



Development of a mobile platform to create and manage digital business cards

Dissertation submitted for the *Grau en Enginyeria Informàtica*

María Florencia Tarditti

28th of October 2016

Carles Farré Tost - ESSI

Software Engineering

Facultat d'Informàtica de Barcelona (FIB)

Universitat Politècnica de Catalunya (UPC) - BarcelonaTech

Abstract

With the raising number of student entrepreneurs and groups of individuals creating start-ups, finding funding for a company is becoming increasingly difficult. The Digital Business Cards project aims to help entrepreneurs exchange contact information without having to rely on expensive physical business cards.

The application consists of a mobile platform that allows users to freely create and manage business cards allowing easy sharing and making cards accessible to everyone, regardless of whether they have an account or not.

Resumen

Con el creciente número de estudiantes emprendedores y grupos de individuos creando empresas, encontrar financiación se está convirtiendo cada vez más difícil. El proyecto tarjetas de contacto digitales tiene como objetivo ayudar a emprendedores a intercambiar información de contacto evitando la dependencia en las costosas tarjetas de visita tradicionales.

La aplicación consiste en una plataforma móvil que permite a los usuarios crear y gestionar tarjetas de forma gratuita, permitiendo compartirlas con facilidad y con cualquiera, independientemente de si tienen una cuenta con la aplicación o no.

Resum

Amb el creixent nombre d'estudiants emprenedors i grups d'individus creant empreses, trobar finançament s'està convertint en una tasca cada vegada més difícil. El projecte targetes de contacte digitals té com a objectiu ajudar a emprenedors a intercanviar informació de contacte evitant haver de dependre de les costoses targetes de visita tradicionals.

L'aplicació consisteix en una plataforma mòbil que permet als usuaris crear i gestionar targetes de forma gratuïta, permetent compartir-les amb facilitat i amb qualsevol, independentment de si tenen una compta amb l'aplicació o no.

Table of contents

1 Context	10
1.1 Introduction	10
1.2 Stakeholders	10
1.2.1 Users	10
1.2.2 Business card printing companies	10
1.2.3 Other digital business card applications	11
1.2.4 Project director	11
1.2.5 Project development team	11
2 Formulation of the problem	12
2.1 Main objective	12
2.2 Specific objectives	12
3 Scope	13
3.1 Obstacles	13
3.1.1 Time constraint	13
3.1.2 Lack of experience with the technologies	13
3.1.3 Obstacles with the synchronization	13
3.1.4 Competitors	14
4 State of the art	15
4.1 Contextualization	15
4.2 Market research	15
4.2.1 Existing applications	15
4.2.2 Technologies available	18
4.3 Market research conclusions	20
5 Methodology	21
5.1 Agile methodologies	21
5.2 Scrum	22
5.3 Monitoring tools	23
5.4 Validation methods	23

6 Requirements	24
6.1 Functional requirements	24
6.2 Usability requirements	25
6.3 Performance and scalability requirements	26
6.4 Political and cultural requirements	26
7 Planning	27
7.1 Phases	27
7.1.1 Project planning	27
7.1.2 Software design and analysis	27
7.1.3 Software implementation	28
7.1.4 Defense preparation	29
7.2 Gantt diagram	30
7.3 Alternatives and action plan	32
7.3.1 Estimated time	32
7.3.2 Action plan	32
7.3.3 Resource consumption	32
7.3.4 Alternative solutions	33
7.3.5 Current state of the project	33
7.3.6 Changes from the original planification	34
8 Budget	35
8.1 Identification of costs	35
8.2 Estimation of costs	35
8.2.1 Human resources	35
8.2.2 Hardware resources	36
8.2.3 Software resources	36
8.2.4 Other expenses	37
8.2.5 Deviations	38
8.2.6 Contingencies	38
8.2.7 Total cost	38
8.3 Control management	39
8.4 Final revision and cost deviations	40
8.5 Return on investment	40
9 Sustainability	43

9.1 Economic sustainability	43
9.2 Social sustainability	43
9.3 Environmental sustainability	43
9.4 Sustainability matrix	44
10 Design	45
10.1 Architecture design	45
10.2 Database design	45
10.3 Server design	47
10.4 Client design	50
10.4.1 AngularJS and the MVC pattern	51
10.5 Testing	52
11 Cross-cutting concerns	55
11.1 Security	55
11.2 User authentication	57
12 Development	58
12.1 Scrum sprints	58
12.1.1 Sprint planning	58
12.1.2 Development of the user stories	58
12.1.3 Daily Scrum meeting	59
12.1.4 Retrospective	59
12.2 Project retrospective	59
13 Laws and regulations	62
13.1 Identification	62
13.1.1 Treatment of data	62
13.1.2 Using free software	62
14 Conclusions	64
14.1 Acquired knowledge	64
14.2 Future development	65
14.3 Technical skills	65
References	67

Acronyms	71
Glossary	72
Appendix A - User stories	73
Appendix B - User manual	76

Table of figures

Figure 1. Scrum methodology phases	22
Figure 2. Gantt diagram	30
Figure 3. Project estimated times	32
Figure 4. Resource consumption	32
Figure 5. Human resources	35
Figure 6. Hardware resources	36
Figure 7. Software resources	37
Figure 8. Other expenses	38
Figure 9. Total costs of developing the project	39
Figure 10. Mobile phone spending statistics	41
Figure 11. Analysis of the expected revenue per application sale	41
Figure 12. Analysis of the sustainability through the sustainability matrix	44
Figure 13. Platform architecture and interaction between different components	45
Figure 14. UML representation of the database	46
Figure 15. Packages used in the development	48
Figure 16. API endpoints	49
Figure 17. The model-view-controller pattern	52
Figure 18. Scrum development velocity chart	60
Figure 19. Scrum burndown chart	61
Figure 20. Application Login and reset password screens	76
Figure 21. Personal cards view and card creation screen	77
Figure 22. Card expanded view and sharing card notice	78
Figure 23. Navigational menu	79
Figure 24. Group creation and viewing	80
Figure 25. Inbox view with any pending invitations	81
Figure 26. User account settings	82
Figure 27. Card renderer shown on different devices	83

1 Context

1.1 Introduction

This is a final year project for the software engineering specialisation in the Faculty of Computer Engineering at the Polytechnic University of Catalonia [1]. The main objective of the project is to create an application that allows users to create, manage and share business cards easily, without the cost of printing them. The idea for the project comes from the realisation that the number of startups is raising and obtaining funding is becoming increasingly difficult, making it necessary to create cheaper ways for entrepreneurs to realize their ideas.

1.2 Stakeholders

Stakeholders are the people, organizations or systems that can affect or be affected by the project's actions, objectives or policies [2]. All stakeholders in a project are interested in fulfilling their objectives, either because it's going to benefit them economically or functionally or because they're trying to minimise the negative effects that something will have on them. The following list specifies our project's identified stakeholders:

1.2.1 Users

The users of the application are entrepreneurs, business owners or employees who are interested in making their own personal business cards or storing those shared by others. They will be interested in the features included in the application as that will determine its usefulness. During development, users will be interested in giving feedback on the developed functionalities in order to suggest changes or give their point of view as users.

1.2.2 Business card printing companies

Companies that print business cards will be interested in the application to see what it offers and use that information to innovate and improve their own products. By doing this, other companies will look to maintain customers and minimise the negative effects that the appearance of new competitors can have on them.

1.2.3 Other digital business card applications

Digital business cards applications already in the market will be interested in the launch of the new product as the appearance of new competitors makes them vulnerable to losing customers or usage shares. Moreover, existing applications will be interested in trying out new ones in order to learn what competitors are doing, analyze threats and use the information to update their own products.

1.2.4 Project director

The project director is interested in the development of the project because he's in charge of supervising it and it is in his best interest for the project to be successful. The director will also be interested in monitoring the progress in order to minimise the effects of deviations or falling behind.

1.2.5 Project development team

The team developing the project is made of a project director, an analyst, a designer, a programmer and a tester. The successfulness of the project depends on them and how well they fulfill their responsibilities. It is the development team's responsibility to meet goals and develop the project in the established time and with the established resources. It is important to specify, however, that in this project one person will be performing all of these roles.

2 Formulation of the problem

With the rising number of companies that are being founded each day with little to no capital, it is increasingly hard for entrepreneurs to get funding. In an attempt to help businesses save money, we decided to design a new way for entrepreneurs to exchange contact information without having to resort to expensive business cards.

After analyzing the needs of users and the different solutions available in the market, we decided that even though there are different applications that fulfill a variety of user needs, there isn't one that successfully groups together everything the user wants.

2.1 Main objective

The main objective of the project can be subdivided into three key ideas:

- We want to create a platform that allows users to easily create a personal business card from zero and without a cost.
- We want users to be able update their business cards at any point in time to ensure that contacts always have access to the correct information - allowing users to update cards shared in the past.
- We want sharing cards to be easy so no user should be forced to have to be a member of the application in order to view a card.

2.2 Specific objectives

In order to achieve our main objectives, we should follow more specific ones:

- Allow users to create cards from zero by filling out the required information inside the application or scanning a physical card and automatically importing the fields on it.
- Allow users to edit or delete cards in a simple and fast way.
- Allow users to share cards without forcing people to have to create an account in the application or in other words, allow cards to be viewed in any browser.
- Make sharing easy by providing different sharing methods that don't require having information about the user with whom we want to share our card.
- Verify user card emails to avoid fraud.
- Allow users to create groups to organize their cards.

3 Scope

The project has three clearly identified components, each of them offering a specific functionality:

- A server that will double as an API, handling business logic and writing and fetching data from the database.
- An application that will provide the user with functionalities such as creating, editing and deleting cards or simply organizing them. Even though a future version of the application should allow offline interaction with data, this functionality will not be included in the scope of this final year project.
- A web application that will act as a card viewer so that users that are not members of the application can view cards shared with them.

3.1 Obstacles

During the development of the project we faced a series of obstacles some of which led to deviations:

3.1.1 Time constraint

While developing the project we realized that some of the tasks took more time than we had initially planned. This led to tasks being set back and the project taking more time than expected. We also realized that we didn't end up having the availability that we initially thought we would have meaning that tasks were stretched out making it impossible to finish the project within the initial deadline.

3.1.2 Lack of experience with the technologies

The speed of development was delayed by the time spent analyzing technologies and learning how to use them. We tested a variety of databases looking for one that would meet our requirements and found that we were spending a lot of time attempting to learn things that in the end weren't used in the project.

3.1.3 Obstacles with the synchronization

Developing the synchronization between the local device and the server proved to be harder than expected. We dedicated time into looking for a database that would automatically provide

us with this feature but seeing the task was too time consuming, decided to provide a limited version of this functionality.

3.1.4 Competitors

Competitors improving and adding features to their application while ours is still in development will lower the attractiveness of our product, reducing our probability of success.

4 State of the art

4.1 Contextualization

Each year there's 472 million entrepreneurs attempting to start 305 million companies worldwide [3]. These companies are startups, most of which have little to no capital to invest in self promotion such as printing business cards for its employees. However, the success of these companies depends on their visibility and how they are able to surpass its competitors, making it essential for businesses and employees to network with potential partners, talent acquisitions and investors.

A good way to keep in touch with business contacts is by exchanging personal cards, however, traditional cards are bulky, expensive and prone to getting lost so we decided to develop a digital alternative.

4.2 Market research

4.2.1 Existing applications

To plan the project and decide what functionalities it should include, we first need to study the different solutions that already exist and are present in the market. We should also review what users say about each of the applications in order to decide whether they like and find each features useful.

We decided to analyze three major applications that are different from each other and form a good representation of the current digital business card market. We studied key features that we consider essential and that could be observed in each of the applications. The results of the analysis were the following:

CamCard ^{[4][5]}		
Reviews 4.3 (68,804 total)	Installs 5,000,000- 10,000,000	Last update February 26, 2016
Offered by INTSIG Information Co.,Ltd	Size Varies	Android required version Varies

Free Yes	In-app purchases/Paid version Yes (In-app purchases 2-23€ and paid version 12€)	Other devices Yes (iPhone)
Feature	Evaluation	
Card creation	Does not allow form filling card creation. Only creates cards from the information in physical scanned cards.	
Card content verification	No.	
Templates	One.	
Privacy	All cards are private.	
Viewing and organising cards	Cards can be organised into custom groups made by the user.	
Sharing cards	Plain text compatible applications and QR code.	
Scanning cards	Yes.	
Other features		
<ul style="list-style-type: none">- Has a card radar to find people nearby.- Allows creating and joining private groups.		
User feedback		
<ul style="list-style-type: none">- Positive feedback regarding card grouping.- Some users requested creating nested groups.		

Inigo Cards^{[6][7]}		
Reviews 4.1 (1,131 total)	Installs 100,000- 500,000	Last update November 17, 2015
Offered by Inigo, LLC	Size 13M	Android required version 4.0.3 and up
Free Yes	In-app purchases/Paid version Yes (1-12€)	Other devices Yes (iPhone, web)
Feature	Evaluation	

Card creation	Allows you to create unlimited cards. Phone, multiple emails, home/work address and social networks.
Card content verification	No.
Templates	One.
Privacy	All cards are private unless shared.
Viewing and organising cards	Does not allow group creation.
Sharing cards	Cards are shared via a link to a web viewer. Supports all applications that allow sending text (a link), QR code scanning and NFC.
Scanning cards	No.
Other features	
- Offers card analytics for paying users.	
User feedback	
<ul style="list-style-type: none"> - Inaccurate analytics. - Limited free customizations. 	

Haystack^{[8][9]}		
Reviews 4.0 (657 total)	Installs 50,000- 100,000	Last update February 15, 2016
Offered by Haystack	Size 9.1M	Android required version 4.0.3 and up
Free Yes	In-app purchases/Paid version No	Other devices Yes (iPhone, web)
Feature	Evaluation	
Card creation	You can only create one card per email address and cards cannot be created unless an email is specified. Information includes phone, email, home/work address and social networks.	
Card content verification	Email has to be verified in order to create a card.	

Templates	One.
Privacy	All cards are private unless shared.
Viewing and organising cards	Does not allow group creation. Cards are shown in a list that can be ordered alphabetically, by added or by updated date.
Sharing cards	Cards are shared via a link to a web viewer. Supports all applications that allow sending text.
Scanning cards	Yes.
Other features	
<ul style="list-style-type: none"> - Launched a web platform offering full functionalities. 	
User feedback	
<ul style="list-style-type: none"> - Requests to share via QR code. - Complaints regarding scanning cards. 	

4.2.2 Technologies available

The platform is composed of a mobile application used for creating and managing cards and a web viewer that allows rendering cards for anyone that is not an user of the application. The mobile platform can be developed using different technologies:

Native application

Native applications are the most common ways to make applications. They usually offer a user experience that is adapted to the specific mobile platform, making navigation more intuitive. When developing native applications, we have access to all available platform components. This facilitates interacting with core components as well as adapting the application to include new functionalities quickly such as extending the application to wearables. On the downside, native development is different for all mobile operating systems meaning no code reuse is usually possible when extending the application to other platforms.

Hybrid application

A hybrid application combines native with web technologies to create a cross-platform mobile application that works on multiple devices. Developers build using web technologies such as HTML5, CSS and Javascript which are then wrapped inside a container that provides access to native platform features.

There's a variety of frameworks that support building hybrid mobile apps. Some of the most famous are the following:

Ionic

Ionic [10] is the most popular framework amongst developers. It includes predefined CSS components that speed up development and easy ways to customize your application according to the platform. It includes a command line interface to easily compile and install the application and has a great community. On the downside, Ionic is designed to work with AngularJS making the initial learning curve a bit steeper for anyone without previous Angular experience.

React Native

React Native's [11] purpose is to build native cross-platform applications instead of hybrid ones that run on a Webview. It is a fairly recent framework in which development is done using JavaScript and React. The framework isn't tailored for beginners and programmers without any previous React experience will have to go through a steep learning curve. What's more, since support for Android application has been released recently, there still isn't a big community behind to provide support with all platforms.

jQuery Mobile

jQuery Mobile [12] was one of the first mobile frameworks to appear. It works on all mobile platforms, including ones with less market share such as Blackberry and Symbian. However, CSS styles are dated and don't resemble current versions of mobile platforms making the platform unattractive to a new developers.

NativeScript

Native Script [13] allows functionalities to be written once and are automatically transformed to work on Android, iOS and Windows Phone. Code only has to be written once but the learning curve is steep and it has a very small community.

As for the development of the server, there's a lot of technologies available including NodeJS, PHP, Java and other languages. We chose to use a known stack called the MEAN stack but changing the MongoDB database for MySQL.

4.3 Market research conclusions

From the market research conducted, we decided that we need an application that combines both scanning cards and providing a way to manually create and organize them. Even though there's a good combination of applications that do both of the things separately, there isn't one that succeeds in combining all of the features users want. These features include: efficiently scanning cards, creating them from zero and being able to group and share them in an easy way. It is also interesting to include card content verification in order to avoid frauds and improve user security and satisfaction. When analyzing technologies, we also saw that we have a wide variety to choose from depending on the functionalities we want to include and whether we want to make the application cross platform.

5 Methodology

In software development, a methodology is a way of dividing work into a series of phases or stages containing activities with the objective of planning and managing the project in a more efficient way. Throughout the degree we studied a variety of software development methodologies used in different types of teams and projects. Some of the methodologies we studied include Cascade, the more advanced Rational Unified Process methodology and a variety of modern Agile methodologies that are gaining increasing popularity.

In order to choose which methodology is more appropriate for our project we have to look at whether the project is creating something new, if it is building something for a client, if the requirements involved are fixed or flexible, etc. In this case, Digital Business Cards is making a new product by trying to improve one that already exists in the market with the objective of making it more attractive to users. However, this doesn't mean that what we're making can be specified in advance in the way a client specific product could be. Since our project depends greatly on user's opinion, our success depends on our capability to respond to user feedback. It is for this reason that we decided to take on an agile approach that will allow us to develop quickly, gain user feedback and make changes to our project depending on the feedback received.

5.1 Agile methodologies

Nowadays, there's a variety of agile methodologies being used by different companies; the most famous ones being Scrum and Kanban. Both of these methodologies are agile and promote the principles in the agile manifesto, however, they differ in how they manage the flow of tasks. While Kanban focuses in controlling the workflow to avoid bottleneck situations, Scrum is very efficiency oriented; promoting fast work and embracing frequent changes.

Considering our need for fast development and the unlikeliness of bottleneck situations due to our single-person team, we decided to develop the Digital Business Cards project following the Scrum agile methodology.

5.2 Scrum

The way Scrum works is by defining tasks that represent what the user wants and creating a series of iterative, incremental iterations in which these tasks are developed. These iterations are called sprints and they help define the team's speed of development for future planification. After each scrum sprint or iteration, the team should have a functional version of the product that they are able to show the user in order to receive feedback. After and during each sprint there are a series of meetings that allow the team to give each other feedback and monitor progress.

When applying the Scrum methodology to the Digital Business Cards project, we adapted it a bit to our needs, reducing the time dedicated to team meetings because our project would be developed by one person. We also defined user stories or tasks in the simplest way possible so that they're independent from each other. As for the iterations, a full description of the tasks developed on each iteration can be found further ahead in this document.

At the end of each iteration or sprint, an updated version of the application was released. This new version was then shown to a group of entrepreneurs that would give us feedback on the added features and on anything they would like the change or add to the application. The entrepreneurs were chosen primarily from the Espai Emprèn at the Polytechnic University of Catalonia, a small co-working space where start-ups can rent a working area for a very small price.

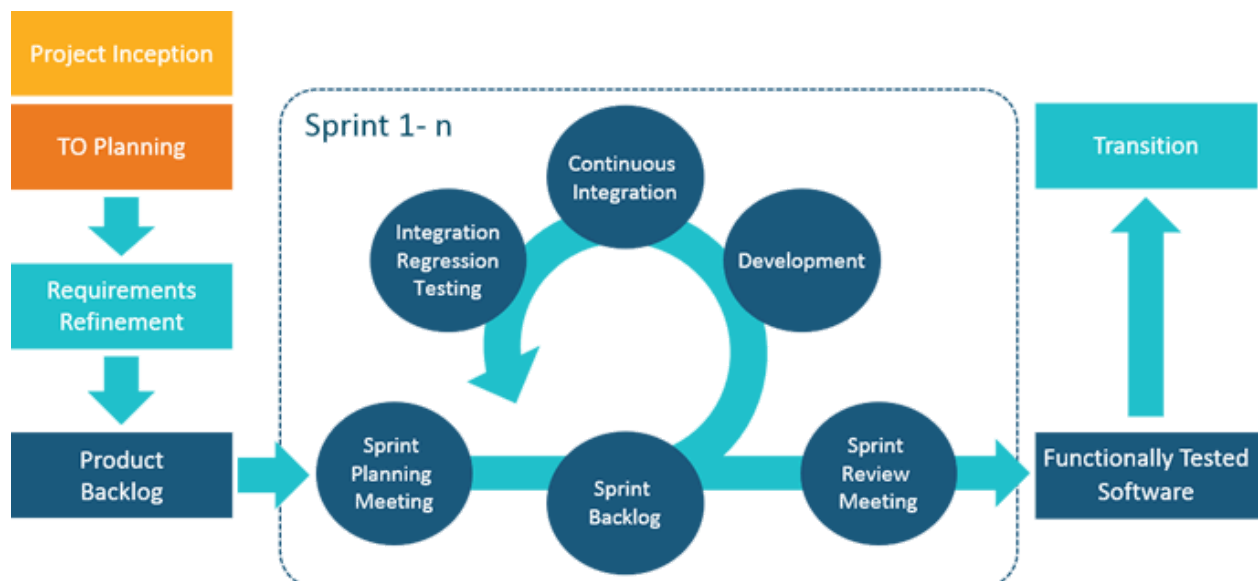


Figure 1. Scrum methodology phases. [14]

5.3 Monitoring tools

Keeping track of the tasks in the product backlog was done using *Evernote*. We made a list of all the tasks in the project and checked them off once we had completed them. We also used *GitHub* issues to keep track of any errors or things that needed fixing but weren't considered tasks.

To avoid losing work and facilitate creating different code versions, we used a version control system. In this case, we decided to use a private *Git* repository hosted on *GitHub*.

In order to communicate with the project director and other interested parts; we used emails and met when necessary.

5.4 Validation methods

In order to validate our work, we asked potential customers to test the incremental versions of the application and give us feedback. Doing so helped us avoid wasting time building unwanted functionalities and focus on making a product that's more desirable for our customers. What's more, when possible, we added automated tests to our code to ensure that adding more functionalities wouldn't break previously developed code.

6 Requirements

6.1 Functional requirements

Register

The platform must allow users to register once they enter the application. Creating an account will require choosing a username and introducing an email and a password which will be stored securely using encryption based on the blowfish algorithm.

Login

The user must be able to login to the platform using his credentials (username and password).

Log out

The user must be able to log out of the application and return to the login and register screens.

Edit account settings

The user must be able to change the settings associated to his account including his primary email, username and password.

Delete account

The user must be able to delete his account so no further access is allowed.

Create card

The user must be able to create personal business cards in order to share with other people.

Edit card

The user must be able to edit and update previously created cards.

Share card

The user must be able to share his personal cards with other users of the application or with anyone via a link to the platform's web viewer.

View card

The user must be able to view a card either in the application or in the web viewer if he doesn't have an account.

Delete personal card

The user must be able to delete any of his personal cards so that they're not available for him or for anyone else to see.

Delete shared card

The user must be able to delete a card that has been shared with him if he doesn't want to store it anymore.

Accept card invitation

The user must be able to accept a card that is being shared with him.

Reject card invitation

The user must be able to reject a card that is being shared with him.

Create group

The user must be able to create groups in order to organize cards.

Add card to group

The user must be able to add cards to groups.

Delete card from group

The user must be able to delete cards from groups.

Delete group

The user must be able to delete groups without deleting the cards inside them.

6.2 Usability requirements

User interface

The user interface must be intuitive and aim to facilitate using the application for all users.

Error handling

The system must be prepared to handle and report errors to the user avoiding causing confusion and giving feedback to users.

6.3 Performance and scalability requirements

Response time

The system must react to changes made by the user in less than 3 seconds considering that the conditions are appropriate (smartphone with at least 1GB of RAM and an internet connection for certain functionalities).

Scalability

Considering the current hosting conditions, the application must support up to 10 users connected on the application at the same time.

Extensibility

The system must be designed in a way that allows and facilitates adding new features.

6.4 Political and cultural requirements

Ley Orgánica de Protección de Datos de Carácter Personal de España (LOPD)

The system must guarantee that complies with the requirements specified in the LOPD.

Open Source licenses

The system must guarantee that it complies with any rules and requirements specified in licenses attached to libraries used in the application.

7 Planning

The project was started on February 22nd with the start of the GEP course and ends on the 24th of October. It lasted a total of 8 months, 4 months longer than expected due to small deviations and an availability problem. Even though the project lasted longer, the overall dedicated hours was maintained and so was the cost.

7.1 Phases

The development of the project was done following the initial planification:

7.1.1 Project planning

Planning the project took part during the development of the GEP course, which was structured in a series of deliverables. Each deliverable included a different part of the project that had to be planned and documented.

During the planning phase we conducted a market research which we used to define the scope of the project. We also defined the state of the art, estimated the necessary resources and considered alternative action plans in case of deviations which are explained further ahead in the document.

7.1.2 Software design and analysis

Before starting any project, it's important to analyze its needs and the technologies available to find the most appropriate tools for development.

We started by specifying the project's architecture and developing the necessary documentation: use cases, database design and interface design. In order to do this efficiently, we based our design on the information collected during the planning phase.

Once the documentation was done, we used it to formally specify the project's product backlog and adapt the tasks to the scrum methodology.

7.1.3 Software implementation

The code implementation took place in a series of 6 iterations lasting 19 days each. These iterations were slightly different from the 4 we had originally planned for our project. This was due to changes in our availability that led to us expending the project's deadline until October which allowed us to have more frequent, shorter iterations.

Following is a small summary of what we want to achieve in each iteration:

First iteration

The first scrum iteration will consist on developing all the basic functionalities of the application to allow the user to create an account, sign in and be able to create a card with the basic data. It will also allow the user to view a simple view of his created cards.

By starting with these tasks, we're setting the base of the application and allowing the user to get an initial taste of where the application is going and what it wants to achieve.

Second iteration

The second iteration will focus on improving the card creation process; allowing to add a telephone number, links and a card image. We will also add the ability to edit a previously created card to finish setting the card design process.

Third iteration

The third iteration will revolve around the ability to share cards. We will use this iteration to work on the ability to share cards with other users, accepting and rejecting card invitations and sending links to anyone that is not using the application.

We will also implement the deletion of a card so that no one else can see a card that has been deleted by its owner.

Fourth iteration

During the fourth iteration we will start developing the creation of groups and the possibility to add and remove cards from them. We will also implement the ability to view our groups to allow users to test the application.

We also decided to include the possibility of deleting cards shared with you to the iteration as we consider it a necessary task to be able to delete a card with outdated information or that is no longer needed.

Fifth iteration

The fifth iteration will consist in adding small features to the application such as group deletion and password modification but most importantly, the creation of the view renderer. The renderer will consist of a responsive web that will allow non-users of the application to be able to view cards shared with them.

Sixth iteration

The sixth iteration will be the last one and possibly the least intensive one. This iteration will include tasks relating to the modification of user account settings such as changing a user's primary email, username or profile image. It was in our best interest to minimize the amount of work expected for this iteration as it will allow us to do any extra work necessary due to deviations in the project.

7.1.4 Defense preparation

The defense of the project takes place once the project has finished. It consists of a period in which we bring all the documentation together in order to present it. We will also use this phase to prepare the slides for the final presentation and defense of the project. We left a one-week margin between the preparation of the defense and the actual presentation date in order to practice the presentation and prepare the debate.

7.2 Gantt diagram

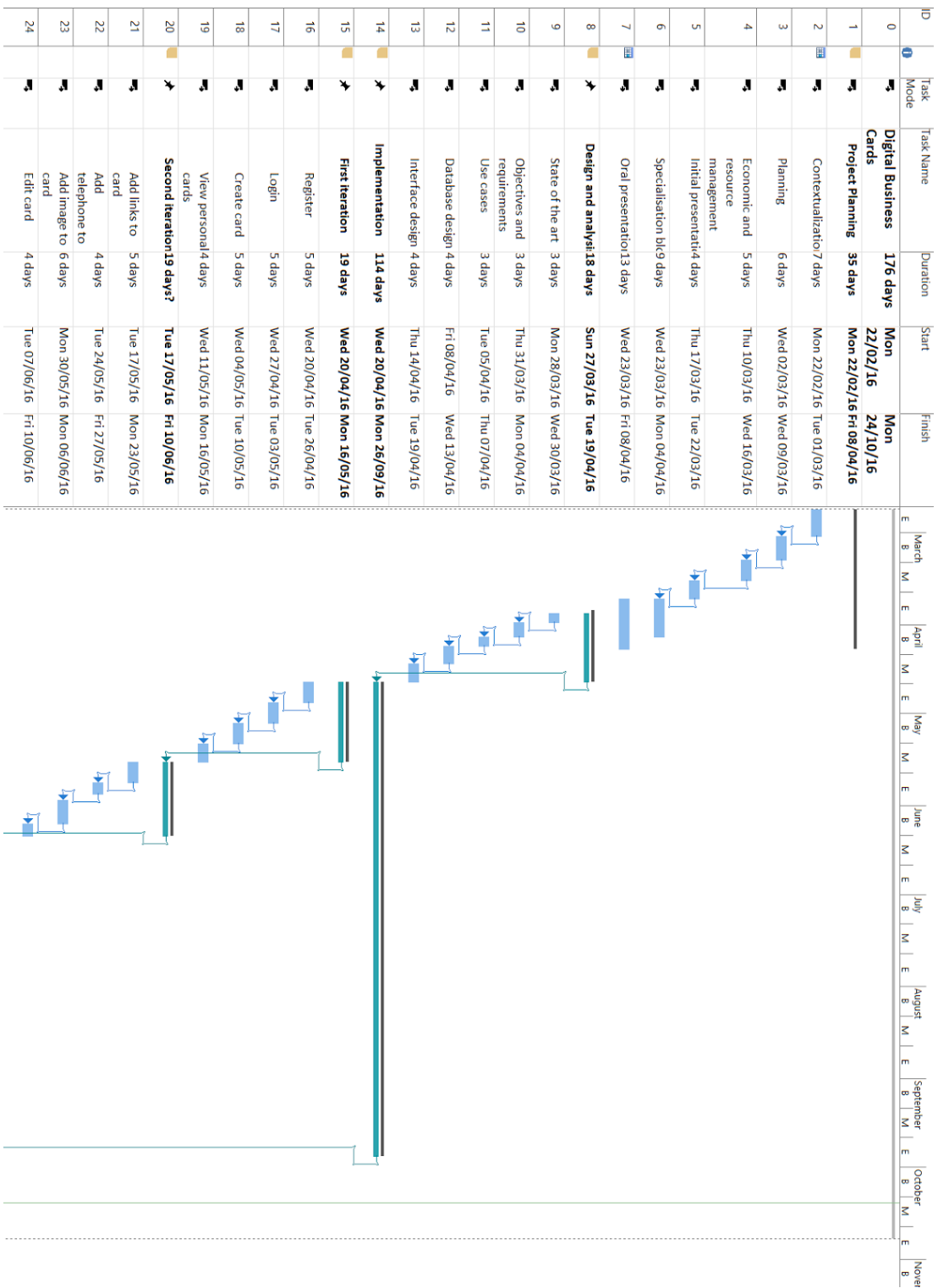


Figure 2. Gantt diagram.

25	🚀	Third iteration	19 days?	Mon 13/06/16	Thu 07/07/16
26	🚀	Share with users	4 days	Mon 13/06/16	Thu 16/06/16
27	🚀	Share via link	6 days	Fri 17/06/16	Fri 24/06/16
28	🚀	Delete personal card	4 days	Mon 27/06/16	Thu 30/06/16
29	🚀	Accept invite	3 days	Fri 01/07/16	Tue 05/07/16
30	🚀	Reject invite	2 days	Wed 06/07/16	Thu 07/07/16
31	🚀	Fourth iteration	19 days?	Fri 08/07/16	Wed 03/08/16
32	🚀	Create groups	4 days	Fri 08/07/16	Wed 13/07/16
33	🚀	View groups	2 days	Thu 14/07/16	Fri 15/07/16
34	🚀	Add card to group	3 days	Mon 18/07/16	Wed 20/07/16
35	🚀	Delete card from group	3 days	Thu 21/07/16	Mon 25/07/16
36	🚀	Delete shared cards	5 days	Tue 26/07/16	Mon 01/08/16
37	🚀	Fifth iteration	19 days?	Thu 04/08/16	Tue 30/08/16
38	🚀	Delete groups	2 days	Thu 04/08/16	Fri 05/08/16
39	🚀	Web-renderer	12 days	Mon 08/08/16	Tue 23/08/16
40	🚀	Change password	5 days	Wed 24/08/16	Tue 30/08/16
41	🚀	Sixth iteration	19 days?	Wed 31/08/16	Mon 26/09/16
42	🚀	Change primary email	2 days	Wed 31/08/16	Thu 01/09/16
43	🚀	Change username	2 days	Fri 02/09/16	Mon 05/09/16
44	🚀	Change profile image	6 days	Tue 06/09/16	Tue 13/09/16
45	🚀	Defense preparation	20 days	Tue 27/09/16	Mon 24/10/16
46	🚀	Final document	14 days	Tue 27/09/16	Fri 14/10/16
47	🚀	Presentation preparation	6 days	Mon 17/10/16	Mon 24/10/16

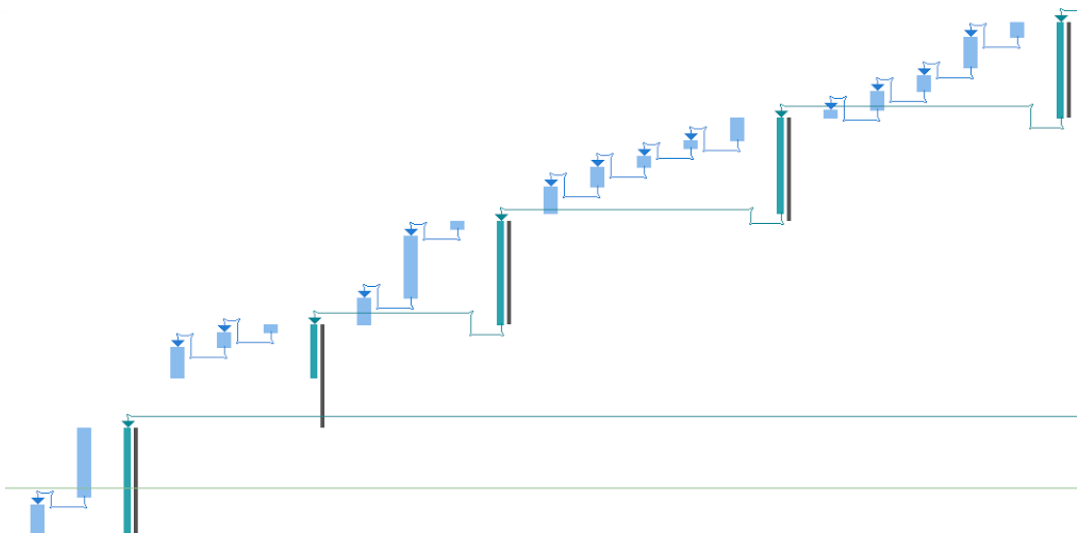


Figure 2. Gantt diagram.

7.3 Alternatives and action plan

7.3.1 Estimated time

Phase	Estimated duration
Project planning	80
Software design and analysis	80
Implementation	240
Defense preparation	50
Total	450

Figure 3. Project estimated times.

7.3.2 Action plan

The execution order of the phases is equivalent to the one we followed when defining them. This is due to the fact that we needed to investigate and design certain parts of the application before starting to implement them. Other than that, the only phase of the project that has a specific method or order of execution is the implementation phase in which we followed the scrum methodology. The use of scrum meant that the development took part in a series of iterations after which we gathered user feedback to improve the project. In order to minimize the effect of these changes, we added a few extra hours to the implementation phase.

7.3.3 Resource consumption

Resource	Type	Aim
Laptop MacBook Pro 15” running OSX 10.10, Windows 8.1 and Ubuntu Linux	Development tool	All code development will take place on this computer.
Android smartphone OnePlus One	Development tool	To test the Android application.
Atom	Development tool	Text editor to develop the server, app and web page.

Git and GitHub	Development tool	To control the different code versions.
Heroku	Development tool	To host the server and web page.
Email (Gmail and FIB emails)	Communication tool	To interact with the stakeholders.
Evernote	Planning tool	To keep track of the project tasks.
Google Drive	Documentation tool	To write the documentation and synchronize it between devices.
Draw.io google drive extension	Documentation tool	To make the UML and other necessary diagrams.

Figure 4. Resource consumption.

7.3.4 Alternative solutions

Agile methodologies try to increase productivity by being flexible. This means that tasks and priorities can change at the start and end of each iteration and extra time has to be reserved to handle these changes. The flexible nature of our methodologies means that there aren't alternative solutions to our action plan but rather changes will be dealt with by cutting off tasks or adding hours to the project depending on the decisions made by the product owner.

7.3.5 Current state of the project

The project is currently in the defense preparation phase in which we develop the final documentation and prepare its presentation. All scrum iterations have been finished so no new functionalities will be developed from this point on. It is worth noticing that one of the features that was initially included in the scope of the project has been simplified: we decided not to include the possibility to interact with data offline in favor of creating a multi platform application. We did this because feedback from users showed that the majority saw it as more valuable to initially have an application available over different devices than offline. When asking users why they saw it as more valuable, the main reason was that nowadays there's network access almost anywhere you go but you're not always working on the same platform. What's more, by making this decision we also facilitated future development such as extending the platform to a web or desktop environment.

7.3.6 Changes from the original planification

When carrying out the different phases of the project, certain tasks changed from what we had planned in the initial planification. The main changes can be seen in the software implementation phase where we initially planned to carry out development by implementing the different layers separately but later decided that following an iterative task implementation approach would adapt better to our agile methodologies. This meant that instead of starting by implementing the API, we only built a very basic version of it and added any necessary functions only when they were needed. This allowed us to minimize writing unnecessary code and helped us develop faster.

As for the amount of dedicated hours, as mentioned before, the extension of the project's deadline didn't imply that more hours were being put into the project but rather that we were carrying them out through a longer period of time.

The resources and action plans remained unchanged, even if we shifted from our original plan of making an Android application to a cross platform one.

8 Budget

Once the project has been planned, it's important to also study the costs resulting from the project's development in order to be able to estimate a budget and plan for any other expenses. Once the budget is estimated, we will proceed to estimate the project's revenue in order to validate the investment.

8.1 Identification of costs

The cost of the project is directly associated with the resources specified both in software and hardware resources and the costs of the employees.

To estimate the budget, we will consider the average salary of the different team members considering the hours we estimated the development would take and the cost of the necessary software and hardware. When calculating the cost, we will also consider adding an extra amount to the total to cover for unexpected changes and contingencies.

8.2 Estimation of costs

In order to improve the estimation of costs, we have separated them into three main groups: human resources, software and hardware.

8.2.1 Human resources

The estimation associated with each role is the average price per role established in the 2016 report from *Page Personnel* [15]. It should also be noted that even though we divided salaries by job title, the project will be developed by one unique person who will be in charge of all of the tasks.

Job title	Estimated hours (h)	Hourly salary (€/h)	Estimated cost (€)
Project director	265	19,5	5.157,5
Analyst	65	17,5	1.137,5
Designer	45	10,5	472,5

Developer	225	10,5	2.362,5
Tester	72	13	936
Total	672		10.066

Figure 5. Human resources.

8.2.2 Hardware resources

When considering the cost of hardware, we considered that each device would have an operating time of around 4 years. In order to estimate the amortized cost per hour, we considered that 1 year of operating life consisted of 251 days with a use of 8 hours per day.

Amortized cost = Price / (4 years * (251 days * 8 hours/day))

Hardware	Price (€)	Estimated hours (h)	Amortized cost (€/h)	Estimated cost (€)
Laptop MacBook Pro 15"	2.249	450	0,2800	126
Android smartphone OnePlus One	299	80	0,0372	3
Total				129

Figure 6. Hardware resources.

8.2.3 Software resources

Software resources will have to be renewed every 3 years.

Amortized cost = Price / (3 years * (251 days * 8 hours/day))

Software	Price (€)	Estimated hours (h)	Amortized cost (€/h)	Estimated cost (€)
OSX 10.10	0	-	0	0
Atom editor	0	-	0	0
Git and GitHub	0	-	0	0
Heroku	0	-	0	0
Email (Gmail and FIB email)	0	-	0	0
Google Drive	0	-	0	0
Draw.io google drive extension	0	-	0	0
Total				0

Figure 7. Software resources.

8.2.4 Other expenses

Other expenses involved in the development are the following:

Energy consumption: to charge both the laptop and the smartphone.

Internet connection: a fiber optic internet connection service will be contracted.

Printing: to hand the documentation of around 100 pages to the 3 board members and to the project director.

Description	Price (€)	Quantity	Estimated cost (€)
Energy consumption	0,15€/kWh	85 watts	160,65 ¹

¹ ((85 watts*18 weeks*7 days)/1000)*0.15 kWh

Internet connection	14,90€/mes	5 months	74,5
Printing	0,05€/page	400 pages	20
Total			255,15

Figure 8. Other expenses.

8.2.5 Deviations

Extra time has been added to the implementation phase of the project in order to handle small deviations that may occur.

In the case of big deviations, since we can't temporarily extend the project, the product owner would be in charge of reconsidering priorities and deciding what's to be implemented. In other words, if deviations led to a shortage of time, the product owner would have to prioritize the remaining tasks and cut off the least important ones.

If we were to extend the project in order to finish all of the tasks, the cost per day would be the following:

Cost per deviation day = (0,2 risk of deviation * duration in hours of the project / 4 hours per day) * ((cost of human resources + hardware costs + software costs + general costs) / duration in hours of the project * 4 hours per day) = 23 * 5,82 = 133,86€

8.2.6 Contingencies

To cover the cost of contingencies, we will add a 10% margin to the total cost of the project which results in an extra cost of 1.047,05².

8.2.7 Total cost

Considerations:

- Price fluctuations during the development of the project haven't been considered.
- No profit margin was added to the total cost because this is a nonprofit application.

² (10066+129+275,5)*0,1

Resource	Estimated cost (€)
Human resources	10.066
Hardware resources	129
Software resources	0
Other expenses	255,15
Deviations	133,86
Partial cost	10.584,01
Contingencies	1.047,05
Total cost	11.631,06

Figure 9. Total costs of developing the project.

8.3 Control management

We will use different methods to control the different parts of the project and prevent deviations. However, it's important to keep in mind that not everything can be controlled.

- In order to control the human resources, we will check that work is done in the established dates and that we aren't falling behind. The checking will take place in a series of meetings that will take part during and at the end of each Scrum sprint.
- To control the other resources, we will also compare the dedicated hours to the ones we estimated in the project planning. If there's a significant difference, we will talk to our product owner so he can decide whether to cut tasks off the project, replanning the remaining tasks or to extend the deadline. In any case, the contingency costs will have to be used to cover for the new expenses.

The calculation of the different deviations will be done using the following formulas:

- Deviation of the labor costs = (estimated cost - real cost) * consumption of real hours
- Deviation of the development of a task = (estimated cost - real cost) * consumption of real hours
- Deviation of the cost of a resource = (estimated cost - real cost) * real consumption
- Deviation of the cost of a task in consumption = (estimated cost - real cost) * real cost
- Total deviation of the task development = total estimated task cost - total task cost

- Total deviation in the resource consumption = total estimated resource consumption - total cost of resources

8.4 Final revision and cost deviations

Now that the project has finished we can analyze the cost, how the deviations affected it and if the estimations were accurate enough to cover for all the expenses. We should start by commenting that, even though the project's time frame was extended, this wasn't because the work took longer than expected or the estimations were wrong but rather due to a lack of availability from the developer. During the time period when the project was expected to occur, the developer found itself not being able to dedicate as much time to the project as originally planned. It is for this reason that daily dedicated hours were reduced and sprints were extended as we couldn't dedicate as many hours as originally planned. However, this didn't affect the budget of the project as it was developed in a similar amount of hours as initially estimated, which means that the total cost of the project was 11.631,06€.

8.5 Return on investment

The initial version of the application will be available for free download. However, upon release, we will include a donations section in our website where people can send us money voluntarily. As we continue development and adding new features to the application, we will analyze how the donations section is working and decide whether it's necessary to create a paid version of the application or no. If it were necessary, we would include some of the most exclusive parts of the application such as offline availability or scanning cards into the paid version in order to increase income without making the platform unusable for users who don't want to pay. To avoid upsetting users who had previously donated, each donation worth 2,5€ or more will allow associating an email to the donation so that so that if we were to release a paid version, users who had previously donated would be able to redeem the application for free.

In order to decide on the price, we decided to look at some key studies. The studies showed that iPhone users spend almost 2.5 times more than Android users in in-app purchases where the average iPhone user spends \$1.08 (0,96€) and the average Android one \$0,43 (0,38€). Not only that but, iPhone users are 50% more likely to spend money on applications and they spend double what Android users spend on the average purchase.

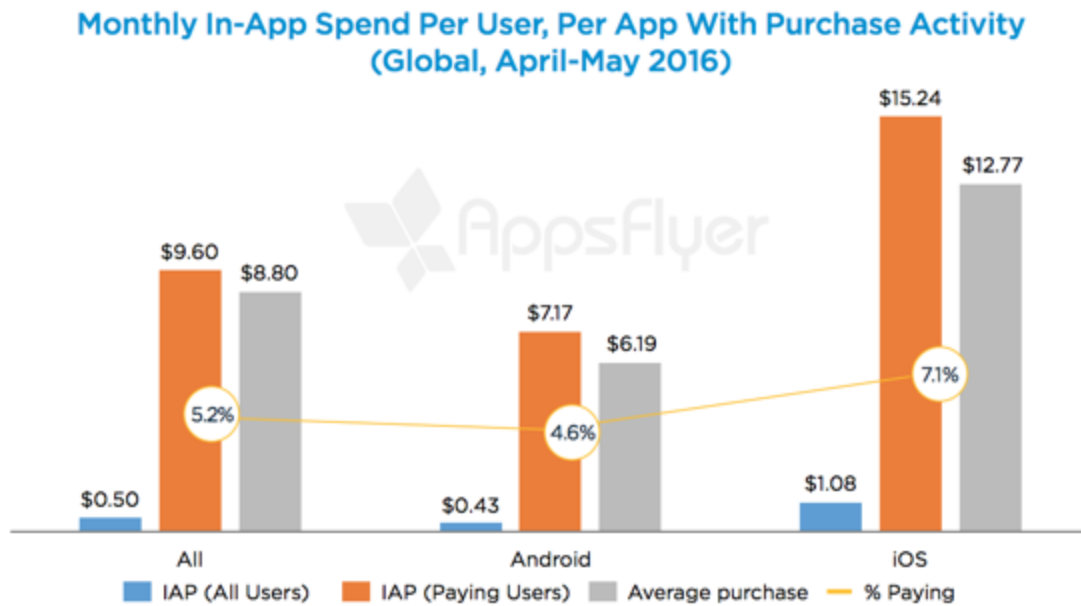


Figure 10. Mobile phone spending statistics. [16]

Taking into consideration the information above, we decided to set different prices for iPhone and Android users; the iPhone one being slightly higher.

Platform	Probability (%)	Price (€)	Weighting
iOS	60	3	1,8
Android	40	2.5	1
Revenue per sale			2,8

Figure 11. Analysis of the expected revenue per application sale.

As seen on the table, the average revenue per sale is 2,8€. This means that in order to amortize the cost of the product, we would have to sell $11.631,06 / 2,8 = 4.154$ units of the finished application in order to recover the investment.

Amortization

Taking into consideration that there are 1 billion Android users in the world and 470 million iPhone users of which a 4.6% and a 7.1% make application purchases; that would mean that we have $1 \text{ billion} * 0,046 = 46 \text{ million}$ potential Android customers and $470 \text{ million} * 0,071 = 33 \text{ million}$ potential iPhone customers [17].

If we sell 360 iPhone applications and 300 Android ones in the first 3 months and 2.700 iPhone and 2.250 Android ones in the remaining 9 months of the year, we would be amortize the cost of developing the application during the first year of sales.

9 Sustainability

9.1 Economic sustainability

The quantity and type of resources have been calculated accurately, taking into account the current prices of the resources and attempting to keep expenses to a minimum. The price of possible adjustments and unexpected events has also been taken into account and different action plans have been established and described. The time spent doing each task is proportional to the amount of work it requires and the project seems a reasonable investment for the price and effort it requires.

9.2 Social sustainability

The project will be developed in a city with an increasing number of startups and entrepreneurs which represent an important part of the target market. Entrepreneurs and small business owners can benefit most out of the development of the application because it will enable them to save on printing and designing cards while maintaining the advantages that they provide. However, due to the initial development not including some important features, this dimension will be awarded with a 6.

9.3 Environmental sustainability

All resources used in the project were previously owned, meaning that no new hardware or software has been bought for the development of the project. However, both the laptop and the smartphone have an energetic consumption that should be considered when estimating the cost of using these resources. Nonetheless, the constant use of both of the devices involved means that the added energetic consumption is almost insignificant. What's more, both the laptop and the mobile phone have been used previously, meaning that their cost has already been amortized.

When developing the project we don't have to worry about recycling as we can simply erase data from the hard drive. There also won't be any pollution as the resources used will continue to be used after the project has finished.

9.4 Sustainability matrix

	Project Development	Exploitation	Risks
Environmental	8:10	15:20	0:0
Economic	7:10	10:20	-10:0
Social	7:10	12:20	0:0
Partial Assessment	22:30	37:60	-19:0
Total	49:90		

Figure 12. Analysis of the sustainability through the sustainability matrix.

10 Design

10.1 Architecture design

The project's is structured into three separate components that interact with each other to form the platform. The components are the following:

- The **database** is where we store all the information registered in the system. Whenever a user wants to add or access information in the application, we will interact with the database in order to store or query the data.
- The **server** is in charge of handling client requests, interacting with the database and replying to the client.
- The **client** is the final application that the user will have in their devices, be able to see and interact with. When the user interacts with the client, this one will be in charge of handling those interactions and interacting with the server if necessary.

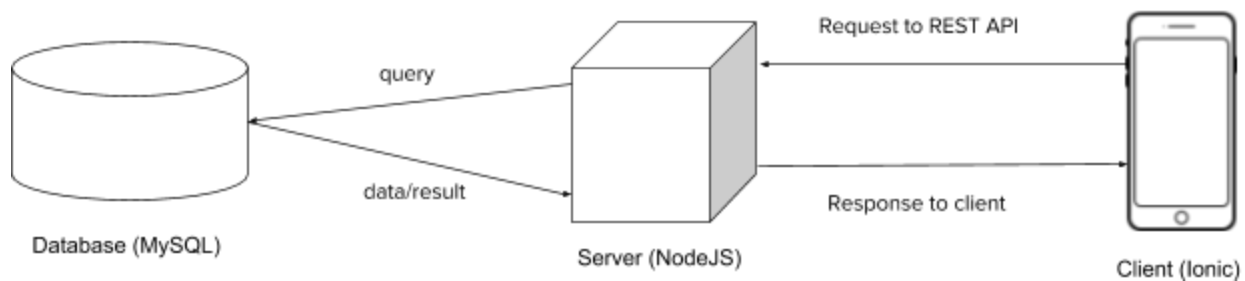


Figure 13. Platform architecture and interaction between different components.

10.2 Database design

In order to understand how the application was structured, we will continue by describing the database that it uses. It was implemented using MySQL and is hosted on heroku. The structure of is the following:

- A **user** is an account inside the application.
- An **email** is equivalent to an email address associated to a user account.
- A **group** is created by a user and contains a subset of cards.
- A **card** is part of a group that belongs to a user.

10.3 Server design

The project's server was made using Node.js. Node.js is a server side JavaScript execution environment built on Google Chrome's V8 JavaScript engine. We chose to use Node.js in our server because it's lightweight and efficient, uses an event-driven and non-blocking I/O model and allowed us to build highly scalable and concurrent applications rapidly. What's more, it comes with npm (Node Package Manager) which is the largest ecosystem of open source libraries in the world.

When building the server we also used several packages from the package manager. Following is a list of the most important packages that we used:

- **Express** [18][19] is a minimalist web framework for Node.js. What it does is apply a level of abstraction to the creation of the server in a way that simplifies development and makes code more readable. What's more, it also simplifies other functionalities such as routing or applying intermediary functions or middleware.
- **Mysql** [20][21] is a Node.js driver for MySQL. It allows us to connect to the database via the Open Database Connectivity (ODBC) API which is the standard means of connecting to any database.
- **Bcrypt** [22][23] is an npm package that implements the password hashing function based on the Blowfish cipher. We use this package to hash users passwords when storing them in the database.
- The **jsonwebtoken** [24][25] package is used to create tokens based on each user's information and maintain login sessions. When the user logs into the application, we generate a token with their login information and save it in the browser until the user logs out.
- We use **nodemailer** [26][27] to facilitate the process of sending email to users whenever they sign-in, create a card with an unverified email or want to change their password.

When choosing packages that were intrinsic in our application's functioning, we took into consideration the amount of downloads they had and how active their code repository was by checking the date of the last commit and the open pull requests:

Package	Downloads in the last month	Open pull requests	Latest commit date
Express	7,161,174	37	31 July 2016
Mysql	513,802	18	30 Sept 2016
Bcrypt	349,522	-	11 Sept 2016
Jsonwebtoken	841,520	5	11 Aug 2016
Nodemailer	670,638	-	22 Sept 2016

Figure 15. Packages used in the development.

Authentication and access tokens

The application's authentication system is based on JSON Web Tokens (JWT) [28]. JWT is an open standard that defines a self-contained way to transmit information between two interested parts. The information can be trusted because it's digitally signed using a secret key.

Following is an example of the authentication process in a series of simple steps:

1. The user authenticates with the application using his username and password.
2. If the authorization is successful, the server generates a Token with specific data about the authenticated user and sends it to the client.
3. Upon receiving the Token, the client saves it in the local storage to have it available to send with any request that requires authentication.
4. Whenever the server receives a request that requires authentication, it will check if the Token information is correct and either give the user access or send an Unauthorized error.

It's important to keep in mind that this authentication method relies on our ability to securely send information through the network. This means that in order for this method to be effective, a secure communication protocol should be used. In order to deal with this, we decided to use the HTTPS [29] protocol that deals with this problem by securely encrypting data before transmitting it.

API endpoints

Following is a list of all the API endpoints available, their path and a small description:

File	Verb	Path	Description
Authentication	POST	/authentication	Register.
	POST	/authentication/login	Login.
Users	GET	/users/{id}/profile	Get user profile data with user id {id}.
	GET	/users/{id}/allCards	Get all user cards where the user id is {id}.
	PUT	/users/{id}	Update user account data with id {id}.
	DELETE	/users/{id}	Delete user account with id {id}.
Cards	GET	/users/{id}/cards	Get user card with id {id}.
	POST	/users/{id}/cards	Add new card to user with id {id}.
	PUT	/users/{id}/cards/{card_id}	Update the user card with card id {card_id}.
	DELETE	/users/{id}/cards/{card_id}	Delete the user card with card id {card_id}.
	GET	/users/{id}/cards/{email}/resendVerification	Resend the verification email for a certain user email {email}.
Groups	GET	/users/{id}/groups	Get all groups of user with id {id}.
	POST	/users/{id}/groups	Add a group to the user with id {id}.
	GET	/users/{id}/groups/cards	Get all cards in groups owned by the user with id {id}.
	GET	/users/{id}/groups/{group_id}/cards	Get cards in the group with id {group_id} owned by the user with id {id}.
	POST	/users/{id}/groups/{group_id}	Add card to group with group id

		d)/card	{group_id} owned by the user with id {id}.
	DELETE	/users/{id}/groups/{group_id}/card/{card_id}	Delete the card with id {card_id} from the group with id {group_id} owned by the user with id {id}.
Invites	GET	/users/{id}/invites	Get invites belonging to user with id {id}.
	POST	/users/{id}/invites	Add invite to user with id {id}.
	POST	/users/{id}/invites/delete	User with id {id} deletes a card shared with him.
	POST	/users/{id}/invites/accept	User with id {id} accepts a card that has been shared with him.
	POST	/users/{id}/invites/reject	User with id {id} rejects a card that has been shared with him.
ResetPassword	GET	/resetpassword/{token}	Shows form for the user identified with the token {token} to reset his password.
	POST	/resetpassword/request	User requests a password reset and an email with a special identifier token is sent.
	POST	/resetpassword/reset	Updates the user's account with the new password.
VerifyEmail	GET	/verifyemail/{token}	Verifies the user's email.

Figure 16. API endpoints.

10.4 Client design

The client was implemented using the Ionic framework. Ionic is an open source mobile SDK for building native and progressive web apps. It builds on top of Angular making development easier for everyone with an Angular background. The framework is performance oriented, designed with practices such as efficient hardware acceleration transitions and touch-optimized gestures. It also includes its own CSS components and allows for platform customization.

We chose to build our app with Ionic in order to facilitate cross-platform without having to worry about not being able to create platform specific customization or about having performance

issues. What's more, we are experienced with Angular meaning that developing with Ionic would make development easier for us.

10.4.1 AngularJS and the MVC pattern

The client side of the project follows the Model-View-Controller (MVC) architectural pattern. This is due to Ionic being based on Angular and Angular implementing the pattern. The Model-View-Controller pattern is one of the most commonly used architectural patterns in platforms that involve implementing user interfaces. What it does is divide a given software application into three interconnected parts that separate the internal representation of information from the information that's being shown to the user.

The three parts are the following:

- The **view** is the part of the application in charge of showing the user the information in the model in a way that the user can understand and interact with it. Since our application was built using web technologies, the views are presented in HTML which is generated by binding the model data into the HTML file.
- The **model** is the one that manages the data and connects the backend data source to the rest of the application. Traditionally, when the user uses the application, the model's are accessed and the retrieved backend data is combined with the view template. As the user interacts with the application, the controller continually queries the model for data and as soon as the model receives a response from the backend data source, either the page is reloaded or information is injected into the view via AJAX calls. However, in an Angular application, the view and the model are intertwined in a way that views are considered a projection or the current model state. This is due to Angular's two-way data binding; which as soon as the model changes, updates the view automatically. By doing this, Angular simplifies development as no additional effort is required besides the initial configuration.
- The **controller** is the one that handles the logic in the system and manipulates data. It accepts input from the user and converts it into commands for the model or view. In the case of a web applications where interactions usually involve HTTP requests, the controller is in charge of deciding how to handle the request, manipulating the necessary data and passing the model to the view.

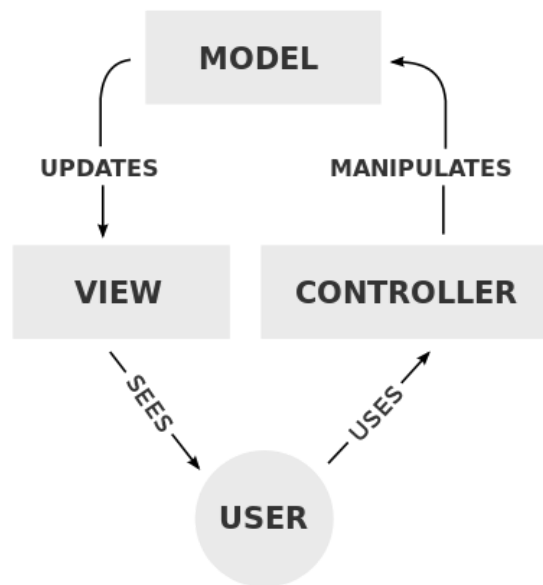


Figure 17. The model-view-controller pattern. [30]

The use of the Model-View-Controller pattern is very effective in this part of the project because it helps us ensure a certain code quality. Even though initially web developers mixed the code of the front end and back end of applications, they found that this was hard to understand, extend and maintain. It is for this reason that software engineers decided to create a series of architectural patterns that would help improve and generalise the structure of the code in projects. By using the MVC pattern we're slightly sacrificing efficiency in order to improve code structure and maintenance.

10.5 Testing

Testing is an essential part of agile software development that guarantees that a program is executing correctly. During the development of the Digital Business Cards platform, different methods were used to ensure the correct functioning of different parts of the application.

Mocha [31] is a testing framework for NodeJS that allows Asynchronous testing. It comes with features such as timeout support, promises and a variety of hooks to prepare the testing environment such as *before*, *after*, *before each* and *after each* hooks which allow executing functions before the start of a testing block, after or before and after each test inside the block. However, even though Mocha lets us send HTTP requests to our API to retrieve data, it doesn't

provide us with the tools necessary to check if the data received is correct. This is where libraries such as Chai [32] come in.

Chai is an assertion library for NodeJS that allows us to check whether the information obtained from the API calls is correct or not. It comes with a variety of interfaces such as `should`, `expect` or `assert` that allow us to compare the data obtained with our expected results while allowing different assertion styles that improve the readability of our tests.

When testing the backend part of our project, we combined these two tools to make HTTP calls to our API and check that the response data obtained matched our expectations. More specifically, we tested the backend of our application using an approach known as black-box testing [33]. This software testing approach verifies the functionalities of an application from a public interface point of view, without peering into its internal structures or considering any knowledge of the internal workings of a specific function or use case.

We applied our black-box testing approach to the different endpoints in the API, grouping them into blocks according to the objects they dealt with (users, cards, groups or invites) and making specific functions for each of the endpoints. In order to know what is in the database and be able to assert expected results, we defined a test database that will be used whenever we run the application's test command. Before starting each test block, we also set up a hook that drops the database and creates it again, erasing any previously stored information. By doing this, we can execute each test in a controlled and isolated environment that will allow us to confidently assert results and check for any errors.

Running the tests is done using the command `npm test` that runs the testing command specified in our project's package json. In our case, this is the mocha testing command with a specific timeout set.

Regarding the frontend of the application or client, no automated tests were written but rather, a lot of focus was put into implementing the interface using Ionic's CSS framework and Javascript UI library. This means that whenever possible, we made sure to use interface components provided by the Ionic framework that have been previously tested and that have been developed to be specifically used with the Ionic framework. By doing this, we followed design guidelines and ensured responsiveness. Using these components also facilitated interaction as a lot of the Javascript view controllers provided by Ionic imitate native UI functionality, improving the overall application design and feel.

When it comes to performance, our focus on minimising costs and thus relying on free hosting both for the database and server made it difficult to provide real performance measurements. Tests such as stress tests that we could provide would not prove significant in our current prototype environment. Nonetheless, both our database and our server are currently being

hosted on Heroku, a cloud platform as a service (PaaS) that allows for easy and fast plan upgrading depending on the resources required by the user. By using Heroku, we can easily change our account plan to allow for a larger number of concurrent database connections or improve our server's performance by buying better machines allowing for vertical scalability.

11 Cross-cutting concerns

Nowadays, all applications have a series of responsibilities that affect both the code and development of the application and that should be considered during the development. Cross-cutting concerns are responsibilities that are applicable throughout the application and should be considered when developing the application as they represent things that could go wrong with our application and that we should try to avoid or plan for. These responsibilities could be things such as application security, reusing code, authorising users, transferring data, etc. Following, we will list a series of cross-cutting concerns that were considered in our application:

11.1 Security

The first cross-cutting concern involves holes in the platform's security. Given that the application is very interactive and allows for a lot of user input, we have to consider several security concerns.

Input variables

Receiving input from the user always represents a security threat as a knowledgeable user can make use of this opportunity to try to exploit the platform. Secure input and output handling are techniques designed to help prevent these attacks:

Input Validation

When data is sent from the client to the server, it travels in the request. A user with minimum web development knowledge is able to bypass javascript client verifications and intercept these requests, modifying them to send unwanted information to the server. It is for this reason that when we expect input from the client, we should always check that the input is correct once it arrives to the server as incorrect input saved in our database or executed in our server can grant a hacker access to our data. By validating all the data that arrives to our server, we're also protecting ourselves against XSS attacks that try to inject malicious scripts into our servers.

SQL injections

SQL injection is a code injection technique in which undesirable SQL statements are inserted into an application's input field in the hope of it getting executed [34]. One of the most common ways to execute an SQL injection is by sending SQL as user variables. If the developer doesn't escape the variables, the SQL query will execute and the hacker might be able to retrieve more information than he should. To avoid this vulnerability in our project, we've injected variables into our queries by using the mysql package method "format" which takes care of escaping the variables for us.

Directory traversal

This attack consists in exploiting software that insufficiently sanitizes user-supplied filenames in a way that characters representing for example traversing to a parent directory (../) are passed to the file APIs. By doing this, the attacker is able to access server files that weren't supposed to be accessed which might mean access to private information or an easier way to find a security hole in those files.

Another vulnerability would be showing the user the file structure in our server as even though this doesn't imply a vulnerability itself, the attacker may use this information to find other exploitations.

What we did to solve this is provide a default route in case no other routes defined by us match the one that the user is trying to access and configure our server to disable directory indexing to avoid it sending the automatically generated index file in each directory which would reveal our directory structure.

Search indexing

Search engines such as Google use software known as "web crawlers" to discover publicly available web pages [35]. These web crawlers look at pages and follow links on them to analyze the information, create a page ranking and be able to serve better and faster results whenever a user searches for something on the engine. Even though the links to our business cards are public, we don't want to search engines such as Google to index our pages and show them on their results as that could put at risk our user's information.

What we did to avoid having our pages indexed is create a file called "robots.txt" and include a meta tag in our html files that tells crawlers that we don't want our web page to be indexed [36].

Brute force attacks

Brute force attack consist in an attacker making requests to a server using predetermined values and analyzing the response [37]. By doing this the attacker can create scripts such as an URL checker where he tests a range of values to try to gain access to private information. Brute force attacks represent a threat to our application as our cards have publicly available URLs that allow sharing with external users. What we did to avoid attackers discovering available URLs by brute force is assign each card a 36 character UUID that would be part of the card URL making it nearly impossible for them to be guessed by brute force.

11.2 User authentication

Users

Almost all operations in the Digital Business Cards platform require user authentication. The way we handle authentication is by sending the user's unique token with every API call that requires it. However, this token is travelling unprotected in the header of the request making it vulnerable to attacks such as man-in-the-middle, packet sniffing and tampering, etc. In a production environment we would avoid this vulnerability by using HTTPS and SSL that encrypt packet information to securely transfer data over the network.

12 Development

The following section will go into a bit more detail of how we developed the project and how we applied the Scrum methodology in the implementation phase of our project.

12.1 Scrum sprints

As we explained earlier in this document, a Scrum sprint is a time period that usually lasts between two weeks and one month and that serves as a small period of time in which a set of project tasks or user stories are developed. The development of our project took part in a series of 6 sprints that lasted 25 days each. Sprints in our project were a bit longer than usual because during the period of time we wouldn't have full-time dedication to the project and the user stories are overall quite big.

Following, we will explain the different phases that each Scrum sprint had in order to explain the development process and how each individual task was developed:

12.1.1 Sprint planning

Each iteration started with a planification of what tasks we were going to develop. We based this on our average iteration speed that was established according to the number of story points developed during the first sprint. That means that we chose as many tasks as possible, taking into consideration our speed and the priority that each user story had; making sure we chose primary stories without surpassing our speed by a great amount.

12.1.2 Development of the user stories

The development of each user story started with us reviewing any available documentation to make sure that we understood the task and knew how to develop it; making any extra documentation when necessary. We started each sprint by developing the user stories with more priority to ensure the development of the most essential tasks.

For each task, we had to implement the different tiers. We started by implementing the application tier or business logic in charge of executing the necessary processes to satisfy the use cases. Once the business logic was done, we proceeded to connect it to the database or data tier to interact with real application data. In the mobile application, this meant interacting

with the API. Even though some of the server endpoints were developed in an initial version of the API, we only built a very basic initial version of the API to avoid building anything unnecessary. For this reason, there were cases in which the development of a user story required an additional API endpoint, in which case we would develop it as part of the user story. When everything worked, we started implementing the interface or presentation tier and connected it to the two other layers.

Once the user story was working, we proceeded to test it thoroughly to ensure good behaviour and reduce the number of application bugs. Initially, the testing was done manually to avoid delaying the development, making automated tests only for key parts of the application in which bugs wouldn't have been easily visible.

12.1.3 Daily Scrum meeting

Daily Scrum meetings consist of the team, Scrum master and product owner meeting to discuss development, the current state of the tasks and if there is anything that is blocking the way or slowing down the project. Having a single developer in our project, daily Scrum meetings were held by the single person. This meant that instead of being a discussion, daily meetings were the developer's reflections on the outcome of the day, reporting any problems that should be communicated to any other parts of the project or communicating any changes if necessary.

12.1.4 Retrospective

Once the iteration was over, we looked back at what was done and evaluated if the tasks were finished or needed extra work. This was done by analyzing each functionality and comparing it to what was initially planned and expected of them. If we didn't consider them to be finished or we found any bugs in them, we included them in the next iteration in order to resolve any errors. All other accepted tasks were put into a separate list of finished user stories.

12.2 Project retrospective

Once all iterations were over, we did a project retrospective to analyze the overall development of the project.

Velocity

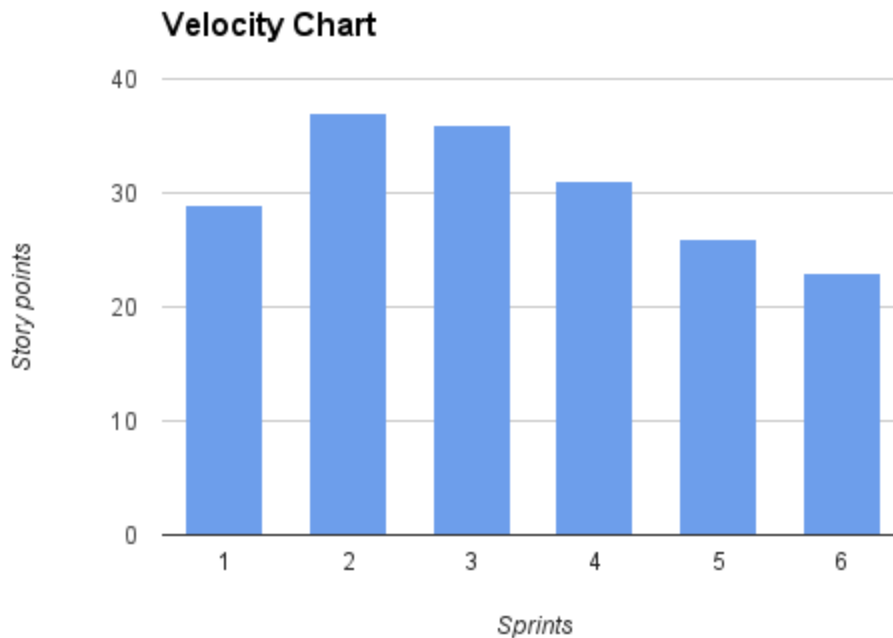


Figure 18. Scrum development velocity chart.

The above chart shows the amount of story points developed in each sprint. It's a good way to visualize distribution of work and overall evolution of the project.

If we look at the story points developed during the first iteration, we'll see that the amount is slightly less than those developed in the second iteration. This is a common thing in software projects as the first iteration requires the extra work of setting up the project and environment and learning how to use any new technologies being used. In our case, we wasted a lot of time analyzing databases that would provide offline functionality. This means that even though the amount of completed story points is lower, the amount of dedicated hours was the same.

As for the second and third iteration, they have a similar number of story points. These are also the peaks of the velocity chart.

The fourth and the other two remaining iterations, on the other hand, have less story points because there were fewer tasks to choose from. We also intentionally aimed to leave a small amount of tasks for the sixth iteration to minimize the effects of possible deviations and leave time to develop necessary documents such as the monitoring report and final document.

Lastly, we left a couple stories that weren't included in the scope of this project but that we hope to include in the application at some point in time.

User stories

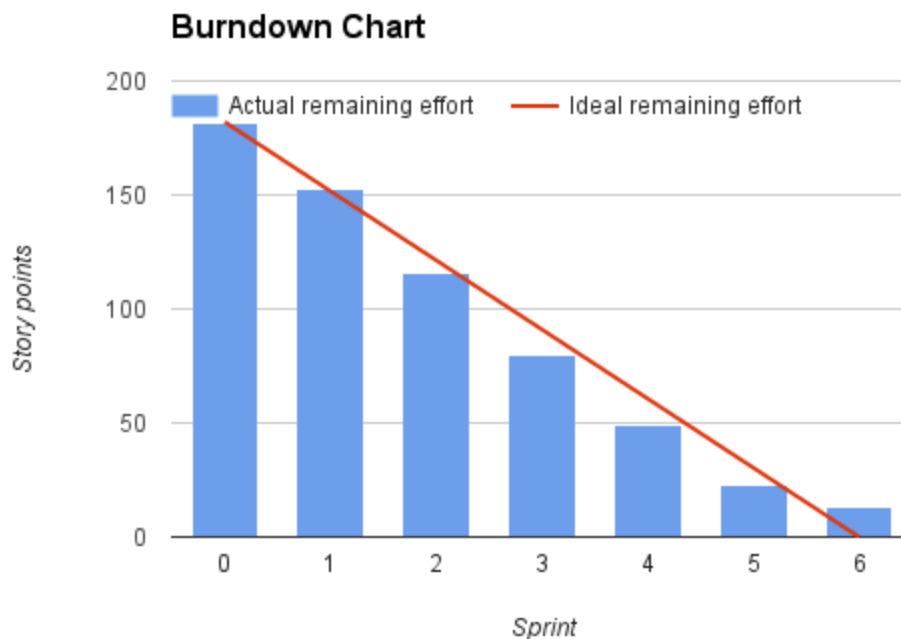


Figure 19. Scrum burndown chart.

The burndown chart shown above shows how the real task development compares to what would have been the ideal development of the project. As observed in the chart, even though the project was extended to last longer than initially planned, tasks were developed in a fairly organized way, distributing them quite evenly throughout the iterations.

Although not included in the burndown chart, we had a couple remaining tasks in the project. After analyzing the cost of these tasks, we decided not to include them in the project as they would require too much time and the project had a time-fixed constraint. These tasks were the ability to interact with tasks while offline and the ability to scan cards and automatically generate digital equivalents.

13 Laws and regulations

13.1 Identification

While developing the project we had to keep in mind any laws and regulations that might affect us; both when writing code as well as when developing the business or treating data. We identified the following affected areas:

13.1.1 Treatment of data

Since our application requires treating and storing personal user data, we have to guarantee users with a minimum protection enforced by the law. This means storing and treating data following the requirements specified in the Ley Orgánica de Protección de Datos (LOPD). The requirements specified can be summarised in the following points:

- Inform users of the personal data being collected, how it will be used and obtain their consent to do so.
- Implement the necessary security measures to protect the data:
 - Guarantee that we won't allow anyone that we haven't previously stated to access the data.
 - Having a weekly updated copy of the data in case something happens to our main copy.
 - Maintain a password policy that enforces an update of all passwords at least once a year.
 - Have an accessible copy of our application's privacy policy always available.
- Register the data files in the Agencia Española de Protección de Datos (AEPD).

13.1.2 Using free software

The development of this project requires the use of open source libraries that come with a license. Each of these licenses state requirements that our project must meet in order for us to legally use them. This means that before publishing the project, we'll have to go through these licenses and ensure that we're meeting all the specified requirements.

Following is a list of licenses included in different open source libraries and packages used in the project that affect our application:

MIT

The MIT License is a permissive free software license that puts very limited restriction on reuse. More specifically, it permits any software under this license to be reused within proprietary software provided a copy of the license terms and a copyright notice is included. This license is also compatible with most copyleft licenses making it a very popular option. [38]

Creative Commons Attribution 4.0 International

This version of the Creative Commons license gives us permission to share and adapt the program as long as appropriate credit is given. In order to comply by this license, we included a Licenses section in our application stating that we are using the open source library. [39]

14 Conclusions

Developing the Digital Business Cards project has been a great way to put into practice all the knowledge acquired during our degree, including planning a project, following a software development methodology and choosing and learning new technologies with which to implement our platform.

14.1 Acquired knowledge

The Digital Business Card project represents the first project that we developed from an idea to a complete final product. Doing this involved defining what we wanted to build, studying the market around it and deciding how and with what technologies we were going to build it. In order to do this, we applied the knowledge acquired during our degree that taught us how to research a project, choose an appropriate methodology to develop it with and successfully carry out the code implementation.

We started our project with an initial idea that we later researched and gathered information to be able to adapt to the needs of a specific target market. While doing this, we gained first hand experience on the importance of conducting a thorough market research and speaking with the different interested parts of the project to fully understand the target market and decide on the best way to develop the idea, what it should include and how we should carry out the implementation depending on the needs of the project.

Working with the Scrum methodology has been very helpful in providing us with ways to maintain continuous contact with the target market, helping us shape our project to meet user expectations while allowing us to develop rapidly. Had we used another methodology, the final result of the project wouldn't have been as successful, specific or time efficient.

Furthermore, developing this project has also helped us practice our ability to abstract the planification of a project from the technologies used to develop it; a concept studied in depth in our software engineering specialization. By doing this we were able to choose our technology in a more accurate way and plan taking into consideration the platform itself and not the tools used to build it.

While developing the project, we also learned about the risks and challenges that using new technologies can represent. We realized that using a technology which we didn't have previous

experience with can imply an unexpected learning curve which can hinder development and that a platform might not be reliable or scalable enough for the dimension of our project. This can be because of the existence of bugs in the tool itself or because it's not optimised or thought out well enough for large projects. Part of these problems were experienced during the development of the mobile application with the Ionic framework which we learned has several bugs on the Android platform. Even though we could work around them during this project, in future projects, it would be interesting to always find some testing with the tool before including it in a project.

14.2 Future development

Overall, the development of the project was successful and met expectations. However, there's more to the platform than the tasks included in the scope of this project.

We plan to continue improving the application once the project is complete, adding useful features such as the previously mentioned offline availability and the ability to auto generate cards from pictures. We also want to extend some of the currently available features such as the card sharing functionality, adding new ways to share cards such as using QR codes or NFC technologies.

14.3 Technical skills

CES1.3: To identify, evaluate and manage potential risks related to software building which could arise. [Fairly]

In order to develop a project, it's essential to plan ahead and evaluate the objectives, scope, market, budget and risks involved. If the project isn't planned, it might not have the budget or resources needed for development. The same logic applies to its risks. If a project's risks aren't analyzed ahead of time, we might not have the necessary resources to take action in the event of unexpected situations, a problem which can lead to project failure.

Before starting the Digital Business Cards project, we carried out some research and planification in which we analyzed the risks involved in the development of the project with the objective of ensuring viability and to be able to determine an action plan in case of deviations or unexpected situations.

CES1.4: To develop, maintain and evaluate distributed services and applications with network support. [A good deal]

The development of the project includes the implementation of a series of services including the server and API, the web viewer and the mobile application. Furthermore, due to initially considering the possibility of developing a synchronization between the server database and a local one in the client, we evaluated a series of databases searching for one that would provide us with an easy way to implement this feature. However, after researching and testing a variety of databases, we found that the task was very time consuming and decided to remove the feature from the scope of this final year project.

CES1.5: To specify, design, implement and evaluate databases. [A good deal]

The project requires storing data, meaning that we had to analyze the project's initial requirements to define what database best fit our needs. Once this was done, we designed and implemented the database in order to start developing the project.

CES1.7: To control the quality and design tests in the software production. [Fairly]

The project will be developed following the Scrum agile methodology, meaning that at the end of each iteration we should be able to deliver a releasable piece of software. That means that with the development of each feature in the iteration we will also consider how well it meets design guidelines, how well it carries out its functionality and develop tests if necessary. As development continues, no code should be added if it breaks previously defined tests meaning that we should always make sure to run any automated tests and carry out any necessary manual ones to ensure that the desired software quality is met.

CES2.1: To define and manage the requirements of a software system. [Fairly]

Before implementing the project, we will develop the necessary documentation to ensure the project's viability. Moreover, in order to develop a successful product, we will carry out a market research to find out what the stakeholders want. We will then use the collected information to specify the project's requirements and adapt them throughout the project according to the feedback received.

References

- [1] FIB UPC. 2016. *Facultad de informática de Barcelona*. [ONLINE] Available at: <http://www.fib.upc.edu/fib.html>. [Accessed 28 February 16].
- [2] Wikipedia. 2016. *Stakeholder (corporate)*. [ONLINE] Available at: [https://en.wikipedia.org/wiki/Stakeholder_\(corporate\)](https://en.wikipedia.org/wiki/Stakeholder_(corporate)). [Accessed 28 February 16].
- [3] Moya K. Mason. 2016. *Worldwide Business Start-Ups*. [ONLINE] Available at: <http://www.moyak.com/papers/business-startups-entrepreneurs.html>. [Accessed 29 February 16].
- [4] CamCard. 2016. CamCard. [ONLINE] Available at: <https://www.camcard.com/>. [Accessed 01 March 16].
- [5] CamCard Free - Business Card R. 2016. Android Application on Google Play. [ONLINE] Available at: <https://play.google.com/store/apps/details?id=com.intsig.BCRLite&hl=en>. [Accessed 01 March 16].
- [6] Digital Business Cards by Inigo. 2016. Inigo. [ONLINE] Available at: <https://inigoapp.com/>. [Accessed 01 March 16].
- [7] Business Cards. 2016. Android Application on Google Play. [ONLINE] Available at: https://play.google.com/store/apps/details?id=com.inigoapp.inigoandroid&utm_source=Inigo%20Home%20Page&utm_medium=referral&utm_term=Android%20HomePage%20Button&utm_content=Homepage%20Android&utm_campaign=Homepage%20Buttons. [Accessed 01 March 16].
- [8] Haystack. 2016. Haystack - Your new business card. [ONLINE] Available at: <http://thehaystackapp.com/>. [Accessed 01 March 16].
- [9] Haystack Digital Business Card. 2016. Android Application on Google Play. [ONLINE] Available at: https://play.google.com/store/apps/details?id=com.theHaystackApp.haystack&referrer=utm_source%3DhaystackWebsite%26utm_medium%3Dsplash. [Accessed 01 March 16].
- [10] Ionic. 2016. *Ionic*. [ONLINE] Available at: <http://ionicframework.com/>. [Accessed 9 October 2016].
- [11] React Native. 2016. *React Native*. [ONLINE] Available at: <https://facebook.github.io/react-native/>. [Accessed 9 October 2016].
- [12] jQuery mobile. 2016. *jQuery mobile*. [ONLINE] Available at: <https://jquerymobile.com/>. [Accessed 9 October 2016].

- [13] NativeScript. 2016. *NativeScript*. [ONLINE] Available at: <https://www.nativescript.org/>. [Accessed 9 October 2016].
- [14] QAT Global. *Agile Meets Scrum in QAT Global's Enterprise Development Framework*. [ONLINE] Available at: <http://www.qat.com/agile-scrum-methodology/>. [Accessed 01 March 16].
- [15] PagePersonnel. 2016. Estudios de remuneración 2016. [ONLINE] Available at: http://www.pagepersonnel.es/sites/pagepersonnel.es/files/er_tecnologia16.pdf. [Accessed 15 March 16].
- [16] AppsFlyer. 2016. *The State of In-App Spending*. [ONLINE] Available at: http://hub.appsflyer.com/hubfs/IAP_Guide/The_State_of_In-App_Spending_AppsFlyer.pdf. [Accessed 8 October 2016].
- [17] Lily Hay Newman. 2014. *Android Users Won't Drop Money on Just Any App*. [ONLINE] Available at: http://www.slate.com/blogs/future_tense/2014/06/26/there_are_twice_as_many_android_users_as_ios_but_ios_users_spend_double.html. [Accessed 8 October 2016].
- [18] Npmjs. 2016. *Express*. [ONLINE] Available at: <https://www.npmjs.com/package/express>. [Accessed 1 October 2016].
- [19] GitHub. 2016. *expressjs/express*. [ONLINE] Available at: <https://github.com/expressjs/express>. [Accessed 1 October 2016].
- [20] Npmjs. 2016. *Mysql*. [ONLINE] Available at: <https://www.npmjs.com/package/mysql>. [Accessed 1 October 2016].
- [21] GitHub. 2016. *mysqljs/mysql*. [ONLINE] Available at: <https://github.com/mysqljs/mysql>. [Accessed 1 October 2016].
- [22] Npmjs. 2016. *Bcrypt*. [ONLINE] Available at: <https://www.npmjs.com/package/bcrypt>. [Accessed 1 October 2016].
- [23] GitHub. 2016. *kelektiv/node.bcrypt.js*. [ONLINE] Available at: <https://github.com/kelektiv/node.bcrypt.js>. [Accessed 1 October 2016].
- [24] Npmjs. 2016. *Jsonwebtoken*. [ONLINE] Available at: <https://www.npmjs.com/package/jsonwebtoken>. [Accessed 1 October 2016].
- [25] GitHub. 2016. *auth0/node-jsonwebtoken*. [ONLINE] Available at: <https://github.com/auth0/node-jsonwebtoken>. [Accessed 1 October 2016].

- [26] Npmjs. 2016. *Nodemailer*. [ONLINE] Available at: <https://www.npmjs.com/package/nodemailer>. [Accessed 1 October 2016].
- [27] GitHub. 2016. *nodemailer/nodemailer*. [ONLINE] Available at: <https://github.com/nodemailer/nodemailer>. [Accessed 1 October 2016].
- [28] JWT. 2016. *Introduction to JSON Web Tokens*. [ONLINE] Available at: <https://jwt.io/introduction/>. [Accessed 10 October 2016].
- [29] Wikipedia, the free encyclopedia. 2016. *HTTPS*. [ONLINE] Available at: <https://en.wikipedia.org/wiki/HTTPS>. [Accessed 10 October 2016].
- [30] Wikipedia, the free encyclopedia. 2016. *Model–view–controller*. [ONLINE] Available at: <https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller>. [Accessed 10 October 2016].
- [31] Mocha. 2016. *Mocha*. [ONLINE] Available at: <https://mochajs.org/>. [Accessed 18 October 2016].
- [32] Chai. 2016. *Chai*. [ONLINE] Available at: <http://chaijs.com/>. [Accessed 18 October 2016].
- [33] Wikipedia, the free encyclopedia. 2016. *Black-box testing*. [ONLINE] Available at: https://en.wikipedia.org/wiki/Black-box_testing. [Accessed 20 October 2016].
- [34] Wikipedia, the free encyclopedia. 2016. *SQL injection*. [ONLINE] Available at: https://en.wikipedia.org/wiki/SQL_injection. [Accessed 3 October 2016].
- [35] Google Inside Search. 2016. *Crawling & Indexing*. [ONLINE] Available at: <https://www.google.co.uk/insidesearch/howsearchworks/crawling-indexing.html>. [Accessed 4 October 2016].
- [36] Google Developers. 2016. *Robots meta tag and X-Robots-Tag HTTP header specifications*. [ONLINE] Available at: <https://www.google.co.uk/insidesearch/howsearchworks/crawling-indexing.html>. [Accessed 4 October 2016].
- [37] Wikipedia, the free encyclopedia. 2016. *Brute-force attack*. [ONLINE] Available at: https://en.wikipedia.org/wiki/Brute-force_attack. [Accessed 4 October 2016].

[38] Wikipedia, the free encyclopedia. 2016. *MIT License*. [ONLINE] Available at: https://en.wikipedia.org/wiki/MIT_License. [Accessed 12 October 2016].

[39] Creative Commons. 2016. *Creative Commons Attribution 4.0 International (CC BY 4.0)*. [ONLINE] Available at: <https://creativecommons.org/licenses/by/4.0/>. [Accessed 18 October 2016].

Acronyms

API - Application Program Interface

CSS - Cascading Style Sheets

HTML - HyperText Markup Language

HTTP - HyperText Transfer Protocol

HTTPS - HyperText Transfer Protocol Secure

JS - JavaScript

JSON - JavaScript Object Notation

MVC - Model - View - Controller

NPM - Node Package Manager

ODBC - Open Database Connectivity

REST - Representational State Transfer

SQL - Structured Query Language

SSL - Secure Sockets Layer

UML - Unified Modeling Language

URL - Uniform Resource Locator

UUID - Universally Unique Identifier

XSS - Cross-Site Scripting

Glossary

API - An Application Programming Interface (API) is a particular set of rules and specifications that a software program can follow to access and make use of the services and resources provided by another particular software program that implements that API.

ODBC - An Open Database Connectivity (ODBC) is a standard application programming interface (API) for accessing database management systems (DBMS). The designers of ODBC aimed to make it independent of database systems and operating systems.

XSS - Cross-Site Scripting (XSS) attacks are a type of injection in which malicious scripts are injected into an otherwise trusted website. These attacks normally occur when a malicious user sends malicious code (usually in the form of scripts) to another end user.

Appendix A - User stories

The user stories involved in the application were developed in the following iterations. Each user story has a number attached that indicates the story points that each story represents.

First iteration

[8] As a user I want to be able to register in the application.

[8] As a user I want to be able to login into my account in order to start using the application.

[8] As a user I want to be able create a personal card.

[5] As a user I want to be able to view my personal cards in order to see the cards that I have created.

Total story points: 29

Second iteration

[8] As a user I want to be able to add links to a card.

[8] As a user I want to be able to add telephone numbers to a card.

[13] As a user I want to be able to add an image to my card.

[8] As a user I want to be able to edit my cards in order to change the information on them.

Total story points: 37

Third iteration

[5] As a user I want to be able to share my cards with other users of the application.

[13] As a user I want to be able to share my cards with people that don't have accounts in the application.

[8] As a user I want to be able to delete my personal cards so that no one else can access them.

[5] As a user I want to be able to accept an invite from a user that wants to share his card with me.

[5] As a user I want to be able to reject an invite from a user that wants to share his card with me.

Total story points: 36

Fourth iteration

[5] As a user I want to be able to create groups in order to organize my cards.

[3] As a user I want to be able to view the groups that I have created.

[5] As a user I want to be able to add cards to previously created groups.

[5] As a user I want to be able to delete cards from groups.

[8] As a user I want to be able to delete cards that have been shared with me so that they no longer appear in my list of cards.

Total story points: 31

Fifth iteration

[5] As a user I want to be able to delete groups that I created in the past.

[13] As a user I want people who don't have accounts with the application to be able to view the cards I've shared with them.

[8] As a user I want to be able to change the password of my account.

Total story points: 26

Sixth iteration

[5] As a user I want to be able to change my account's primary email.

[5] As a user I want to be able to change my account's username.

[13] As a user I want to be able to change my account's profile image.

Total story points: 23

Tasks to be developed in future iterations

[55] As a user I want to be able to access and make changes to the data while i'm offline.

[34] As a user I want to be able to take a picture of a physical business card so that it automatically creates its digital equivalent.

Appendix B - User manual

This manual will act as a small guide to help users understand what features the application offers, learn how to use them and how to navigate through the different screens.

Account creation and access

The first thing we stumble upon when opening the application is the Login screen. This screen lets us insert our username and password in order to access our account. Contrarily, if we don't have access to our account or we need to create one, it lets us reset our password or links us to the Register screen. The Register screen is very similar to the Login one except it asks you for your email as well as your chosen username. If we were to change our password, we would have to insert our username or email as shown in the screenshot below and we would receive an email with an unique link where we can change our account's password.

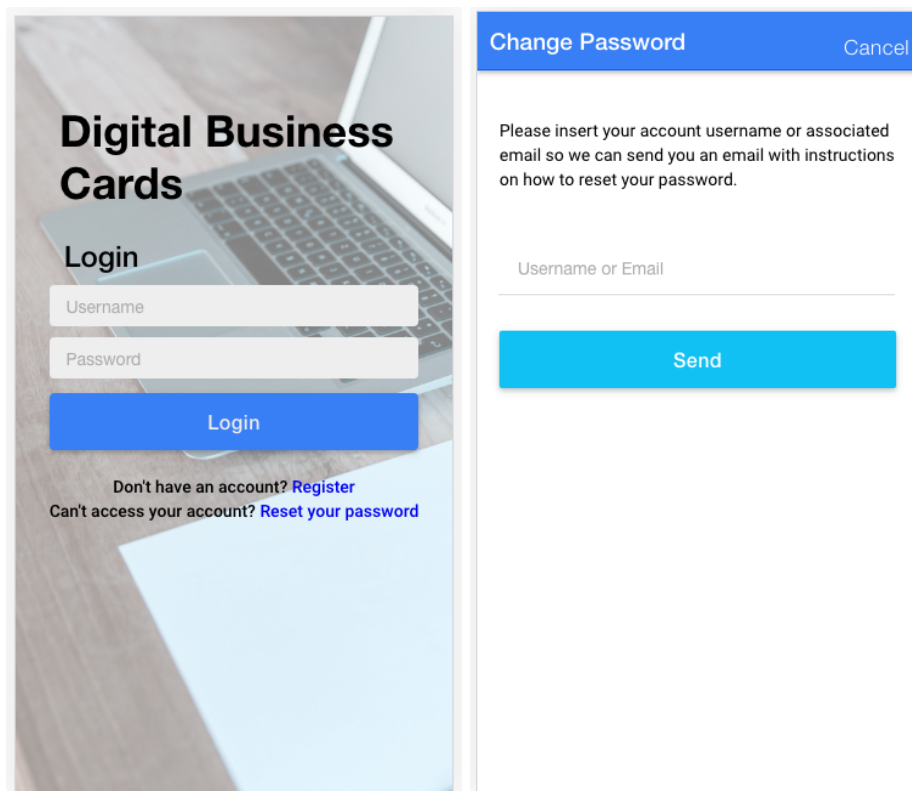


Figure 20. Application Login and reset password screens.

Home screen

Once we access the application, we're greeted with our main screen which is a list of our personal cards. This screen is made up of a tabbed view that lets us quickly access our All cards group which is made up of all our cards; both our personal ones as well as any other cards that have been shared with us. If we touch the hamburger icon on the left side of the navigation bar, we can access the menu.

The floating button on the personal tab of the home screen allows us to create a new card. In the all tab, the floating button acts as a shortcut to the search bar used to search for a card in between all the personal and received cards.

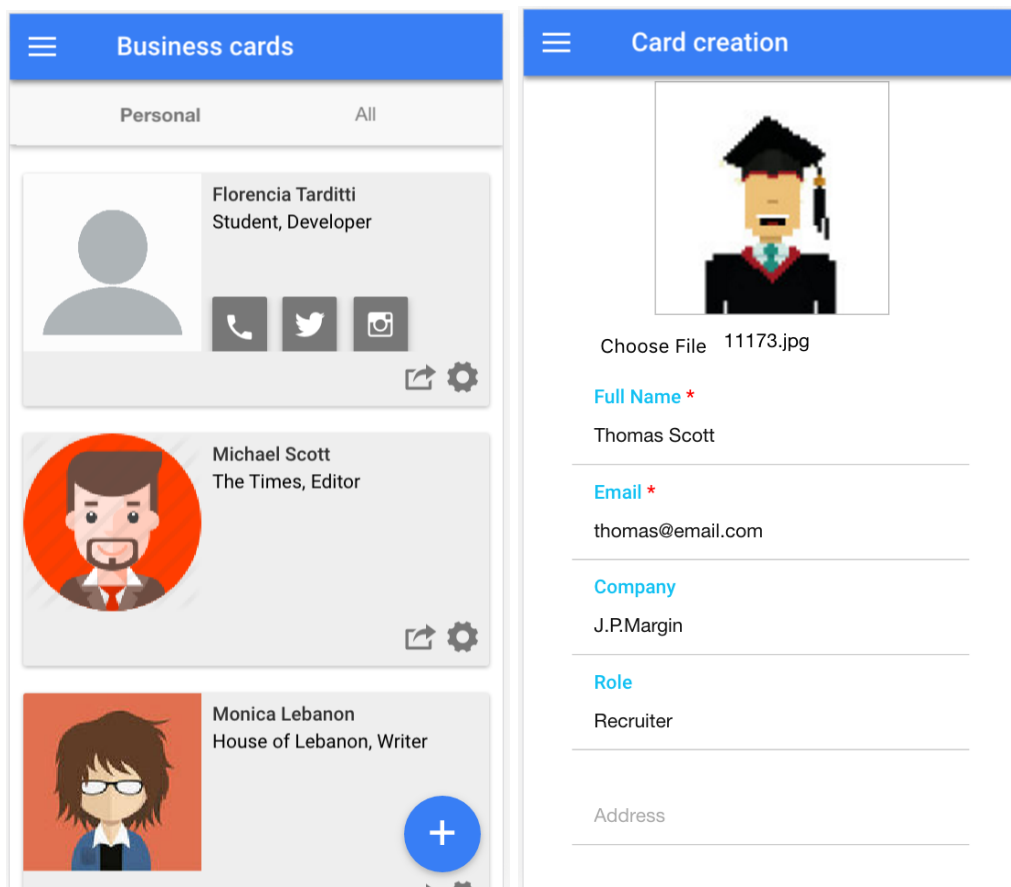


Figure 21. Personal cards view and card creation screen.

Card interaction

Whenever a card is being showed, clicking on it will always take you to an expanded view of the card where you can see a detailed view of the information. When it comes to using the sharing and settings buttons, they won't be shown on all screens and will have slightly different functionalities depending on what screen you're on. Whenever you're seeing your personal cards, you will have the opportunity both to share and use the settings, however, it should be considered that the delete card button when viewing a card that's owned will delete the card both from the user and from anyone with whom it has been shared. On all other views, the sharing button won't be available and the delete button on the settings will only delete our own version of the card (not the original card).

When sharing one of our cards, it is required for the email associated to the card to be verified. If the email is not verified, we would get a notice that will prompt us if we want to resend the verification email.

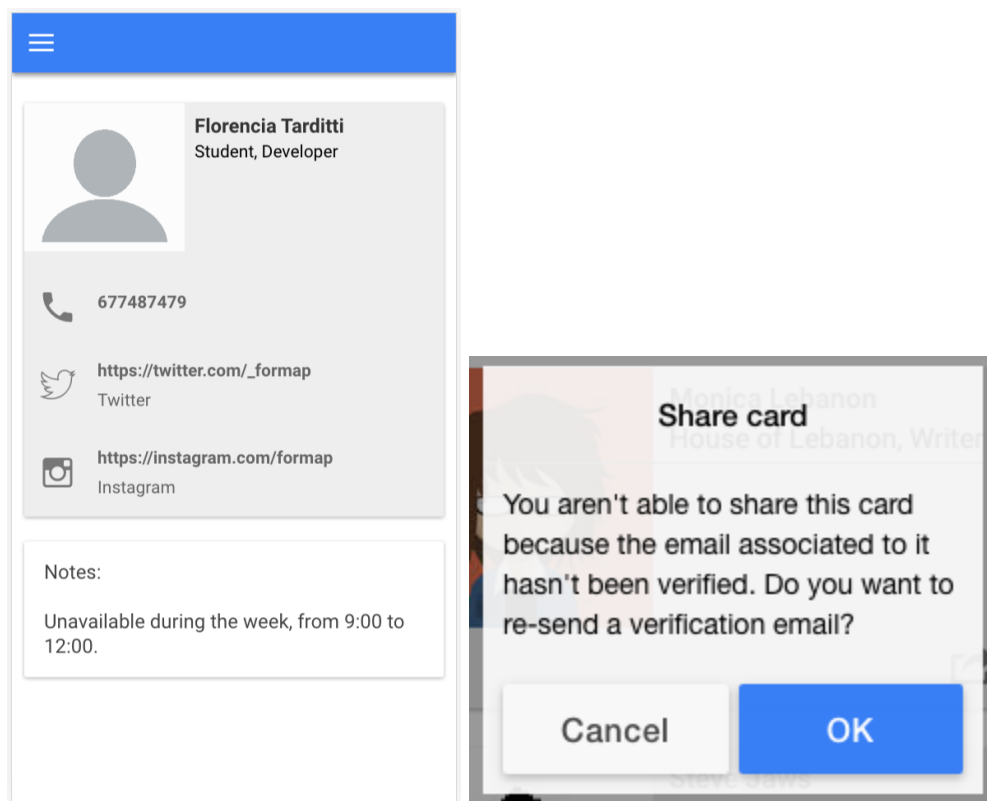


Figure 22. Card expanded view and sharing card notice.

Navigation menu

The navigation menu can be opened using the hamburger icon found on the navigation bar in all main screens. This menu gives you access to all the main parts in the application.

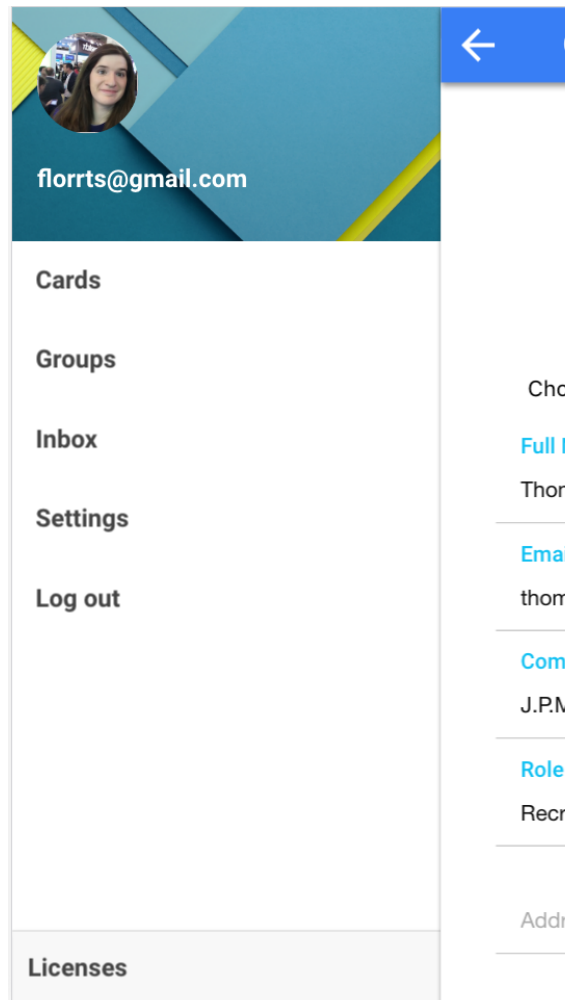


Figure 23. Navigational menu.

Groups screen

The groups screen shows a list of all the groups that the user owns. The floating button on the bottom right of the screen lets you create a new group and clicking on any item in the list will let you view a list of all the cards in the group.

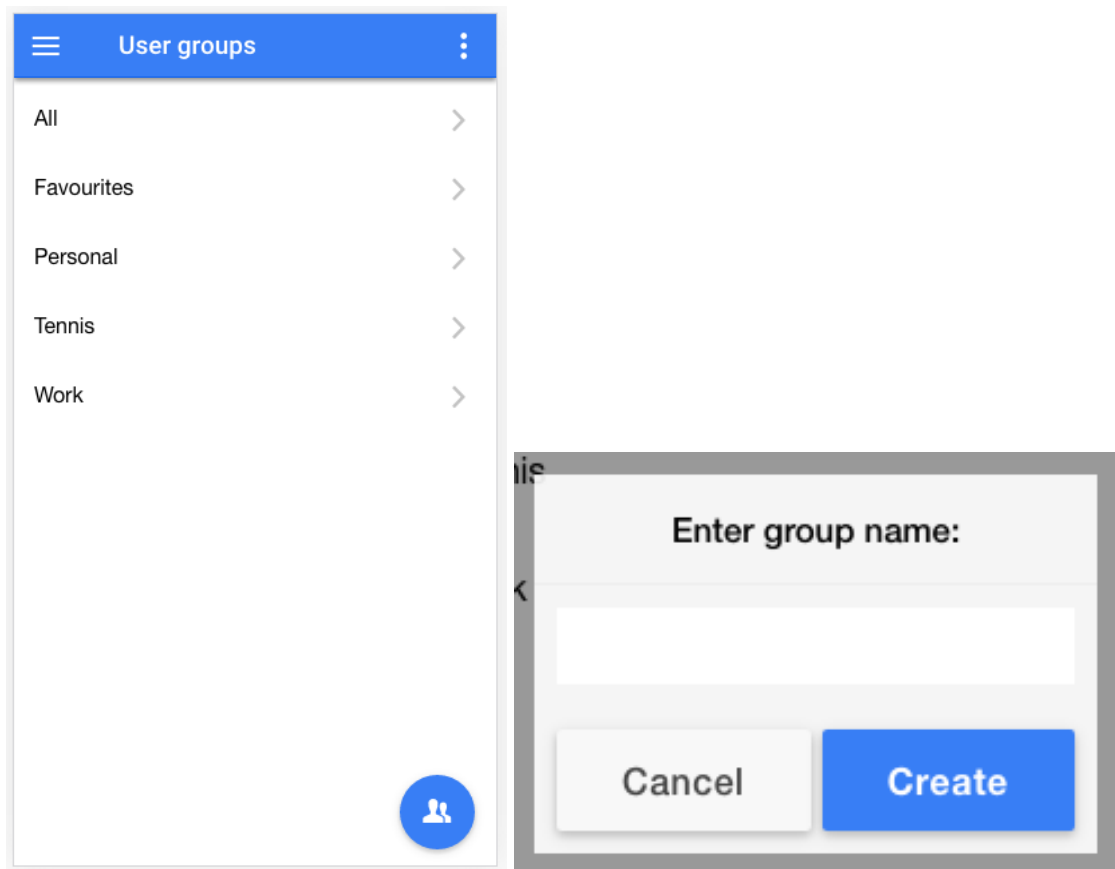


Figure 24. Group creation and viewing.

Inbox screen

The inbox is where all card invitations arrive. Whenever a user shares a card with you, you will receive an invitation in your inbox that you can either accept or reject. If you accept a card from the inbox, this one will be added to your All cards group which you can access from the home or groups screens. If you reject an invitation, it will disappear and you won't be enquired about this card any further.

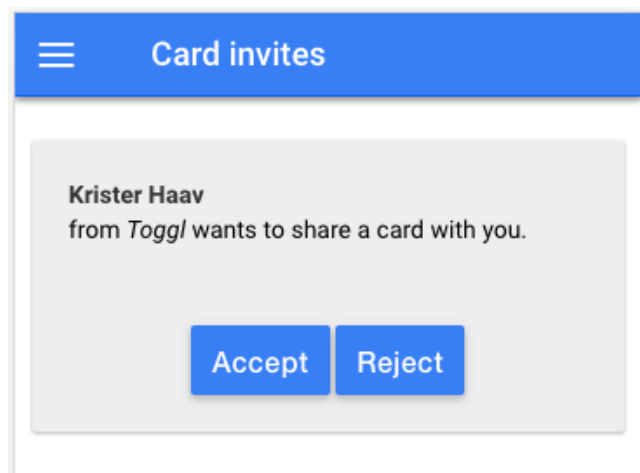
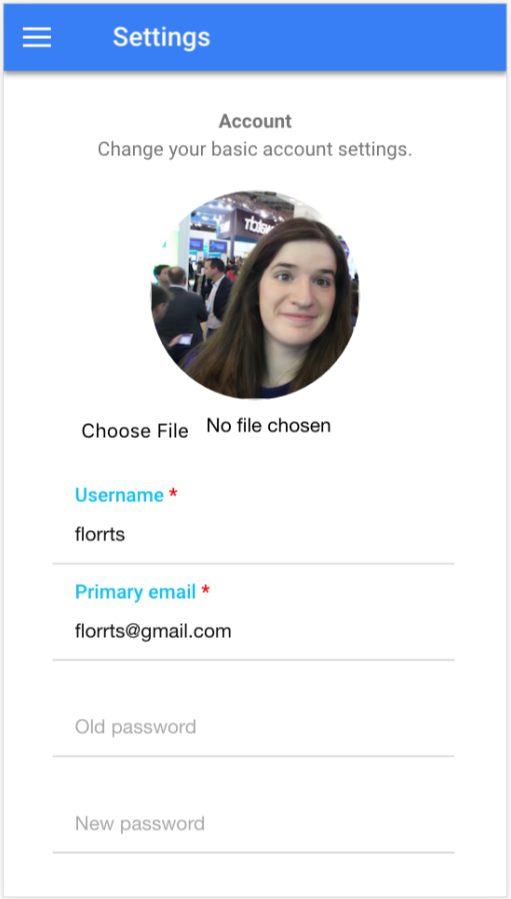


Figure 25. Inbox view with any pending invitations.

Account settings

The settings screen lets the user change his account information and save it. It also lets the user delete his account, were he willing to do so.




The image shows a mobile application's 'Settings' screen. At the top is a blue header with a white hamburger menu icon and the word 'Settings'. Below the header, the section is titled 'Account' with the subtitle 'Change your basic account settings.' In the center is a circular profile picture of a woman with long brown hair. Below the picture, there is a text label 'Choose File' followed by 'No file chosen'. Below this are four input fields, each with a label and a red asterisk indicating a required field: 'Username' with the value 'florrts', 'Primary email' with the value 'florrts@gmail.com', 'Old password', and 'New password'.

Settings

Account

Change your basic account settings.



Choose File No file chosen

Username *

florrts

Primary email *

florrts@gmail.com

Old password

New password

Figure 26. User account settings.

Web renderer

The web renderer allows non-users of the applications to view cards created inside the application. It can be loaded on any modern browser and is responsive.

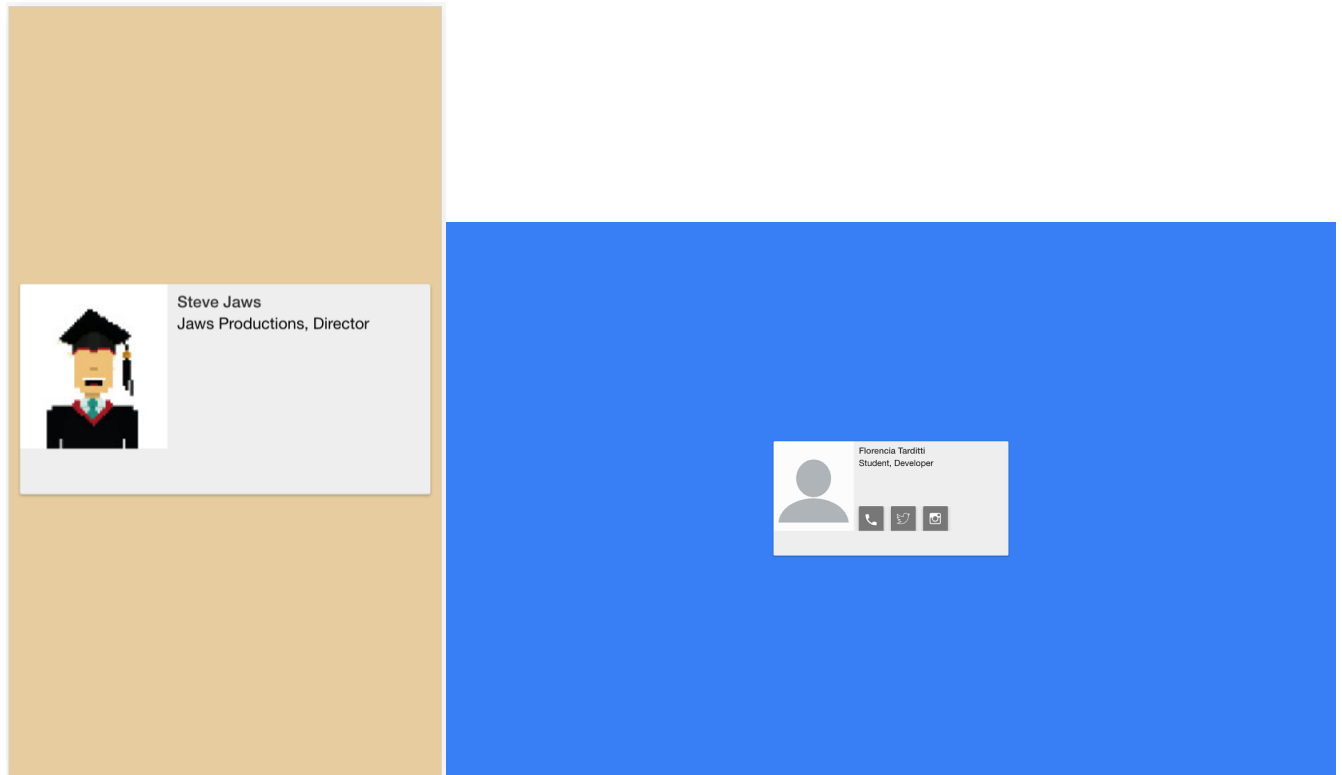


Figure 27. Card renderer shown on different devices.

