UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

WHERE IS THE MATCH?

A Degree Thesis

Submitted to the Faculty of the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

by

Manuel Alba Avilés

In partial fulfilment

of the requirements for the degree in

SCIENCE AND TELECOMMUNICATION TECHNOLOGIES ENGINEERING

Advisor: Jordi Fornes de Juan

Barcelona, May 2015

Abstract

WhereIsTheMatch is a very ambitious project that consists of developing my own social network mobile app for Android. This app is focused on easily finding people and places to play sports.

The main idea of this project is create a first step to a final product and my own developing style. I knew from the beginning that this would be a very hard and a very demanding task because of the several areas that are involved in the development of a mobile app, such as database implementation, interface design or back-end software programming.

But I wanted to test my skills and to gain knowledge in a field that was very unknown to me. The first step is always the most difficult, but everything must start with a first step.

Resum

WhereIsTheMatch és un projecte molt ambiciós que consisteix en el desenvolupament de la meva pròpia xarxa social en una aplicació per a Android. Aquesta aplicació té com a objectiu fer més fàcils la tasca de trobar persones i llocs on practicar esports.

L'idea principal d'aquest projecte és crear una primera aproximació a la que seria l'aplicació final i la cerca del meu propi estil com a desenvolupador. Era conscient des del primer moment que seria un treball molt complicat degut a les diferents branques que incorporava, així com coneixement de bases de dades, disseny d'interfícies o programació de software per controlar el comportament de l'aplicació.

De tota manera, volia comprovar de que era capaç i guanyar coneixements en uns camps bastant desconeguts per a mi. El primer pas sempre és difícil, però tot ha de començar amb un primer pas.

Resumen

WhereIsTheMatch es una proyecto muy ambicioso que consiste en el desarrollo de mi propia aplicación para Android, una red social. Ésta tiene como función facilitar el contacto entre personas que buscan lugares y otros compañeros para practicar deporte.

La idea principal de este proyecto es conseguir una primera aproximación a la que sería la aplicación final y la búsqueda de mi propio estilo como desarrollador. Era consciente desde el principio que sería un trabajo muy complicado debido a las diferentes áreas que incorporaba, así como el conocimiento de bases de datos, diseño de interfaces o programación del software para controlar la aplicación.

Pese a esto, quería comprobar de que era capaz y aprender en unos campos que eran bastante desconocidos para mí. El primer paso siempre es el más difícil, pero todo empieza con un primer paso.

I want to thank all the people that has supported me during these hard years. It was very encouraging to have people on my side who always believed in me.

"You must expect great things of yourself before you can do them"

Michael Jordan

Acknowledgements

First of all, I want to thank my tutor Jordi for always give me support and help me in the different areas of my project. I also want to specially thank my friend Vidi for the pieces of advice that he gave me during the designing phase. Finally, I want to thank my family and friends who helped me in the everyday things. Without all of them this could not have been possible.

Revision history and approval record

Revision	Date	Purpose
0	11/05/2015	Document creation
1	15/05/2015	Document revision

DOCUMENT DISTRIBUTION LIST

Name	e-mail
Manuel Alba Avilés	92manel@gmail.com
Jordi Fornés	jfornes@ac.upc.edu

Written by: Manuel Alba		Reviewed and approved by: Jordi Fornes	
Date	11/05/2015	Date	15/05/2015
Name	Manuel Alba Avilés	Name	Jordi Fornes de Juan
Position	Project Author	Position	Project Supervisor

Table of contents

The table of contents must be detailed. Each chapter and main section in the thesis must be listed in the "Table of Contents" and each must be given a page number for the location of a particular text.

List of Figures

List of Tables:

1. Introduction

This is my own project, which is called "Where is the match?". I got the idea after several years of playing sports and a few years here in university.

Have you ever tried to meet your old friends to play football or other sports? I remember when I was studying high school that in order to play a football match I only needed to speak with one or two friends and easily after school we met at the park and we played. Just as easy as that. Nowadays I need almost a miracle so as to even play a tennis match. And the worst situation is when you almost get to it, but finally cannot play. That moment when you need 10 people to play football but finally you and your friends only manage to be 8 and you have to cancel the meeting.

But now is not as difficult as before to find people to play sports with. We have different ways like FB (Facebook) or TWTR (Twitter) that allow us to contact with a lot of people and invite them to join our sport meeting. The problem is that in these social networks we have trouble to identify who can be a good candidate. "Would like this man/woman to come?" or "It is been a while since I talked to him/her and I do not know if he/she likes this sport" are common thoughts that could make someone to not invite other people and finally the match will not be played. It would be much easier if we could ask these questions in a social network focused on this kind of activities.

But there are lots of social networks in Google Play or in App Store that can do this". Yes, I agree, but it seems they really do not help us in the task of meeting people in order to play sports. Because of the poor number of users, the design, the functionalities, the poor integration with other social networks... And I want to change that. I want to solve my own problems and also help other people who I think they can have similar problems. I want to develop a mobile app which changes the way you organize an sport meeting onward.

I expect to at least define a clear way in order to get to that point, and the in a near future be able to upload and enjoy this app. And I want to use this project to be my first step so as to one day be a good developer.

2. State of the art

In this section I am going to explain all the knowledge that a developer must have before the project to effortlessly do it. There are several areas involved in the development of a mobile app and all of them are essential.

2.1. Programming languages

Let's begin with the basics. In order to develop software it is essential to know some programming languages that must be used. This was an strong reason to decide in which OS (Operating system) will be developed the project. Later or sooner it must be developed for the most extended OS's but in this period of time I have to start with one.

So thanks to the background in **Java** achieved during the degree and a few other projects that I have developed in Android, Google's OS, I decided to do my first version in **Android**. Java is also a good choice due to the fact that software built in Java can be used in every OS that can install the JVM (Java Virtual Machine) so the generated coded will easily be ported to other OS.

2.1.1. Front-end development

The front-end development consists of the part of the project that is going to be shown to the user.

The user interface is as important as the app's behaviour because is the first thing an user perceives about your app. One bad front-end development design translates directly into users uninstalling your app so it is really something to take into account. So as to develop the front-end I used **android native code**, which is similar to XML (Extensible markup language) /HTML (Hypertext markup language) and CSS (Cascading style sheets) but personalized for android devices and components.

My background in this area was very limited, I just only coursed one subject that introduced a bit of HTML and CSS, so I had to investigate a lot and learn lots of things about these languages.

2.1.2. Back-end development

The back-end development consists of the brain of the app. It is the part that controls everything, since which screen is going to be the next to which action is going to be triggered when the user clicks anything.

An efficient back-end development translates directly in less time to make actions happen and obtain the desired actions at each moment. It also makes the difference to do a good **UML (Unified Modeling Language) design** beforehand and then starting the programming. It helps an engineering or programmer to acknowledge the relation between the different classes of the app, their attributes, the relation between their methods...etc.

2.1.2.1. Application's behaviour

The application's behaviour is the back-end development subsystem that controls the interaction of the user and the app. This part decides if an action has been executed correctly and therefore which path is going to follow the app in the different use cases that an app could confront in different scenarios.

Google recommends the developers to user their SDK (Software development kit), NDK (Native development kit) or ADK (Accessory development kit) to build android application. NDK is a toolset that is thought to control low-level features such as CPU (Central processing unit) workloads. ADK is a toolset that is thought to build and connect different hardware to an android device. So after this brief analysis, the only available options are NDK and SDK. Due to the non-intensive CPU workloads that the app is going to use and the unnecessary access to low-level features, I decided to work with the **SDK**.

2.1.2.2. Database

The database system management is a back-end subsystem that works as an independent software module from the Application's behaviour, acting as a complement of it. However, it is a very important part in social networks such as Where is the match.

A database system management is arranged in tables with rows and columns where the data is stored, so the better the design is, the less times you must access to the database and the most efficient is the use of your device resources such as the processor and the RAM (Random-access memory). In addition, in order to protect the user information, you must develop method and parameters that risk no private information during the access, and that only risk temporary information that change every time one starts the application such as a session ID (Identifier).

Due to the good integration of SQL (Structured query language) language with Android and a bit of practice in the field in one subject, I decided to do a first approximation using **SQLite** database. First a local database will be used to do all the testing, later a different solution such as mySQL or postgreSQL will be implemented so as to access to the information in a web service.

2.2. <u>Integrated development environment (IDE)</u>

The IDE (integrated development environment or interactive development environment) is the software application which is used by developers to build software. It is the factory of my project.

There are several IDEs that could be used so as to build an android app, but due to the constraints imposed before (such as using the Android SDK and build the project in Java), I choose from 3 options:

- AppInventor: designed for novices and people with little knowledge or experience in programming, AppInventor offers an IDE that is based on modules or blocks for Android devices. Moreover, you do not need extra software installed in your computer in order to develop; AppInventor is a web service that store the final app there and then you can downloaded it to your smartphone and even publish it in the Google Play.

- Eclipse: a very well-known IDE that I have used before in several subjects at university. It allows a pretty good integration with Android SDK and is an environment that I am familiar with. In addition, since last year, Eclipse was the first recommendation of Google to build apps in its system.

- Android Studio: a new IDE created by Google specifically to build apps in its system. Google took a few ideas after the android developers feedbacks using Eclipse these year, so the company decided to build an oriented IDE to its mobile OS. Based on the IDE Intellij IDEA, a top-level IDE as Eclipse, is the main option nowadays to develop apps for Android.

AppInventor is a great choice, but it is too simple to build a social network with all its functionalities. So the decision was between Eclipse or Android Studio. Finally, I decided to follow the Google's recommendation and begin the project in **Android Studio**. It has a better debbuger, a better way to distribute more app versions, and a more optimized and improved designing tools to create the interface.

I do not take into account the chance to use other really good IDEs such as Xamarin, that allow one to build an entire app in C# and then compiling it to be used in the most common OS's like iOS (Iphone operating system), Android, Windows, Mac and even more. I think that is not balanced the big effort you must do to build an entire app in C# against the possibility of having your app running in different OS's.

2.3. Social network functionalities

Another important point is being aware of the capabilities of my project. I am concerned that it is going to be a social network, but that is not enough. There is the need to define the actions an user would be able to use once the app is finished. Investigating and analyzing other social networks we can identify some common features that any social network must have:

- Managing users accounts: create account, edit account, delete account, login, logout and password recovery.

- Managing events: create event, edit event, delete event, join event, abandon event and rate an event.

- Managing users' configuration: profile, public profile, private profile, edit profile.

- Managing search: users and events search using key words.

There is more information about the social networks functionalities in the document Other Social Network analysis (**Appendices 2.**).

2.4. Marketing

The concept of marketing in the project is importantly connected to the front-end and back-end development. I had to study the market and in particular which social networks are being successful nowadays beforehand so as to improve their weaknesses and maintain their strengths related to both their functions and their interface.

Not having a very common feature of other social networks (such as changing your profile picture) could be translated into users uninstalling the app because of the missing action/option. The same happens if your interface is too different or too complex. If one user is not capable of knowing how an app works in a few minutes they would probably uninstall it, even if they really need it.

Something very important is also decide if an app should be paid or free. For example, it is obvious nowadays that a greater number of users would download earlier an app if it is free than if they must pay to download it. Even if the free app's quality is worse than the paid app. But that does not mean the better solution is uploading a free app. I think both models can coexist or even a good option between paid or free apps is also good (the called pay-in-app apps, an app that is free to download but to get some features you must pay).

A more extended analysis is available in the document Other Social Network Analysis (**Appendices 2.**).

2.5. Hardware and devices

To properly develop a project with these characteristics, there are some components that are indispensable: a computer where IDE is installed to start developing the project and an android device where I can test the app an see its evolution. Two test devices where used in the project: the Android studio emulator for standard devices and my own mobile phone, One plus one. To ensure a most trustful running I must test it on other real devices.

3. Design and analysis

In this section, I am going to define the whole project implementation. As said before, the project is divided in mainly 3 areas, which each one can be developed separately.

3.1. Interface

The interface design is complicated. It cannot be completely different from other social networks because it will bring the user to a permanent confusion due to the comparisons with the others and the learning curve will be difficult. So I decided to combine the best of the most popular social networks and try to user Material Design to the result.

I noticed during the realization of the document Other Social Network Analysis (**Appendices 2.**) that every social network has a **main colour** apart from black and white. FB, for example, dark blue, TWTR light blue or G+ (Google plus) light red. So I decided to use purple as my primary colour, to easily identify the app due to its purple marks in the screen/components.

Then I started to think in which **uses cases** will be needed:

- Splash: shows a few seconds the developer signature (usually a logo) and the main app icon.

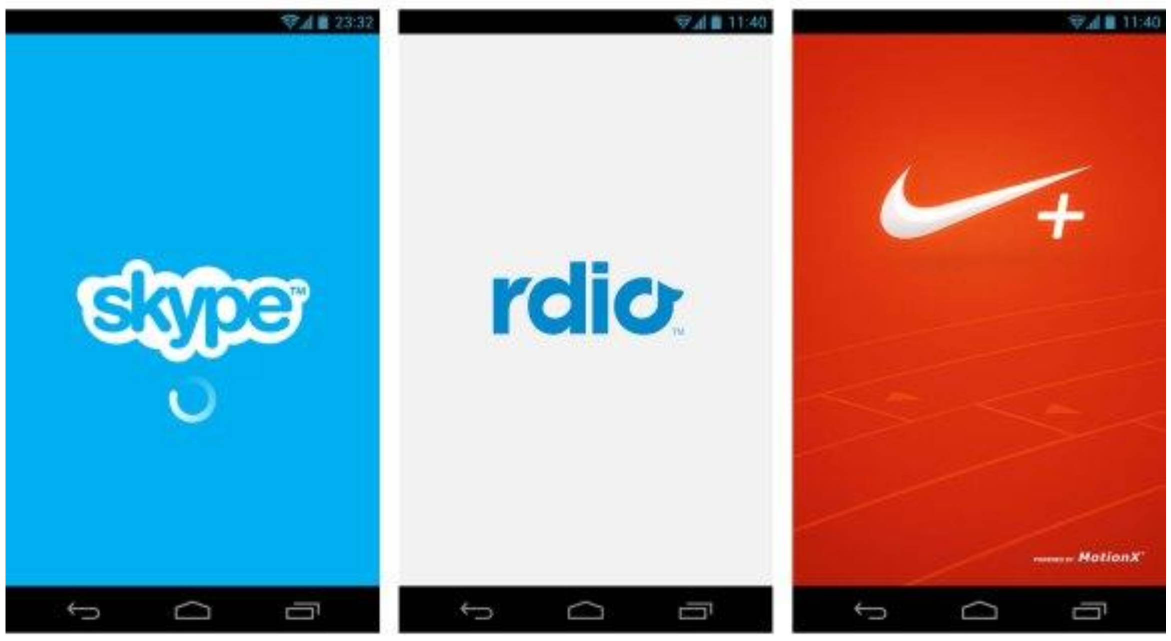


Fig. 1 Splash screen example

- Login: shows the common screen where a user can enter to the app.

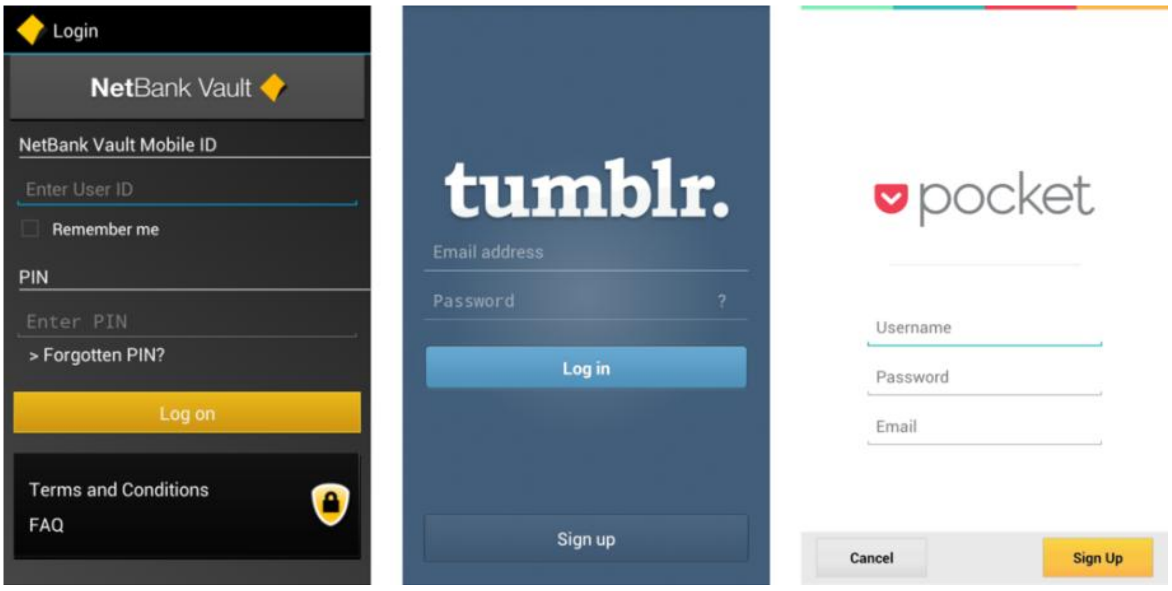


Fig. 2 Login screen example

- Recover password: shows the common screen where a user can recover their password.

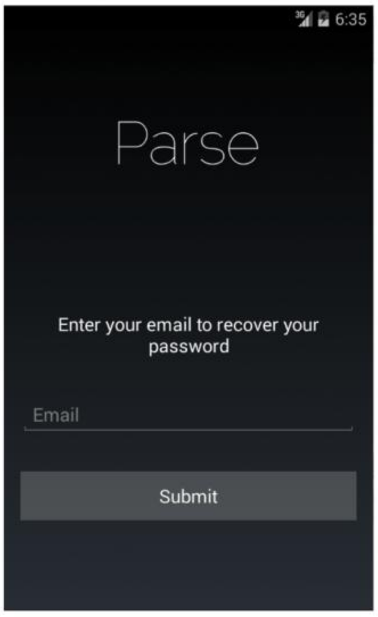


Fig. 3 Password recovery screen example

- New user: shows the common screen where a user can create an account.

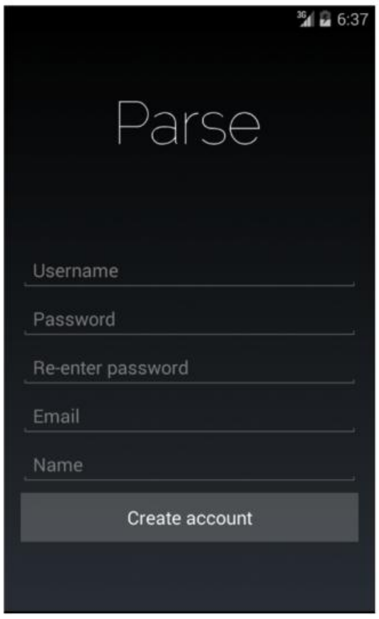


Fig. 4 Create user screen example

- Timeline: shows the matches available. The style is very similar to the one used in FB, TWTR or Google Now, using independent cards as an information unit.

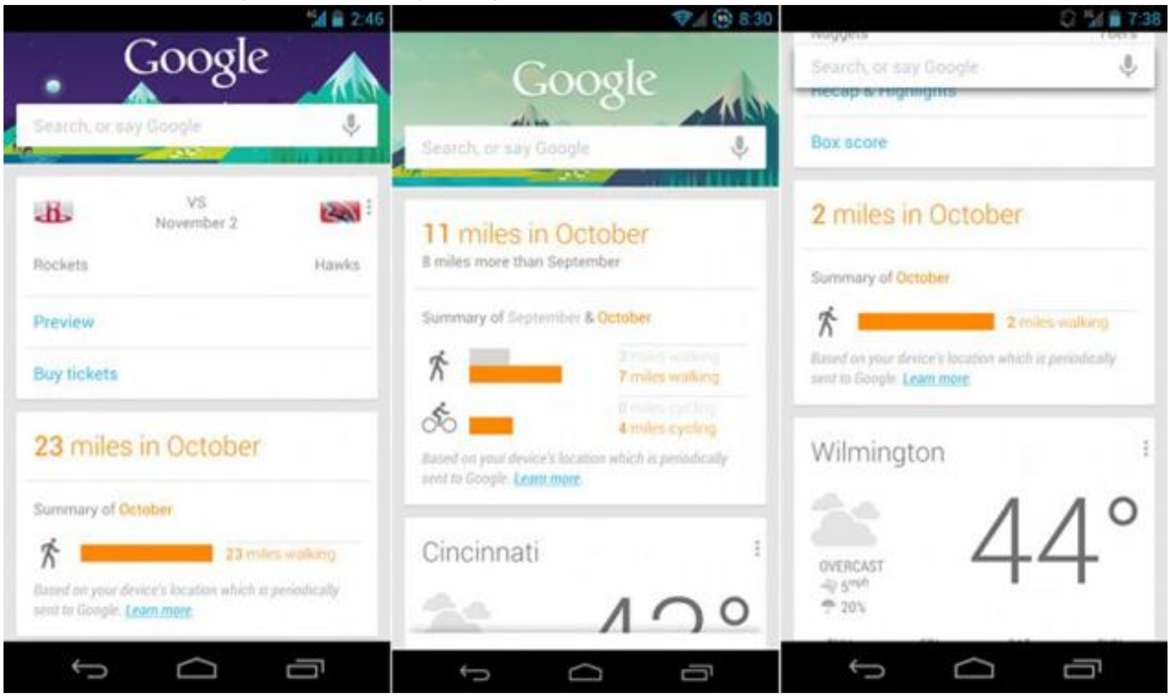


Fig. 5 Timeline card style screen example

- Search: shows users or events when you introduce a name in the search section.

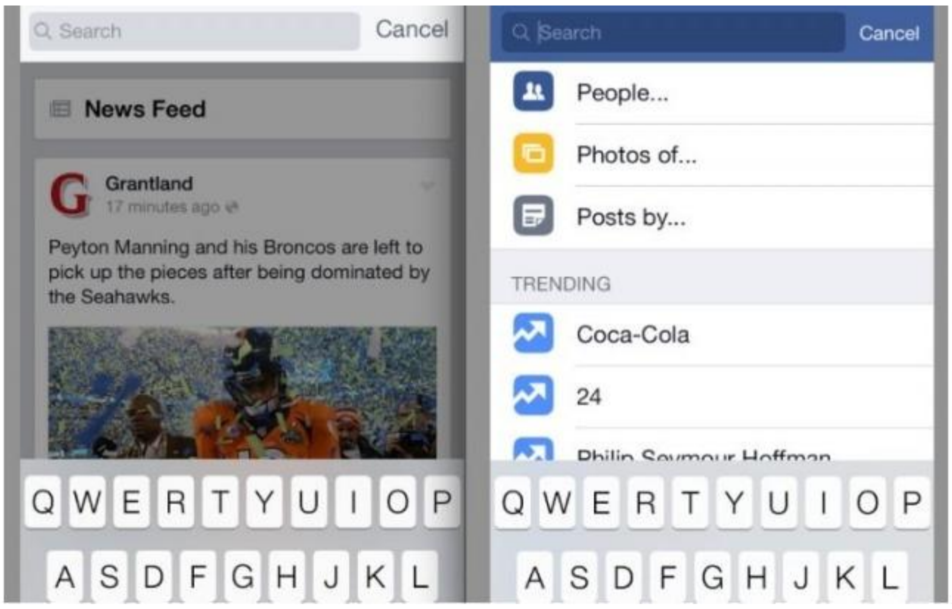
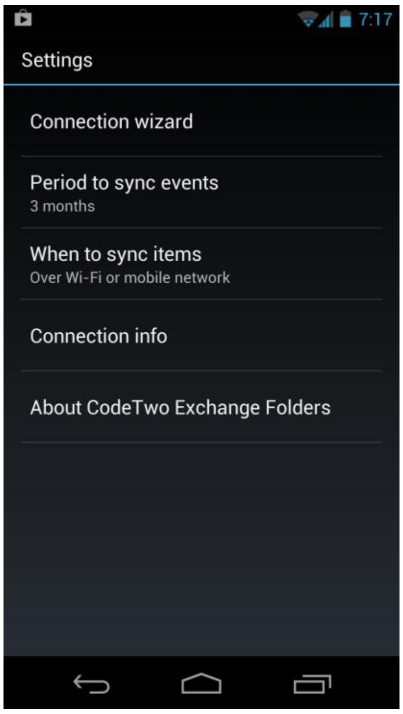

Search
Cancel
News Feed
Grantland
17 minutes ago
Peyton Manning and his Broncos are left to pick up the pieces after being dominated by the Seahawks.
People...
Photos of...
Posts by...
TRENDING
Coca-Cola
24

Fig. 6 Search screen example

- Settings: shows the common screen where a user can view and change their settings.



Settings

Connection wizard

Period to sync events
3 months

When to sync items
Over Wi-Fi or mobile network

Connection info

About CodeTwo Exchange Folders

Fig. 7 Settings screen example

- Profile: shows the common screen where a user can view and edit their profile.

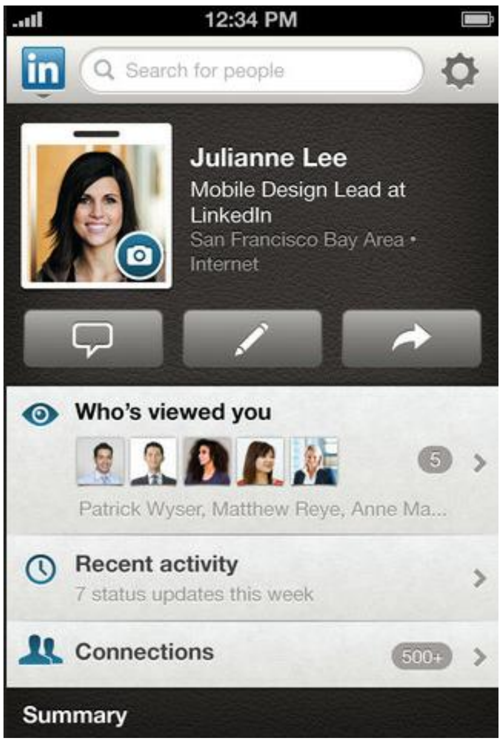

12:34 PM
in
Search for people
Julianne Lee
Mobile Design Lead at LinkedIn
San Francisco Bay Area • Internet
Who's viewed you
5
Patrick Wyser, Matthew Reye, Anne Ma...
Recent activity
7 status updates this week
Connections
500+
Summary

Fig. 8 Profile screen example

- Create match: shows a screen where a user can add information about a match and create it.

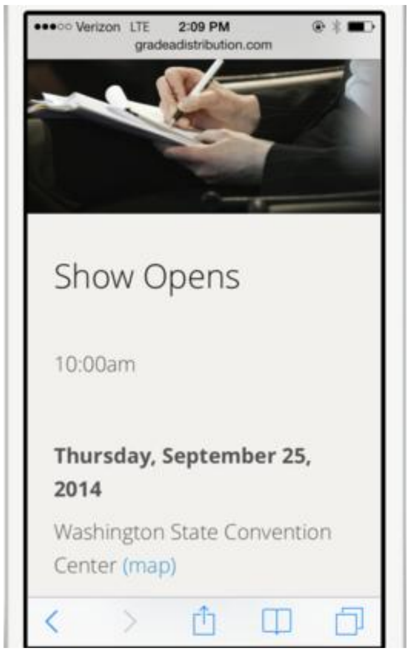

●●●○○ Verizon LTE 2:09 PM
gradeadistribution.com
Show Opens
10:00am
Thursday, September 25, 2014
Washington State Convention Center (map)

Fig. 9 Match screen example

- Edit match: shows a screen where a user can edit information about a match and update it. Similar to create match.
- My matches: shows a screen where a user can see in which matches is enlisted. Similar to the timeline.
- My friends: shows a screen where all the user's friends are listed.

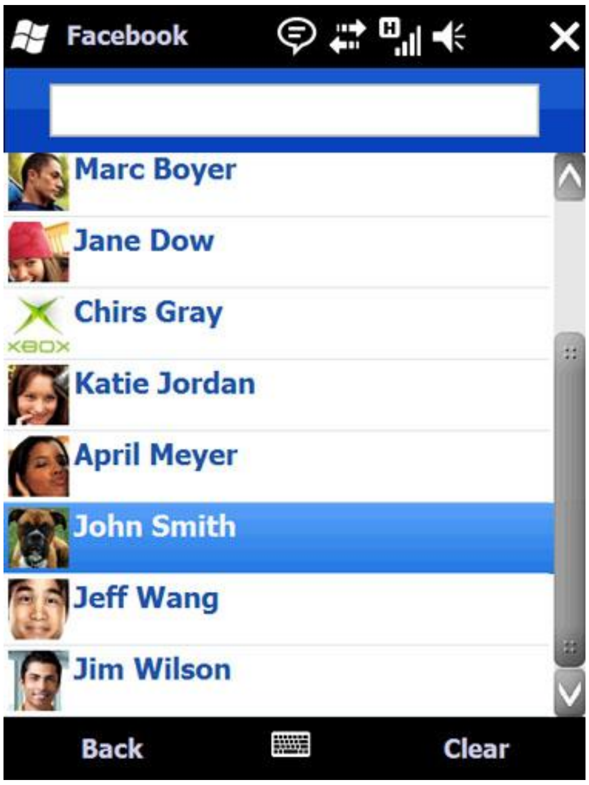


Fig. 10 Friends list screen example

- Connections: shows the common screen where a user can accept or decline an invitation to connect with another user.

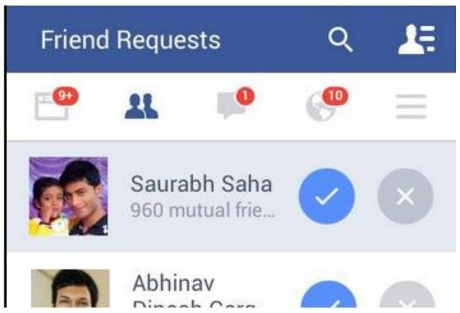
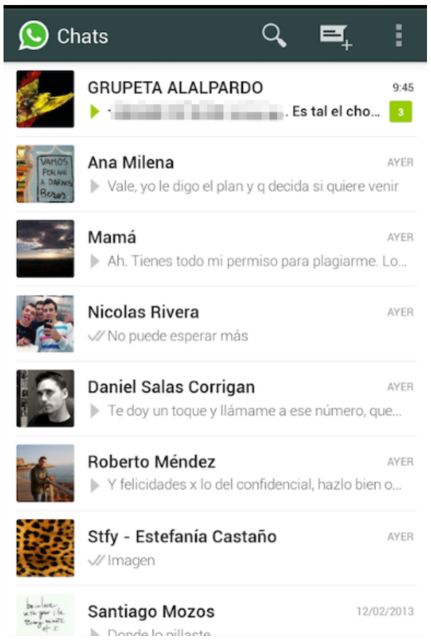


Fig. 11 Connections screen example

- Messages: shows the common screen where a user can start or access to a chat with other users.


Chats

GRUPETA ALALPARDO 9:45
· [redacted]. Es tal el cho... 3

Ana Milena AYER
Vale, yo le digo el plan y q decida si quiere venir

Mamá AYER
Ah. Tienes todo mi permiso para plagiarme. Lo...

Nicolas Rivera AYER
No puede esperar más

Daniel Salas Corrigan AYER
Te doy un toque y llámame a ese número, que...

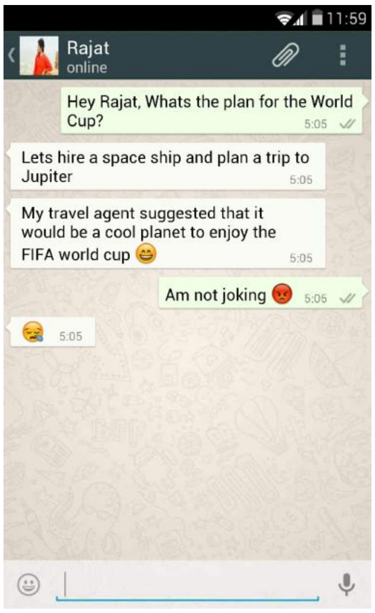
Roberto Méndez AYER
Y felicidades x lo del confidencial, hazlo bien o...

Stfy - Estefanía Castaño AYER
Imagen

Santiago Mozos 12/02/2013
Donde lo pillaste

Fig. 12 Messages screen example

- Chat: shows the common screen where a user can chat with other users


11:59

Rajat
online

Hey Rajat, Whats the plan for the World Cup? 5:05

Lets hire a space ship and plan a trip to Jupiter 5:05

My travel agent suggested that it would be a cool planet to enjoy the FIFA world cup 😄 5:05

Am not joking 😡 5:05

😪 5:05

Fig. 13 Chat screen example

.

More information about the screens design is available in the document Other Social Network Analysis (**Appendices 2.**).

The **notifications** will be the standard Android's notification that appear in the notification bar and you can interact with them in order to get to the new information.

Actions that must trigger notifications:

- Connection request sent.
- Match cancelled.

- New chat received.
- An specific event is created.

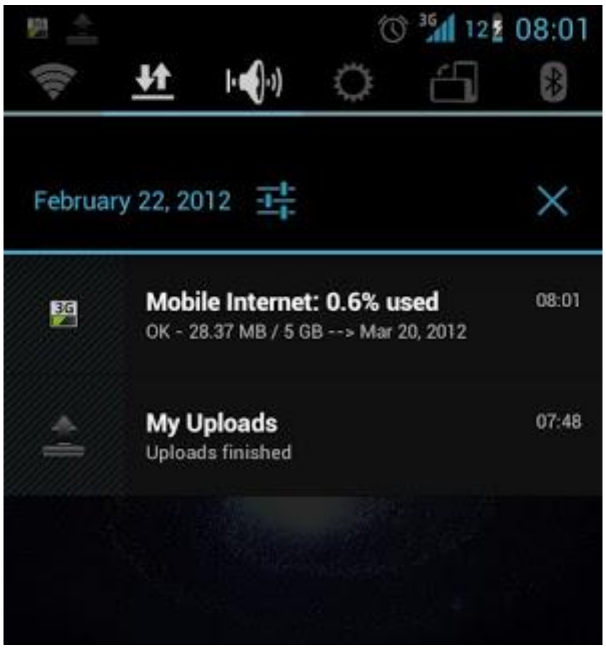

08:01
February 22, 2012
Mobile Internet: 0.6% used
OK - 28.37 MB / 5 GB --> Mar 20, 2012
08:01
My Uploads
Uploads finished
07:48

Fig. 14 Notification example

3.2. Functional Methods

As described in the document UML design (**Appendices 1.**), so as to develop an efficient application the first step must be define the main classes, their relationship and their methods.

During the first analysis, the main classes identified are:

User: the main class of the app. Each physical person or company has an associated user once they login. Due to simplicity and to avoid redundancy, I will define the Player's attributes and method (which are very similar to the company's).

- Main attributes: user ID, username, password, email, matches array, friends array, is public and allow public. All the attributes uses are pretty clear except is public and allow public, which mean if the user has a public profile (one who other users can freely see it without being friends) and if the user wants to see public matches (matches that can be seen by all the users and not only the owner's friends).
- Main methods: getters and setters for its attributes. Also methods in order to do the basic actions of the app: join a match, abandon a match, create a match, edit a match, send connection, accept connection, get connections and get matches. All the methods are pretty clear except the ones referred to the connections, which in fact are used to list, get and ask other users to be friends.

Match: the most important user's attribute and also an important object in order to know how the application works.

- Main attributes: match ID, match name, match owner description, location, start time, end time, sport, privacy, max number of players and number of players. The attribute privacy is used for indicate that an event is public or only can be seen by the match owner's friends (strongly relationated with the User's Class attribute allow public).

- Main methods: getters and setters for its attributes.

Database: the class developed in order to connect the database with the application and extract, update and delete the information.

- Main attributes: these attributes are SQL queries that are executed in order to get, update or delete a table element (which can be the whole table, a row or only a table cell). The major part of these are constants because they are going to be used a lot of times due to their usefulness.
- Main methods: these methods are operations that execute the SQL queries defined as class attributes.

- **Exceptions**: this is the class that its main function is detect and react to exceptions.

- main attributes: android native objects needed to show the different kinds of alerts used by the application.
- main methods: android native methods needed to show the different kinds of alerts user by the application.

- **Notifications**:

- main attributes: android native objects needed to show the different kinds of notifications used by the application.
- main methods: android native methods needed to show the different kinds of notifications used by the application.

More information and the whole process of identifying classes, relations, attributes and methods is better explained in the document UML design (**Appendices 1.**).

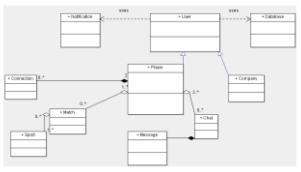


Fig. 15 Basic UML diagram

This is the basic UML that I designed before starting to implement the app.

3.3. Database

In order to test the basic operations used in the application, the first approximation will be an SQLite database. This database works locally, so the database is stored in the device and is not in a web service.

The interaction with the database is very important, so the design and the decision of which tables are going to be implemented, how many columns are going to be implemented and which fields will become a column is crucial so as to minimize the number of access to the database.

After a long reflection process about the app's behaviour , I decided to build 5 tables:

Users

id_user	username	password	email
24729834	example1	example1	example1@gmail.com
772773	example2	example2	example2@gmail.com

Table 1 Users database

This is the users' table. In this table, the users store their basic data used for login, such as the username, password and email but also stores their used ID. This ID is a random parameter that would be needed in order to access to the rest of the data related to the user. The access to this table must only be used to login and to obtain the id_user to access to the other databases.

Sessions

id_user	id_session
24729834	-7892345
-772773	7883256

Table 2 Sessions database

This is the sessions' table. In this table, the users obtain a random parameter connected to his user ID each time they login. This parameter is called session ID. To obtain their user ID and access to other tables or main information such as a password, it must be used the session ID.

UsersConfig

id_user	is_public	allow_public
24729834	0	0
-772773	1	1

Table 3 Users configuration database

This is the users' configuration table. In this table, with the user ID, the users can see his own configuration or other users configurations. The parameters is_public and allow_public mean that one user can be public (even the users who are not friends with this user can see their profile) and/or be notified from public events respectively. Both parameters could only have 2 values that mean allow or not allow.

UserConnections

id_user1	id_user1	status	id_connection
24729834	-772773	3	-256720

Table 4 Users connections database

This is the users' connection table. The users will contain an id_connection array with which can access to the connection status. The connection status can have 3 values: friends, user1 sent a request and user2 sent a request. Depending on the status (must be friends), other users could access to their friends id_users so as to access to their friends non-private data (for example their email, username or configuration but never to their passwords).

Matches

id_match	990990	991991
id_match_owner	24729834	-772773
name	match_example_1	match_example_2
location	location_example_1	location_example_2
description	description_example_1	description_example_2
privacy	0	1
sport	Football	Basketball
max_number of players	10	12
num_players	6	7
start_time	2015-05-13 18:00:00	2015-05-15 19:00:00
end_time	2015-05-13 20:00:00	2015-05-15 21:00:00

Table 5 Matches database

This is the matches' table. In this table, with the match ID, the users can access to all the data related to a match. In order to delete or update the match, that user must be the match's owner, so their user ID must be the same as the value of the column id_match_owner. The table is shown horizontal and not vertical due to its big number of fields and the not correct visualization in word.

MatchUser

id_match	id_user
990990	24729834
991991	-772773

Table 6 Match-user database

This is the table that shows the relation between matches and users. When one user is going to play a match, a new row will appear in the table to show the connection between the id_match and the id_user. This table is used to show which users are enlisted in a match and also in which matches is enlisted a user.

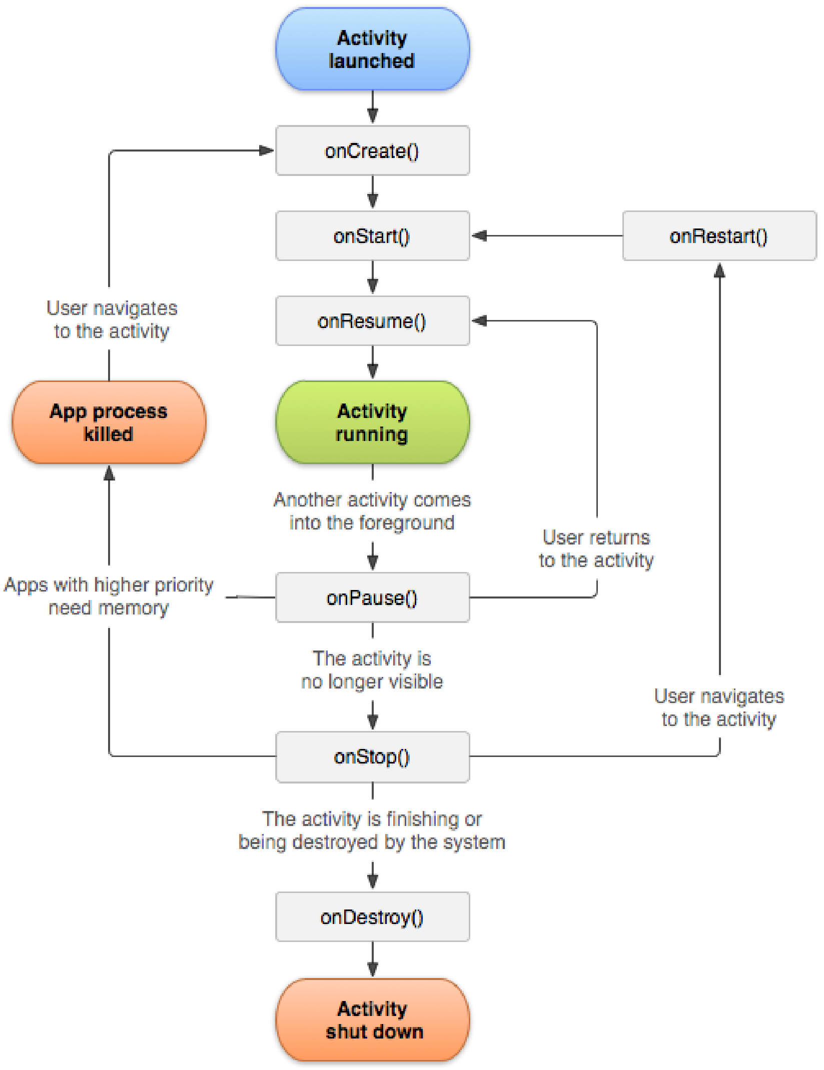
4. Implementation

The main details of the implementation of the project are in the folder WhereIsTheMatch where all the project's code is. In this section I will only comment curiosities or developments that I consider interesting.

For example, instead of starting the app directly from the login (which is the first screen), I decided to build a test screen with lots of buttons where I can decide just by pressing one button which screen is going to be shown. This was very useful when I was testing only one screen interface and its functionalities.

First of all, I want to make a brief comment about the **activities** and their **lifecycle**. An activity is an android class associated to a screen. This means that you must have an activity for each screen you want to include in your app. Because of the fact that I developed the project using Android Studio, this action was automatically made by the IDE (when you created an activity, automatically 2 files are generated, one associated with the functionalities - back-end and other with the interface - front-end).

An activity lifecycle basically is the activities' behaviour when they are running (foreground state) and receive focus or when another activity or app is called and the activity is moved into the background (where the activity is no longer visible, but the instance and its state remains intact). You can configure your activity in order to follow an specific behaviour (for example, in a video streaming app, you might pause the video and terminate the network connection when the user switches to another app. When the user returns, you can reconnect to the network and allow the user to resume the video from the same spot).



Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

Fig. 16 Activity lifecycle [5]

4.1. Interface

One thing that I would like to explain about my front-end development is the styles file. In this file I have defined my own styles that can be applied to a component or an XML file (typically an activity file).

```
<style name="MySwitchButton"
parent="@style/Widget.AppCompat.Light.ActionBar.Solid">
    <item name="colorAccent">@color/dark_purplish</item>
    <item name="colorControlHighlight">@color/purplish</item>
</style>

<style name="NoActionBarTheme" parent="Theme.AppCompat.Light">
    <item name="android:windowActionBar">false</item>
    <item name="android:windowNoTitle">true</item>
</style>
```

Once the styles were defined, I could use them as theme for activities (screens) or components (such as buttons, spinners...) and maintain easily the same style.

4.2. Functional Methods

One of the basic parts of the back-end application was the use of the Intent class to change from one activity to another.

```
public void showDataBaseTest() {
    Button button1 = (Button) findViewById(R.id.data_base_test);
    button1.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            Intent dataBaseTest;
            dataBaseTest = new Intent(getApplicationContext(),
AndroidDatabaseManager.class);
            startActivity(dataBaseTest);
        }
    });
}
```

Implementing an onClickListener method in a button (a method that triggers when the button is pressed), an Intent is created to change the activity. It is a common method that is applied in almost every activity and encapsulated in its own method. I tried to maintain the names of these kind of methods, so I usually used "showSomething()".

4.3. Database

When I was implementing the database, I tried to build a system as safe as possible and I developed an access method that makes the action of stealing information more difficult than normally.

```
private static final String ID_SESSION = "id_session";
private static final String ACTIVE_USERS_TABLE_NAME = "Sessions";

public int getUserIdBySessionId(int sessionID){
    SQLiteDatabase db = getReadableDatabase();
    String selectQuery = "SELECT  * FROM " + ACTIVE_USERS_TABLE_NAME + "
WHERE "+ ID_SESSION + " = " + sessionID;
    Cursor c = db.rawQuery(selectQuery, null);
    if(c!=null){
        c.moveToFirst();
        return c.getInt(c.getColumnIndex(ID_USER));
    }
    return -1;
}
```

All the methods that need the user ID in order to do something (that are almost all), first use this method to obtain it instead of using for example an user class method get. This translates in better security, because if the session ID is stolen, it would only works until the next user login.

5. Difficulties during the project

5.1. Design problems

Thanks to the document Other Social Networks Analysis(**Appendices 2.**), the idea of the design was very well defined. The problem was that with the integration of Android v21, which actually is the one that my mobile phone (One plus one) implements. This is a very recent version, only used in devices with at least Android 5.0 Lollipop as mine. The documentation about this version is very limited and there are a few basic features that are different such as implement an action bar or define the colours of basic elements as an spinner or a toggle button. Because of this, I spent several days reading API (Application programming interface)'s in order to discover which attribute was the desired and which method changed that attribute. I also read lots of documentations about how to remove the action bar or add buttons to it using this new version.

5.2. Development problems

Due to the document UML design (**Appendices 1.**), the classes to be created and the relations between them were pretty clear.

My first problem developing appeared when I tried to test the database in my mobile phone. It was very complicated to see if the database's tables were created because of my non-root (administration) permission in my device. Without being administrator, you cannot see the database file so I had to implement an API to be able to test the database behaviour and functionalities.

Later I discovered a problem, that some of the classes will be useless in future implementations, where the database will be in a web service. When the final database is developed and integrated, there is the need to update matches and friends lists each time any change is done to the database or even when the timeline is shown. So the solution is implement the classes Match and User as DAO(Data access object)'s. A DAO would provide synchronization between our web service database and the real objects that the user is going to use in order to execute their methods. I think a better approximation would be the update and the store of only the ID's of the objects that are changed in the database. But in order to implement this solution, which is less demanding in the device side, the server side must replace the old row by one copy with a different ID.

5.3. External problems

During the project I had several problems that had nothing to do with it but that have affected the project.

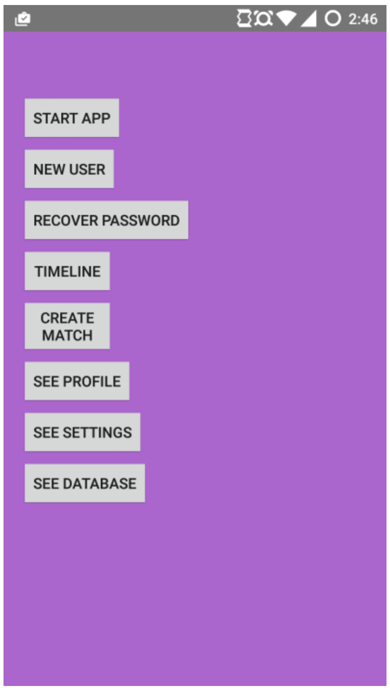
First of all, I changed my mind about which project I was going to do after a month working for an university ex-student. Due to the fact of several changes in the Gantt diagram that usually ended in useless work, I decided to start my own project and split up with this team. This is translated into a waste of time, but not completely because in that phase of the project I researched a lot of general information so as to start developing apps in android and I started to prepare my work environment with all the necessary tools in order to start the project.

Then I got the chance to start working in a company, Everis. I was looking for practices before I finished the degree, so I took the opportunity. My idea was to be able to do the project there, but I entered in a department that dedicates to SAP. Due to the new job and several subjects that I was studying at university, I decided to postponed the project and to present it on the next call.

Later, when I was playing a football league match, I had bad lucky and I broke one bone. I was two months injured with my leg putted on a cast. I decided to continue going to work even with the cast, so each day I ended exhausted and in this period my contribution to the project was less than I would have liked to contribute.

These are the main reasons that the Gantts diagrams presented in the project documents are so different and some deliverables had to be postponed.

6. Results and test



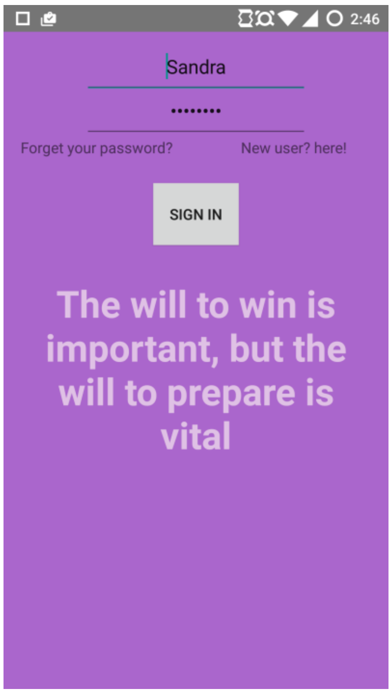
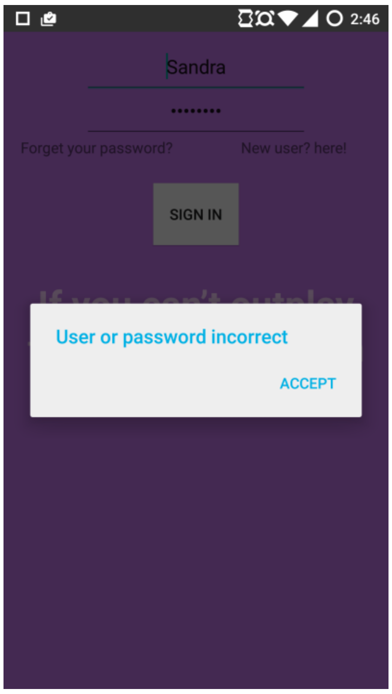

Sandra
Forget your password?
New user? here!
SIGN IN
The will to win is important, but the will to prepare is vital

Fig. 17 My test screen and Fig. 18 My login screen



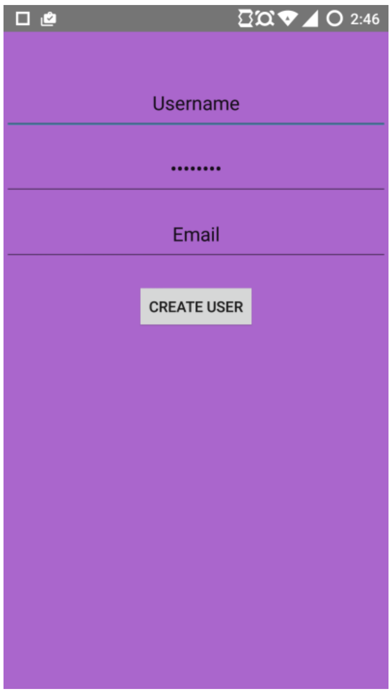
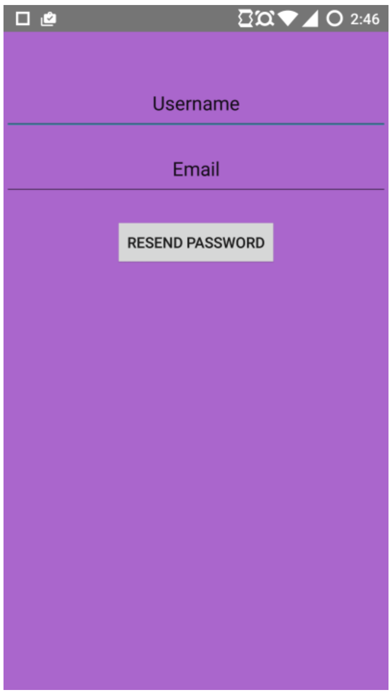
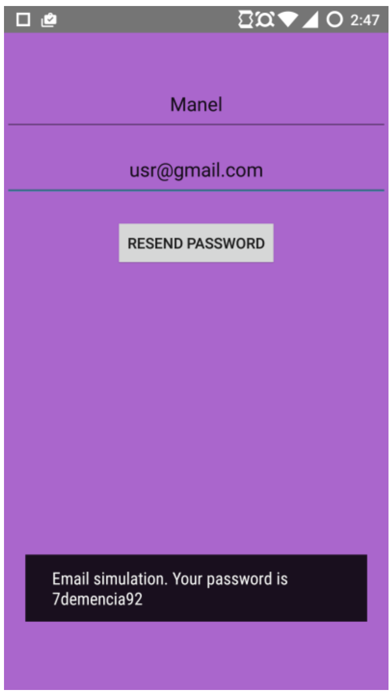


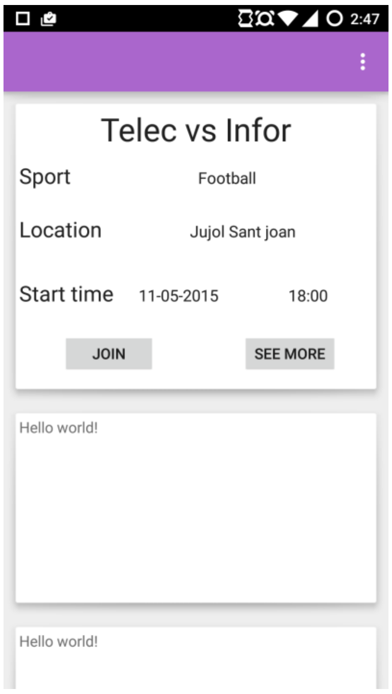
Fig. 19 My alert notification and Fig. 20 My create user screen





Email simulation. Your password is
7demencia92

Fig. 21 My recovery password screen and Fig. 22 My toast notification

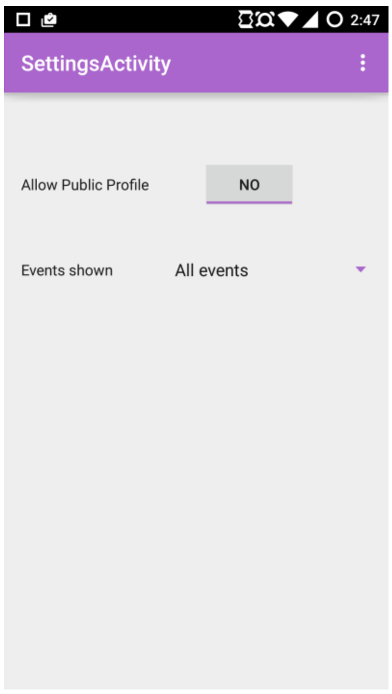


Telec vs Infor

Sport Football

Location Jujol Sant joan

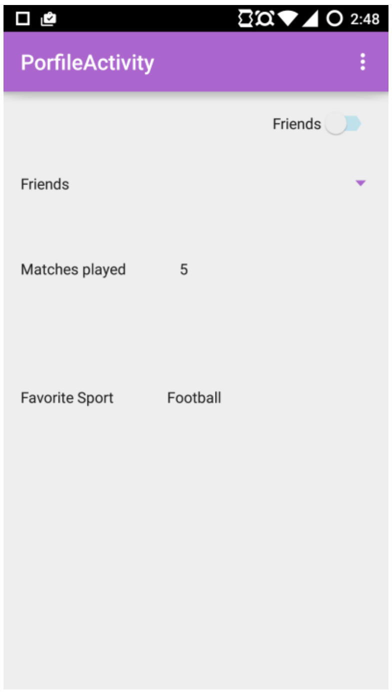
Start time 11-05-2015 18:00



Allow Public Profile NO

Events shown All events

Fig. 23 My timeline screen and Fig. 24 My settings screen


PorfileActivity

Friends

Friends

Matches played 5

Favorite Sport Football

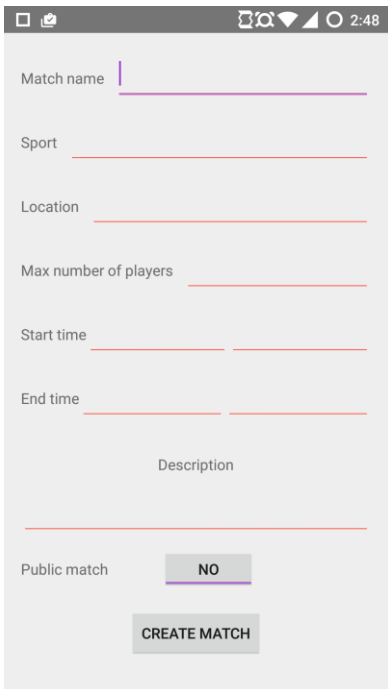
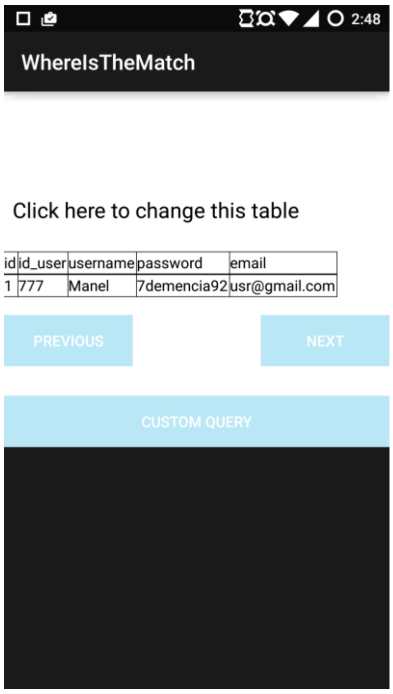


Fig. 25 My profile screen and Fig. 26 My create match screen


WhereIsTheMatch

Click here to change this table

id	id_user	username	password	email
1	777	Manel	7demencia92	usr@gmail.com

Fig. 27 Database screen

I tested the system in a single scenario with 2 users. The interaction in order to login work perfectly, being able to correctly recover a password, login and creating a new user. In the current version, all users are friends so they share matches and they can join and see them, but not delete or update them. The application has been tested only in my device but due to the SQLite database used, it can be easily used in another sending the database file and storing it in the current data folder. The problem is that in order to do that, you must have root (administrator) permissions in your device, for both storing the database and sending it to others. Other solutions has been thought for adapting this model into an multi-user with accesses via internet to the database.

These are the most common app executions that would the users follow:

- First time one installs the app. The users will take a look to the main app design, its features and configure their profiles.

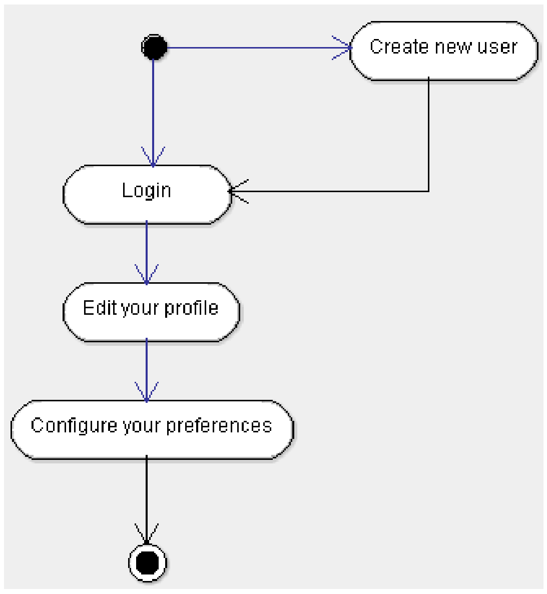


Fig. 28 Activity diagram example 1

- Both examples 2 and 3 are a common user's behaviour of an user who has used the app more than once. The behaviours are a bit different due to the intentions: the first diagram represents a user who is planning to play a completely public match and the second diagram represents a user who is planning to play a public with some friends (this is the reason why they communicate via chat before creating the match) It must be taken into account that some points are still in development and will be implemented in the future.

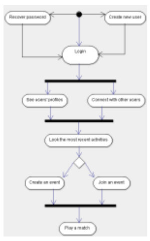


Fig. 29 Activity diagram example 2

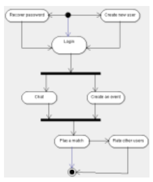


Fig. 30 Activity diagram example 3

7. Further design and development

So as to continue the development of the project, there are several things that can be done. I have arrived to a point that the project starts to have his own form and style but there is plenty of work to do.

- **Interface**: the primary colours of the app are defined as well as the screen styles. It would be easy now to identify or associated a new screen / component that follow the same style. More components style should be defined in the styles file in order to adapt each android element to the app's style, following the example of the toggle button or the spinner. Furthermore more screen formats should be styled following the app patterns just in case they are needed in the future, for example in other devices/platforms such as the web, tablets or even wearables. I also think a good idea would be start working on the design of new personalized icons to use in the app and to create a brand, like a developer signature.

- **Application's behaviour**: this is probably the most incomplete part of the project. There are basic features that should be implemented as soon as possible, such as delete a Match, send invitations in order to be connected, a chat service or a notification system. It would also be important to think ahead and start designing special methods for sports companies and not only common users.

- **Database**: the database implementation is one of the best things of the project. It is well organized and implements parameters and access methods that make difficult to trick it and obtain private information. However, its current state is not useful enough to build a social network. Thanks to the android implementation and integration with MySQL databases, it would be easy to port this system to a server connected to the Internet. That should be the next step. After that, I would supervise the methods that access to the database and apply the use of monitors theory and DAO implementation model so as not to have problems with the access to more than one user into the database and with useless objects. Finally, some kind of codification should be applied to the passwords and ID's to make almost impossible the information stealing from the database.

8. Commercialization

8.1. Project cost

Thanks to the fact that the major constituent of the project is software, the cost is not going to be so high as in other projects. If we take into account that at least it will be needed 5 times the amount of work that I have done to finish the project, and the necessary devices in order to build the app, we could estimate the cost in:

- We can estimate the total time in 18 weeks (4 months).

- We can estimate the total time work per week as 20 hours.

- We can estimate the salary per hour as 8€ (common internship salary) for 1 person.

- Both a mobile device and a computer are needed per worker. We can estimate this cost as 800€ per person.

The estimated total cost is 18400€.

8.2. Introduction to the market

Making money with a social network is not difficult nowadays if it has enough users. The typical way is to add advertisements in screen places that not disturb the actions of the users (for example, during the login or at the bottom of the timeline) and win money with their visualization or their clicks in the advertisements.

This should be the main model of business but I also proposed another. I think a good way to make money with this app is to have two types of users: normal and premium. The premium must pay a low tax per month (for example 1 €) but in exchange they will get the notifications a few minutes before the normal users get theirs. For example, if one user only needs 3 more people to play a match, the premium users will get the notification before the other users and this will give them priority. Simple and fair.

Another option to make money is to store the information about where the matches are played and sold it to companies that maybe could use it. Interested companies could vary a lot: for example from an sport company which wants to know where new possible customers are to a local bar which is interested in when people play near the place to offer sales those days.

The first step should be introducing this business model in Google Play and if the model works well, develop the app for iOS. Perhaps in a few a years and some negotiations, the app could suggest places or accessories sponsored by sports companies in exchange of paying a tax per month to the developer team.

9. Conclusions

During the process of developing the project, I realized my idea was too ambitious. Rome was not built in a day, and FB or TWTR neither, and of course FB and TWTR were not developed by just one man. Developing an app like this in less than half a year and by only one student who barely know how to properly programming in Java is a very hard task. I am happy to have had the chance to start this project but I am also a bit disappointed with the final result.

On one hand, I have discovered that I really love doing this; all of this. Thinking how the design must be in order to succeed, investigating other ideas similar to mine to complement it, implementing and testing the interface and the functionalities, creating an efficient database system and all the concepts related to build an app. I would enjoy continuing to doing this kind of projects on the future.

On the other hand, I think that I overestimated myself. I must have divided the project in an specific area from the start and finished it: back-end development, front-end development or database module. Trying to achieve all the areas has resulted in a bad final product; it has a part of everything but at the same time it is nothing yet. I would have liked to have more time to finish it or to have a team which I could work with and later regroup our work as a brilliant final app.

I encourage other students into this kind of projects but they must learn from my mistakes. The best way to build something completely functional is to make a team with different and complementary profiles such as graphic designer, two or three engineer and an economist.

Bibliography:

Most of the reference are only used to help during the sofware development of the project.

[1] Android developers design [Internet] [2015 May] Available from: http://developer.android.com/design/index.html

[2] Códigos de colores HTML. [Internet] [2015 May] Available from: http://html-color-codes.info/codigos-de-colores-hexadecimales/

[3] Android developers training. [Internet] [2015 May] Available from: http://developer.android.com/training/index.html

[4] Android developers guide. [Internet] [2015 May] Available from: http://developer.android.com/guide/index.html

[5] Activities. [Internet] [2015 May] Available from: http://developer.android.com/guide/components/activities.html

[6] Dialogs. [Internet] [2015 May] Available from: http://developer.android.com/design/building-blocks/dialogs.html

[7] UML. [Internet] [2015 May] Available from: http://login.osirislms.com/offline/uml/

[8] UML. [Internet] [2015 May] Available from: https://docs.kde.org/stable4/es/kdesdk/umbrello/uml-elements.html

[9] SQL support. [Internet] [2015 May] Available from: http://www.w3schools.com/sql/sql_where.asp

[10] SQLite database Android. [Internet] [2015 May] http://www.androidhive.info/2013/09/android-sqlite-database-with-multiple-tables/

[11] Correctly managing your SQLlite Database. [Internet] [2015 May] Available from: http://www.androiddesignpatterns.com/2012/05/correctly-managing-your-sqlite-database.html

[12] Android and general support. [Internet] [2015 May] Available from: http://stackoverflow.com/

Appendices:

An explanation about the files contained in the .zip file:

1. UML Design [PDF]: a file where is described the whole process of designing and analysing the classes, attributes and methods implemented in the application.
2. Other Social Network Analysis [PDF]: a file where is described the whole process of designing functionalities and screen interfaces. Also a comparison between other social networks.
3. WhereIsTheMatch [Folder]: the whole project code.

Glossary

IDE - Integrated Development Environment or Interacting Development Environment

FB - FaceBook

TWTR - TWiTteR

OS - Operation System

JVM - Java Virtual Machine

XML - eXtensible Markup Language

HTLM - HyperText Markup Language

CSS - Cascading Style Sheets

UML - Unified Modeling Language

SDK - Software Development Kit

NDK - Native Development Kit

ADK - Accessory Development Kit

CPU - Central Processing Unit

RAM - Random-Access Memory

ID - IDentifier

SQL - Structured Query Language

iOS - Iphone Operating System

API - Application Programming Interface

DAO - Data Access Object

G+ - Google Plus