] Spring official website. [Online]. Available: https://spring.io/

[21] Spring boot official website. [Online]. Available: https://spring.io/projects/spring-boot

[22] Hibernate official website. [Online]. Available: https://hibernate.org/

[23] D. M. from Zenkit Blog, "Agile software development," 2018. [Online]. Available: https://zenkit.com/en/blog/agile-methodology-an-overview/

[24] Atomate, "Agile methodology: An overview," 2018. [Online]. Available: https://atomate.net/process/agile-software-development/

[25] J. Schwaber, Ken; Sutherland, "The scrum guide: The definitive guide to scrum: The rules of the game," 2017. [Online]. Available: https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf

[26] Gitlab official website. [Online]. Available: https://about.gitlab.com/

[27] Eclipse ide and tools. [Online]. Available: https://www.eclipse.org/ide/

[28] Documentation for visual studio code. [Online]. Available: https://code.visualstudio.com/docs

[29] Adobe launches experience design cc, a new tool for ux designers. [Online]. Available: https://web.archive.org/web/20171020041101/https://techcrunch.com/2016/03/14/adobe-launches-experience-design-cc-a-new-tool-for-ux-designers/

[30] Draw.io by diagrams.net. [Online]. Available: https://app.diagrams.net/

[31] Staruml official documentation. [Online]. Available: https://docs.staruml.io/

[32] Overleaf documentation. [Online]. Available: https://www.overleaf.com/learn/latex/Main_Page

[33] S. E. I. from Carnegie Mellon University, "Software product lines collection," 2014. [Online]. Available: https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=513819

[34] N. R. Brisaboa, A. Cortiñas, M. R. Luaces, and O. Pedreira, "Aplicando scaffolding en el desarrollo de Líneas de Producto Software. (Spanish) [Applying scaffolding to the development of software product lines]," in *Actas de las XXI Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2016)*, Salamanca, 2016, pp. 23–36.

[35] E. R. Frank J. van der Linden, Klaus Schmid, *Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering*. Springer, 2007.

[36] I. Sommerville, *Software Engineering*, 10th ed. Pearson, 2016.

[37] Spring security. [Online]. Available: https://spring.io/projects/spring-security

[38] Vue-router guide. [Online]. Available: https://router.vuejs.org/

[39] Vue-i18n, internationalization plugin for vue.js. [Online]. Available: https://kazupon.github.io/vue-i18n/

[40] Axios, romise based http client for the browser and node.js. [Online]. Available: https://github.com/axios/axios

[41] Ckeditor 5. [Online]. Available: https://ckeditor.com/ckeditor-5/

[42] Ckeditor 5 documentation. [Online]. Available: https://ckeditor.com/docs/ckeditor5/latest/index.html

[43] Aracne, red de humanidades digitales y letras hispánicas. [Online]. Available: http://www.red-aracne.es/presentacion