Universitat de les Illes Balears

Escola Politècnica Superior

Memòria del Treball de Fi de Grau

Development of an Android APP for promoting the communication and exchange of information among tourists

Feng Ma

Grau d'Enginyeria Informàtica

Any acadèmic 2017-18

DNI de l'alumne: X9134693J

Treball tutelat per Isaac Lera Castro
Departament de Matemàtiques i Informàtica

	Autor		Tutor	
S'autoritza la Universitat a incloure aquest treball en el Repositori Institucional per a la seva consulta en accés obert i difusió en línia, amb finalitats exclusivament acadèmiques i d'investigació	Sí	No	Sí	No
	X		X	

Paraules clau del treball:
Android, Mobile application, tourism service, communication platform

CONTENTS

Contents i

Acronyms iii

Abstract v

1 Introduction 1

2 Market Research 3

3 Design of the application 7
- 3.1 Requirements . . . 7
  - 3.1.1 Functional Requirements . . . 7
  - 3.1.2 Non-Functional Requirements . . . 8
- 3.2 Use Cases . . . 9
- 3.3 Architecture . . . 17
- 3.4 Database . . . 19
- 3.5 User Interface . . . 24

4 Implementation 35
- 4.1 Authentication System . . . 36
- 4.2 Live Chat System . . . 37
- 4.3 Share Articles System . . . 37
- 4.4 Support of multi-languages . . . 39
- 4.5 Database for tags . . . 41
- 4.6 Levenshtein Algorithm . . . 42
- 4.7 Protection of data . . . 43

5 Timetable and Legalization 45
- 5.1 Timetable . . . 45
- 5.2 Legalization . . . 46

6 Conclusion 49

Bibliography 51

ACRONYMS

APP Application

OS Operating System

JSON JavaScript Object Notation

SQL Structured Query Language

URL Uniform Resource Locator

XML Extensible Markup Language

AES Advanced Encryption Standard

DES Data Encryption Standard

RSA Rivest Shamir Adleman

ABSTRACT

Nowadays, mobile applications have brought a variety of functions to smartphones, and make them more convenient and useful for users. This project aims to develop a mobile application which is related to the tourism. The main objective of the application is to promote and enhance the communication and exchange of information among tourists in the world. The application is addressed to tourists for sharing their travel experiences, feelings and suggestions, or search related information for preparing their future travel plans. The prototype of the application will be done in the end of this project, completing its major objectives. The work of release, improvements and maintenance of the application is not in the scope of this project.

Personally, I have been interested in mobile technologies and applications for a long time, its quick development and variety of functions really attract me. So, in my end-of-degree project, I decide to develop my first mobile application. Due to the lack of related experience, in the project, I will focus on the design and implementation of the application functions, the application's user interface will be simple. I expect to learn some knowledge about the mobile technologies and the process of developing an application from this project. I also want to get a outlook about the mobile applications' market. Furthermore, I hope the developed application could really be helpful and useful for the tourists meeting some of their demands.

CHAPTER 1

INTRODUCTION

This project is dedicated to developing a mobile application which aims to facilitate and promote the communication and exchange of information among tourists. In this application, they will be able to find and make relationships with other tourists, search and exchange information for preparing their future travel plans. Another main reason to carry out this project is my personal interest in mobile applications. I want to learn mobile technologies and understand the process of developing a mobile application from this project. This application runs in Android platform, since it has much more market share in the world which means more potential users the application can get.

In order to reach the goal of facilitating and promoting the communication and exchange of information among tourists, the application will provide the following services as its main features:

1. **Live Chat System**: a system which provides live chat service, which make users to be able to start a live chat with other users, including the chat between two individuals and among a group of users. This is the most straightforward way for users to communicate and exchange information with others.

2. **Share Articles System**: this is the key feature of the application. Users are able to write down their travel experience, feelings and suggestions in a form of articles and share with others. Each article is composed with title, content, release date and author's name. Authors are also able to add related map with markers to complement their articles' information. To promote the interaction between authors and readers, there is a set of actions that readers can do with the articles, such as writing comments, ratings and adding proper tags to the articles.

3. **Establishing relationships among tags**: this is a feature conducted by the developers of the application. Its purpose is to improve the function of searching articles with keywords by establishing relationships among tags. This feature will allow the application to find proper articles even if the keywords have not appeared in their titles and tags.

In addition to these three key features, the application also requires Authentication System for the management of accounts. This project will end with a prototype of the application containing the features above, the work about the application's release, improvement and maintenance is not in the scope of this project. To develop such an application, the project is divided into five phases.

The first phase consists of the research of the market and comparison with other similar applications. This research aims to analyze and study the viability of the application, get clear the situation of the target market and the advantages and disadvantages between this application and others, make sure this application is capable of surviving in the market and helping users.

The second phase is the design of the application. I have already presented the main features of this application, and this phase begins by analyzing all possible requirements and use cases of the application based on its main features. With the information resulted from the analysis, I can start to design the application's architecture, its database and user interface.

Once the design of the application has been finished, the implementation phase begins. This phase involves the development and construction of the application based on the design from previous phase. In this phase, all modules related to the application start building, including server's preparation and setup, building of user interfaces and adding functions to the application to realize its requirements. In this phase, the prototype of the application will come out and be ready for starting the next phase.

The fourth phase of the project consists of running and testing the application. The application will be tested by running on devices, and all bugs found in the tests will be fixed to ensure that the application can run perfectly without any errors and is ready for release.

The final phase is to make a conclusion about the whole process of the project, record all learnings and thoughts about this project, and plans about the future works with the application, such as possible improvements and maintenance.

In this document, there are six chapters in total. Chapter 2 records information about the research of the market, which corresponds to the first phase of the project. Chapter 3 describes all works and results from the design phase of the project. In chapter 4, I present the building of the application with some examples of codes. It also corresponds to the implementation phase. Chapter 5 explains the project's timetable and requirements for the legalization of the application. The final chapter corresponds to the final phase of the project with conclusions about the project, including the review of the achievement of the objectives and some of my feelings from this project.

CHAPTER 2

MARKET RESEARCH

The project starts with market research, this is an important task before officially starting the design and development of the application. Its objective is to study the viability of the application by comparing it with other applications in the market. This task begins with choosing some most popular applications with similar target markets and objectives, then analyze their main features and compare them with my application to understand the advantages and disadvantages.

Large tourism market has attracted many companies and programmers to develop related applications, but none of them is working for promoting and enhancing the communication among tourists. Major companies and programmers dedicate their applications to helping tourists to make travel plans and recording travel experience. For tourists, these application just like digital travel guides, they can easily get information about hotels, restaurants and tourist places in a city, which help them to find places for sleeping, eating and playing. To know what advantages and disadvantages will exist between my application and these applications, I have randomly chosen three of the most popular applications in the market and listed their main features and advantages against others. These three applications are Trip.com [1], Minube[2] and Guides by Lonely Planet[3], all of them are searched from Google Play using "travel guide" as the keywords, their ratings are more than 4 stars and have been downloaded at least one million times (data until 30/09/2017).

- Trip.com: in addition to providing information about hotels, restaurants and tourist places, it also generates suggestions for users based on their location, time and some other information. For example, generate a list of hotels and restaurants near the user. Moreover, users can read comments about hotels or restaurants from others users of the application. And the biggest advantages is that, the company Trip.com has cooperated with some hotels to offer discount service for users (Figure 2.1). In the application, users can find hotels with discounts on their price and links which lead them to book the room . In this way, more users

will be attracted to use this application and book hotels recommended by the application since they can pay less on hotels with a discount on price.

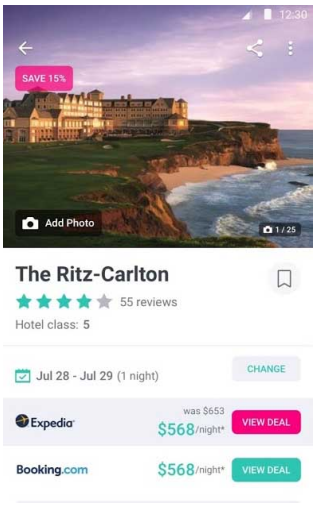

Figure 2.1: Discount service in Trip.com

- Minube: it has been chosen as one of the best 50 APPs for 3 years (2013-2015) in Google. Provision of the travel information is one of its main features, users can also read comments about hotels or restaurants from other users. It can also find tourist attractions near the user and displayed them on the map. The most unique feature of this application is the feature of albums (Figure 2.2). Users are able to create and manage albums, upload photos taken during the travel to these albums. There is also an option allows users to set geolocations for these albums to record the location where the photos were taken. It helps Minube to attract and retain users, because a cost will be generated for users when they stop the use of Minube, they have to abandon all photos that they have uploaded or they have to spend time to download all photos and save them in somewhere.

- Guides by Lonely Planet: an application developed by the company Lonely Planet. In this application, users are also able to find information about hotels, restaurants and tourist places and this information is available for downloading in order to use offline. Moreover, it has provided a set of tools to help and resolve some possible problems during the travel, such as a currency converter for calculating the price, an audio with some most frequent phrases in different languages (Figure 2.3). In the audio feature, there are up to 19 different languages, covers most of the popular languages in the world. With this service, it will be much easier for users to make some simple conversation with local inhabitants.

All of them have some additional features for attracting and retaining users, but their main feature is the same: provision of travel information. They provide information about hotels, restaurants and tourist places to tourists. Some of them also provide information about stores, bars or the history and culture of a city. But in my application,

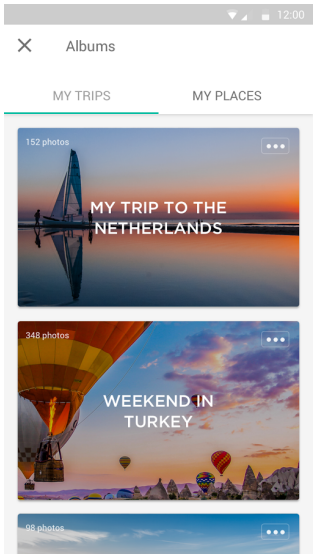

Figure 2.2: Albums service in Minube

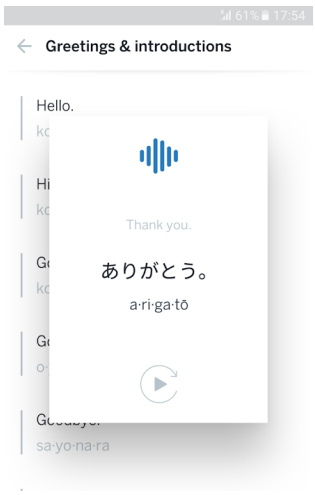

Figure 2.3: Audio feature in Guides by Lonely Planet

tourists are able to get different kind of information: the true experience, feelings and suggestions from other tourists, without any tricks or false information. They can choose to communicate with other tourists by using the *Live Chat System*, or search shared information in the *Share Articles System*. Despite providing such kind of information, there are some disadvantages as well. First of all, since all information is edited and published by the users of the application, it will not cover everything. For example, it is hard to find all hotels located in a city, users can only find the hotels recommended by others. Moreover, it is hard to find specific information like displaying hotels and restaurants near them. Considering this issue, my application does more

works in the function of searching articles, one of them is establishing relationships among tags and store them in the database, this could help the application to find proper articles even though the keywords do not appear in articles' title and tags.

I think the relationship between my application and other travel guide applications is to complement each other. Travel guide applications provide information captured from the Internet or edited by the experts and is pretty complete. And in my application, the information is about the experience, feelings and suggestions from the tourists. These are two kinds of information from two different perspectives, both of them are useful for making travel plans.

CHAPTER 3

DESIGN OF THE APPLICATION

This chapter presents some most important works in the design phase of the project. It includes the analysis of the application's requirements and use cases, and design of the application's architecture, database and user interfaces.

3.1 Requirements

To start designing the application, the first step is to make a list of all possible requirements of the application based on its features and objectives. According to their characteristics, these requirements are divided into two types: functional requirements and non-functional requirements.

3.1.1 Functional Requirements

A functional requirement defines a function of a system or its component. It can be calculations, technical details, data manipulation and processing and other functionality that define what a system is supposed to accomplish [4]. In this section, all functional requirements of the application will be listed and grouped according to their relationships with the application's features. Each requirement has an unique ID with a format like RE_XX, where XX represents the number of the requirements and a number of priority, from 1-5, 5 represents the highest level of priority.

Requirements for building Authentication System:

- **RE_01 Account Management.** Users are able to register new accounts and modify the accounts' data, such as the password, account's name, avatar, etc. *Priority*: 5.
- **RE_02 Switch Accounts.** Users are able to log out from current account and log in with another account. *Priority*: 5.

Requirements for building Live Chat System:

- **RE_03 Search Users.** Users are able to find other users and check their information. *Priority*: 5.
- **RE_04 Contact List Management.** Users are able to add other users to the contact list as friends or remove them from the list. *Priority*: 5.
- **RE_05 Chat Room Management.** Users are able to create new chat rooms, invite others users or remove members from the chat rooms. They can also modify room's data, like the name, icon, etc. *Priority*: 5.
- **RE_06 Live Chat Service.** Users are able to start a live chat with others. It could be an individual chat or group chat. *Priority*: 5.

Requirements for building Articles Share System:

- **RE_07 Articles Management.** Users are able to manage their articles, including create new articles, modify articles' content or delete articles. *Priority*: 5.
- **RE_08 Articles Search.** User are able to search articles by introducing keywords. *Priority*: 5.
- **RE_09 Share Articles to Chat Rooms.** Users are able to add their articles to a chat room's articles list or remove them from the list. *Priority*: 5.
- **RE_10 Add Related Maps.** Users are able to edit related maps for complementing the information of their articles. *Priority*: 5.
- **RE_11 Add Tags.** Users are able to add tags to all articles. These tags will make the articles to be found with keywords more easily. *Priority*: 5.
- **RE_12 Tags Management.** Users are able to manage tags of their articles, delete tags they think are improper to the content of the articles. *Priority*: 5.
- **RE_13 Articles Comments.** Users are able to check comments of an article written by other users or write their own comments. *Priority*: 5.
- **RE_14 Articles Rating.** Users are able to check the average rating stars of an article or make a new rating for it. *Priority*: 5.
- **RE_15 Search Articles on Maps.** Users are able to search articles on maps. Each article will be located in the corresponding place based on its content. *Priority*: 5.

3.1.2 Non-Functional Requirements

A non-functional requirement is a requirement that specifies criteria which is used to assess the operation of the application. It can be the requirements about system's usability, security, maintainability, etc [5]. Below is a list of non-functional requirements for the application, each requirement has an unique ID with a format like NRE_XX, where XX represents the number of the requirements and a number of priority, from 1-5, 5 represents the highest level of priority.

- **NRE_01 Server 24/7 online.** The server has to be online in every day, every hour. Otherwise, the services of the application will be affected. *Priority*: 5.
- **NRE_02 Multi-language support.** The application can adapt the language of the OS. In this project, the application supports Chinese, English and Spanish. *Priority*: 4.
- **NRE_03 Finding proper articles.** The application must be able to find all proper articles with the keywords introduced by users. *Priority*: 5.
- **NRE_04 Protect sensitive data.** The sensitive data need to be encrypted before being stored in the database in order to protect users' privacy. *Priority*: 5.

3.2 Use Cases

A use case is a list of actions or event steps, typically defining the required interactions between a role and a system, to achieve a specific goal [6]. It is required to design possible use cases of the application according to its features and requirements, and all use cases need to be specified in details with a tabular format and each table has the following fields (this an use case template found in the Internet[7]):

- Identifier: a unique id for every use case with a format like UC_XX, where XX corresponds to the number of use cases.
- Title: describes the goal of this use case.
- Description: describes the goal and context of this use case in details.
- Requirements: indicate which requirement of the application will be realized in this use case.
- Actor: it could be a person or a software/hardware system that interacts with the system to achieve the goal of this use case.
- Pre-condition: describes the state of the system before this use case.
- Post-condition: describes the state of the system after this use case.
- Scenarios: specify the actor's actions step by step during this use case.
- Extensions: describe all the other scenarios for this use case, for example, possible exceptions and errors produced in this use case.

Here I present some use cases with diagrams as examples(More use cases are saved in the document name "UseCases.pdf"). The first diagram (Figure 3.1) shows use cases related to accounts. When the users have logged in successfully, they can manage the account's information, including the profiles (name, password) and its articles.

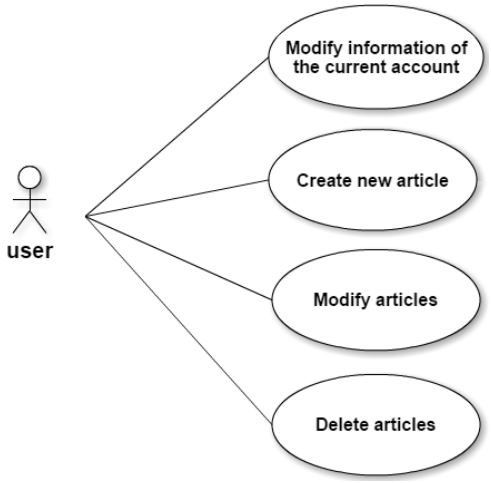

Figure 3.1: Use Case Diagram: Account's information management

Identifier	UC_03
Title	Modify information of current account
Description	Modify information of current account, including the password, the nickname and the icon.
Requirements	RE_01 Account Management
Actor	User
Pre-condition	1. The user has already logged in with an account. 2. The device has been connected to the server.
Post-condition	1. Information on the screen will be updated immediately. 2. New data will be transmitted to the server.
Scenarios	1. User clicks on the area where displays the account's icon, nickname/password. 2. The application displays all images in the device or a dialog asking for new nickname/password. 3. User chooses an image from the device or entering a new nickname/password. 4. User clicks on button to implement the changes.
Extensions	1. Password error: the password is too short, too long or contains same special characters. 2. Nickname error: the nickname is too short, too long or contains same special characters. 3. Fail to update information because of the problem of the Internet.

Identifier	UC_04
Title	Create new articles
Description	Create new articles by filling the title and content. A related map can also be added if it is necessary.
Requirements	RE_07 Articles Management
Actor	User
Pre-condition	1. The user has already logged in with an account. 2. The device has been connected to the Internet.
Post-condition	1. New article will be added into the user's article list. 2. The data of the new article will be transmitted to the server.
Scenarios	1. User clicks on the option of creating new articles. 2. User edits the new articles, entering the title and content. 3. User clicks on adding images option for adding images into the content part. 4. User clicks on buttons to save new articles.
Extensions	1. Fail to create a new article because of the problem of the Internet.

Identifier	UC_05
Title	Modify articles
Description	Modify articles' data, it could be the title, content or the related map. If the article does not have a related map, the user could add a new one for the article.
Requirements	RE_07 Articles Management
Actor	User
Pre-condition	1. The user has already logged in with an account. 2. The device has been connected to the server. 3. The user has already created some articles.
Post-condition	1. Changes will be updated and implemented immediately in the application. 2. New data will be transmitted to the server.
Scenarios	1. User clicks on one of his/her articles. 2. User modifies the title or the content. 3. User modifies the related map. 4. User clicks on save button to save and implement the changes.
Extensions	1. Fail to load information of the article because of the problem of the Internet. 2. Fail to load the related map of the article because of the problem of the Internet. 3. Fail to save changes because of the problem of the Internet.

Identifier	UC_06
Title	Delete articles
Description	The user deletes his articles.
Requirements	RE_07 Articles Management
Actor	User
Pre-condition	1. The user has already logged in with an account. 2. The device has been connected to the server. 3. The user has already created some articles.
Post-condition	1. The data of article will be deleted from the server. 2. The user's article list will be updated immediately.
Scenarios	1. User long clicks on one article. 2. The system displays a dialog asking the user to confirm this operation. 3. User clicks on button to delete the articles.
Extensions	1. Fail to send the request to the server because of the problem of the Internet.

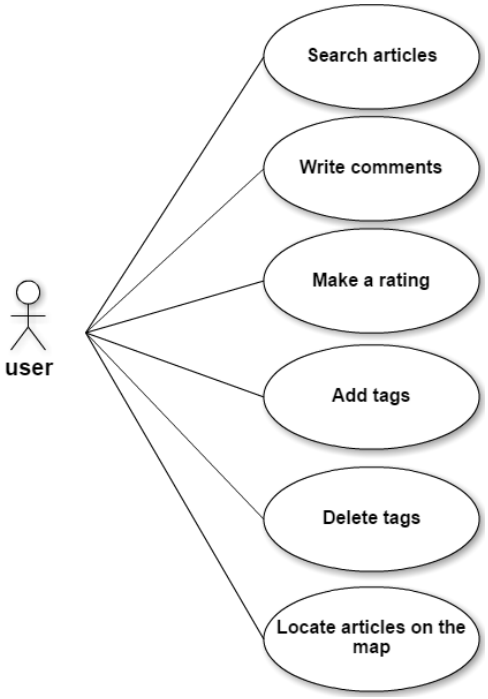

Figure 3.2: Operations about the articles

The second diagram (Figure 3.2) shows a list of use cases between users and articles, including searching articles, writing comments for the articles, ratings, adding and deleting tags.

Identifier	UC_20
Title	Search articles
Description	Search articles with keywords.
Requirements	RE_08 Articles Search
Actor	User
Pre-condition	1. The user has already logged in with an account. 2. The device has been connected to the Internet.
Post-condition	1. All articles related to the keywords will be displayed on the screen.
Scenarios	1. User enters the keywords 2. User clicks on the button to start searching articles.
Extensions	1. Fail to send the request because of the problem of the Internet.

Identifier	UC_21
Title	Write comments
Description	Write comments for articles.
Requirements	RE_13 Articles Comments
Actor	User
Pre-condition	1. The user has already logged in with an account. 2. The device has been connected to the server.
Post-condition	1. The new comment will be added to the comment list. 2. New data will be transmitted to the server.
Scenarios	1. User clicks on the comment option. 2. User writes new comments. 3. User clicks on button to save the comments.
Extensions	1. Fail to send the request because of the problem of the Internet.

Identifier	UC_22
Title	Make a rating
Description	Make a rating for an article from 1 star to 5 stars.
Requirements	RE_14 Articles Rating
Actor	User
Pre-condition	1. The user has already logged in with an account. 2. The device has been connected to the server.
Post-condition	1. The rating of this article will be re-calculated and display the new final rating. 2. User's rating will appear under the rating bar. 3. The button "Submit" will disappear. 4. New data will be transmitted to the server.
Scenarios	1. User clicks on the rating bar. 2. User clicks on a button to submit his/her rating.
Extensions	1. Fail to send the request because of the problem of the Internet.

Identifier	UC_23
Title	Add tags
Description	Add new tags for the articles.
Requirements	RE_11 Add Tags
Actor	User
Pre-condition	1. The user has already logged in with an account. 2. The device has been connected to the server.
Post-condition	1. New tags will appear immediately in the area where shows all tags of this article. 2. New data will be transmitted to the server.
Scenarios	1. User clicks on "add tags" option. 2. User enters new tags. 3. User clicks on button to save the new tags.
Extensions	1. Fail to send the request because of the problem of the Internet. 2. New tags contains some special characters.

Identifier	UC_24
Title	Delete tags
Description	Delete tags of an article.
Requirements	RE_12 Tags Management
Actor	User
Pre-condition	1. The user has already logged in with an account. 2. The device has been connected to the server.
Post-condition	1. The tag will disappear immediately from the screen and be deleted from the database. 2. New data will be transmitted to the server.
Scenarios	1. User long clicks on one of the tags. 2. The application displays a dialog asking the user to confirm the operation. 3. User clicks on a button to delete the tag.
Extensions	1. Fail to send the request because of the problem of the Internet. 2. Fail to delete tags because the user does not has the required permission: author of the article.

Identifier	UC_25
Title	Locate articles on the map
Description	Search articles on the map: every article will located on the map according to the city it describes in the article.
Requirements	RE_15 Search Articles on Maps
Actor	User
Pre-condition	1. The user has already logged in with an account. 2. The device has been connected to the server.
Post-condition	1. A new page will appear showing the information of the article which one has been selected by the user.
Scenarios	1. User clicks on the map search option. 2. User moves the focus,zooms out or zooms in the map for searching cities 3. User clicks on markers around the city, which represent articles. 4. User clicks on the title of the marker to check the information of its corresponding article.
Extensions	1. Fail to send the request because of the problem of the Internet.

3.3 Architecture

In the chapter 1, I have explained that the application has three main features: *Live Chat System* and *Share Articles System*, which offer two ways for users to communicate with others, and *establishment of relationships among tags* for improving the search articles function. In addition, an *Authentication System* is also required for the management of accounts.

To build these modules, I need a server for storing data and image files. Eventually, I choose to use the server and tools provided by Firebase[8]. Firebase is a company aims to provide a series of tools and services to help programmers to develop applications and websites efficiently. Compared to other servers I have found in the Internet, the management of Firebase's server is easier with a set of measures to guarantee its safety. Moreover, Firebase has provided documents about how to implement its services and tools in the applications in different languages. It allows me to do less works on the management of servers and concentrate on the design and building of user interfaces and features of the application. Among all services provided by Firebase, I use three services to build the modules of the application: Authentication Service, Realtime Database Service and Cloud Storage Service.

- Authentication Service: allows the application to register accounts with emails or accounts of other social medias, such as Facebook, Twitter. In addition, it also allows the application to modify or reset the password of an account[9]. Both the email and password will be stored safely in Firebase server and it will generate an unique ID for each registered account.

- Realtime Database Services: it provides a NoSQL database for storing data, and the data could be synchronized in realtime to all users' applications connected to the database[10]. Its synchronization feature is realized by setting a listener in the application to monitor the status of the database (it could monitor ths status of the whole database or a specific location in the database). When there are some changes happened to the database, such as storing new data or modification of existing data, the application would receive a signal from database, be aware that the data in the database has been changed.

- Cloud Storage Service: it provides a cloud storage where can upload and manage archives like photos, videos, etc[11].

The Figure 3.3 shows the relationships between the application's modules and Firebase services, what services are required for building each module of the application.

Firebase realtime database plays the key role in the building of *Live Chat System*. To implement live chat function, the application sets a listener to monitor user's messages list in the database, when a message has been sent, the application will create a message object storing this message's information, including its id, message content, sender and receiver's id. Moreover the message's id will also be added into sender and receiver's messages list in the database. In this way, if the user is the sender or receiver of the new message, his/her application will be triggered in order to read the messages list again and display them on the device, including the new message.

In *Share Articles System*, all articles are also stored in the realtime database. With its synchronization feature, the application could display the latest articles' information. The way to realize it is the same as the as the way of live chat function. A listener will be set to monitor the database where stores all shared articles. Considering that the realtime database does not support the storage of files like images, I need to use cloud storage service of Firebase for storing all images uploaded by users.

The way to establish relationships among tags is to create a database in the Firebase Realtime Database for storing tags with their related tags. When the application tries to find tags related to the keywords, what it has to do is access this database and search the data with the keywords. More details about the tag relationships database will be explained below.

The building of *Authentication System* is based on the authentication service provided by Firebase, the email and password of an account are stored in the authentication service server with functions allow the application to send requests to the server for registering new accounts, authenticating and changing the password. Moreover, I use realtime database and cloud storage for storing additional data such as usernames, avatar image, etc.

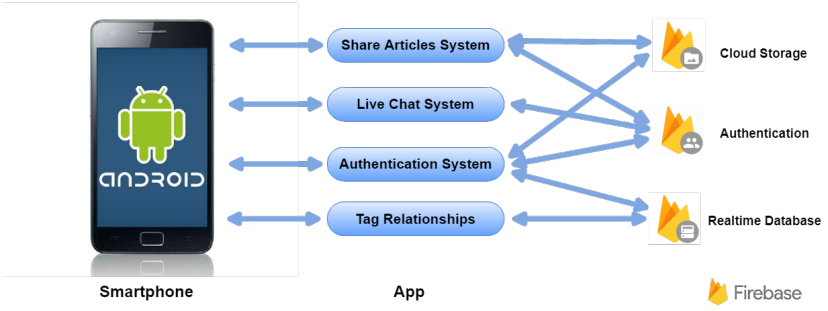

Figure 3.3: Relationships between modules and Firebase services

3.4 Database

Although Firebase has provided a set of commands that help me to store and retrieve data quickly and simply from the database, there is still some issues that I need to pay attention to it.

Firebase Realtime Database is a NoSQL database system where there are not tables nor records, all data is stored as JSON objects[10]. When I try to store new data to the database, it will become a node in the existing JSON structure with an associated key. I can also choose to use my own keys, such as user IDs or semantic names [12]. As what I have explained before, since this database cannot store files like images and videos the application will upload images to Firebase cloud storage first, then get the images' URLs and save them in the database. This means, when the application tries to display an article's information on the device, it needs to check if there is any URL inside, and replace it with the corresponding image. Another problem is that it cannot store data with value null, if the value of a data is null, the data will be deleted automatically from the database.

Figure 3.4 demonstrates the structure of the database. All data will be stored in the Firebase Realtime Database. In this database, there are five sub-databases for storing data related to users, chat rooms, messages, articles and markers on the map. More information about the objects stored in each sub-database are explained below.

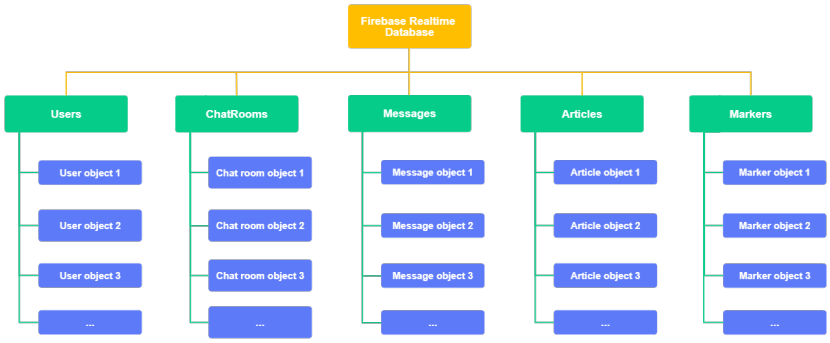

Figure 3.4: Database Structure

```
"user-id":{
    "id": "id",
    "email": "email",
    "name": "name",
    "icon": "icon",
    "list-of-articles": {
        "article1-id":"article1-id",
        "article2-id":"article2-id",
        ...
    }
    "list-of-friends": {
        "user1-id":{
            "message1-id":"message1-id",
            "message2-id":"message2-id",
            ...
        },
        ...
    }
    "list-of-chatrooms": {
        "chatroom1-id":"chatroom1-id",
        "chatroom2-id":"chatroom2-id",
        ...
    }
}
```

Figure 3.5: Structure of User object in the database

Figure 3.5 demonstrates the structure of an User Object in the database. Each user object contains id, email, username and URL of the avatar image. Besides, there are three more lists for storing the id of user's articles, friends and chat rooms.

Figure 3.6 is the structure of a chat room object. Each chat room object contains id, name, URL of the icon image, id of the administrator and a brief announcement about the chat room's topic. Moreover, there are three lists for storing the id of all members, messages and articles shared in this chat room.

The structure of a message object is simple (Figure 3.7). Each message object contains an id, the content of the message, send date, sender and receiver's id.

```
"chatroom-id":{
    "id": "id",
    "name": "name",
    "announcement": "announcement",
    "icon": "icon",
    "administrator": "administrator",
    "list-of-articles": {
        "article1-id":"article1-id",
        "article2-id":"article2-id",
        ...
    }
    "list-of-members": {
        "user1-id":"user1-id",
        "user2-id":"user2-id",
        ...
    }
    "list-of-messages": {
        "message1-id":"message1-id",
        "message2-id":"message2-id",
        ...
    }
}
```

Figure 3.6: Structure of Chat room object in the database

```
"message-id":{
    "id": "id",
    "content": "content",
    "date": "date",
    "sender": "sender",
    "receiver": "receiver"
}
```

Figure 3.7: Structure of Message object in the database

```
"article-id":{
    "id": "id",
    "title": "title",
    "content": "content",
    "date": "date",
    "rating": "rating",
    "author-id": "author-id",
    "author-name": "author-name",
    "location-city": "location-city",
    "list-of-markers": {
        "marker1-id":"marker1-id",
        "marker2-id":"marker2-id",
        ...
    }
    "list-of-comments": {
        "comment1-id":{
            "commenterId":"commenterId",
            "commenterName":"commenterName",
            "content":"content",
            "date":"date",
            "id":"id",
        },
        ...
    }
    "list-of-ratings": {
        "user1-id":"user1-id",
        "user2-id":"user2-id",
        ...
    },
    "list-of-tags": {
        "tag1":"tag1",
        "tag2":"tag2",
        ...
    }
}
```

Figure 3.8: Structure of Article object in the database

Figure 3.8 is the structure of an article object. Each article object contains an id, title, content, id and name of the author and release date. In addition, there are also a list of tags which are related to the article, ratings and comments made by readers, and a list of markers' id which should be displayed the article's map.

Figure 3.9 shows the structure of a marker object. Each marker object contains an id, title, a brief description and its location on the map (latitude and longitude).

```
"marker-id":{
    "id": "id",
    "title": "title",
    "description": "description",
    "latitude": "latitude",
    "longitude": "longitude"
}
```

Figure 3.9: Structure of Marker object in the database

In addition to these five databases, there is one more database designed for storing relationships among tags, which I have mentioned in Chapter 1. By establishing relationships among tags, when the application tries to find articles with a specified tag, it will also access this database in order to get all tags which are related to this specified tag, and then search articles using these tags. In this way, if the keyword does not appear in an article's title and tags, the application will check if any of its tags are related to the keyword, if yes, this article would be added into result list as well. For example, tag "Spain", it has two related tags named "España" ("Spain" in Spanish) and "Madrid" (the capital city). When the user tries to search articles using "Spain" as the keyword, the application will also display articles which contain tags "España" and "Madrid" in the result list, since these two are tags relate to the keyword "Spain". Figure 3.10 shows the structure of this database. In this database, each node's name is a tag, and it children are other tags which are related to this tag. Finding tags related to the keyword means searching if there is a node whose name is the same as the keyword and getting all its children.

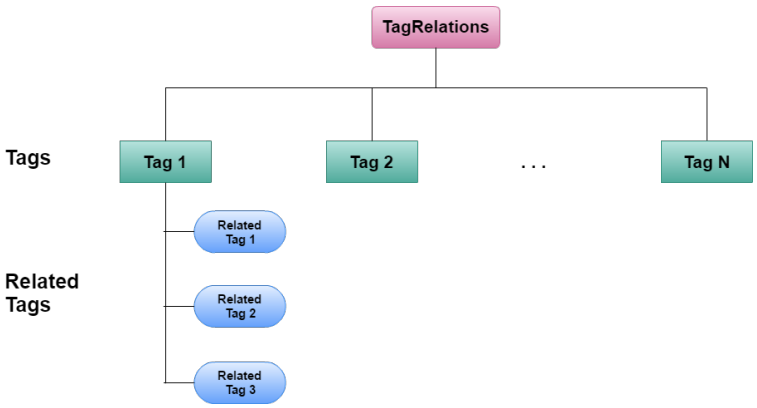

Figure 3.10: Code: Structure of tags database

3.5 User Interface

Design of the user interfaces is very important in the development of an application. The user interface is the space where users interact with the application. It determinates the effectiveness of the operations to the application from users, the usability of the application is up to it, too. It can reduce the time users need for learning how to use the application[13]. In this section, I present the design of the user interfaces of the application, explain the functionalities available in every interface and how to switch between different interfaces. The design of the interfaces is very simple because I don't have much experience on this area, more work will be done in the future to improve the interfaces.

When the user launches the application, the first interface appears is the login page with a logo and a brief description about the application. There are two available options for the user: a button for logging in after entering the email and password, and a link which leads to a register page(Figure 3.11). In the register page, the user needs to fill the form about the email, password and username for the new account. When the user has logged in successfully, the application will jump to the main menu page, where the user can search articles and check the contact list and the account's data.

Figure 3.11: Start using the application

Figure 3.12: Main menu page of the application

As we can see in the figure 3.12, there is a navigation bar in the top of the main menu page. The user can use it to switch pages between the articles page, contact list

page and account's information page. Besides, the user can also slide his/her finger to the right or left to switch pages. There is float button at the right bottom corner with two hidden options: search users and create new chat rooms.

In the account's information page, all basic information about the account will be displayed, such as the account's avatar, email, username, etc. The password will be hidden for the account's security. The user can change the account's avatar, username and password by clicking on them. Email can not be changed, so it is unclickable. There is also a text named "My articles" which leads to a new page where displays all articles belong to the user and a button for logging out (See figure 3.13).

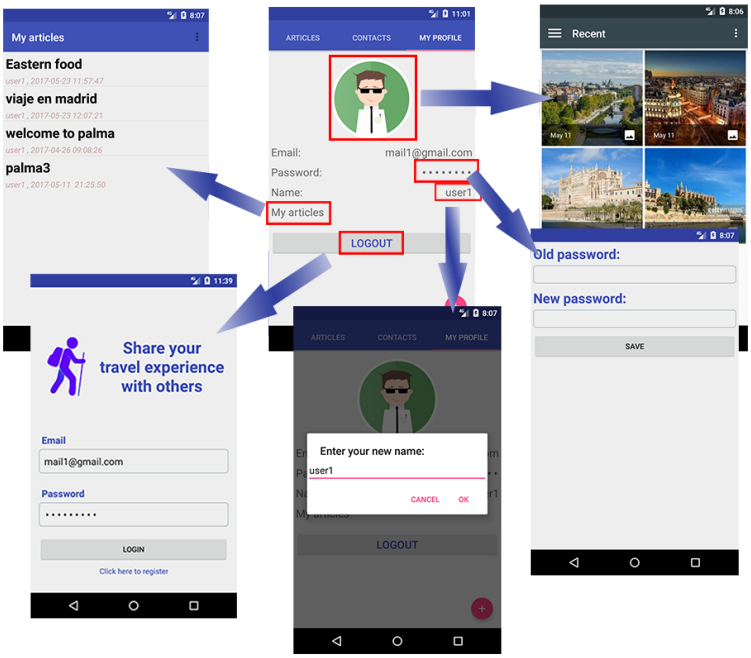

Figure 3.13: Available operations in the account's information page

Search users option hidden in the float button allows user to find other users by entering an email and the information of the found user will be displayed in a new page, including the avatar, username, email, link for checking his/her articles and a button for adding him/her into the contact list (Figure 3.14). Another hidden option allows the user to create a new chat room, the application will display a dialog asking the user to enter a name for the new chat room (Figure 3.15).

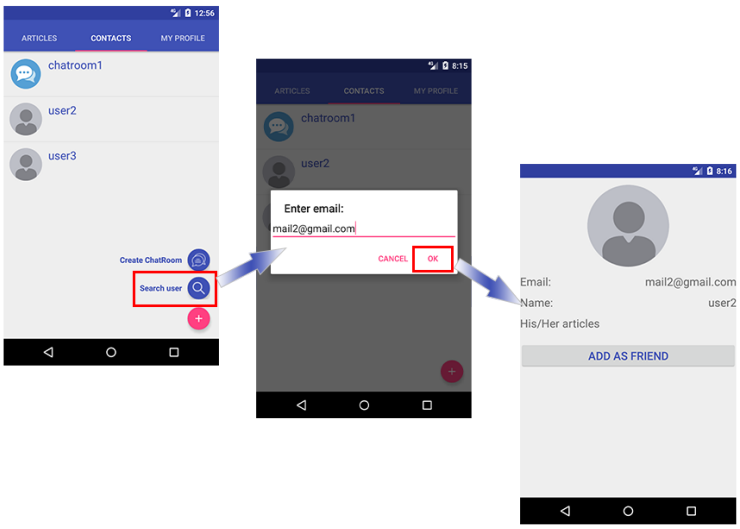

Figure 3.14: Process for searching users

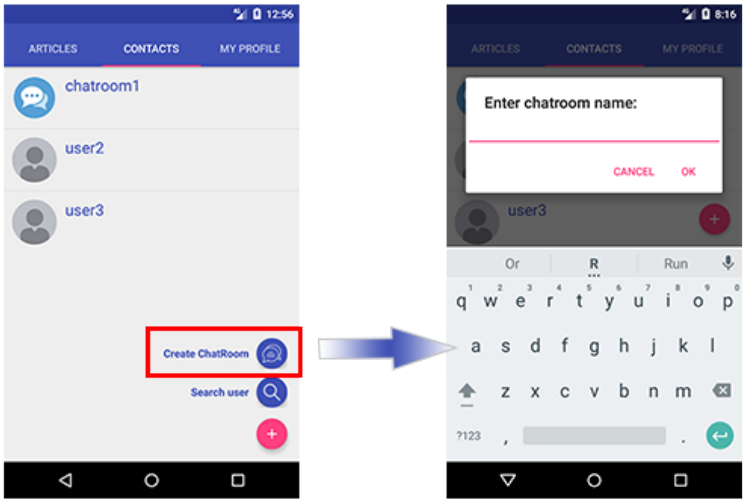

Figure 3.15: Process for creating new chat rooms

The way to start a live chat with friends is going to contact list in the menu page and clicking on friends' name or avatars. The application will jump to a chat page displaying the historical messages in the central area. In the top right corner of this page, there is an option name "INFO" which allows user to check friends' information: avatars, emails, names and articles. In the bottom of the chat page, there is a blank text box that allows the user to enter messages and a button to send them to friends(See figure 3.16).

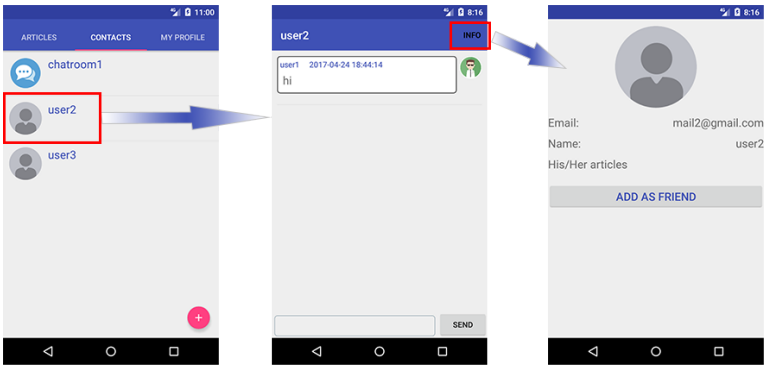

Figure 3.16: Check friend's information

The operation of removing friends from the contact list is simple. The user only needs to long click on friend's avatar or name and a dialog will appear asking user to confirm this operation. The user can click on "YES" to remove friends or cancel it by clicking on "NO" (Figure 3.17). This dialog can prevent the user from removing friends accidentally.

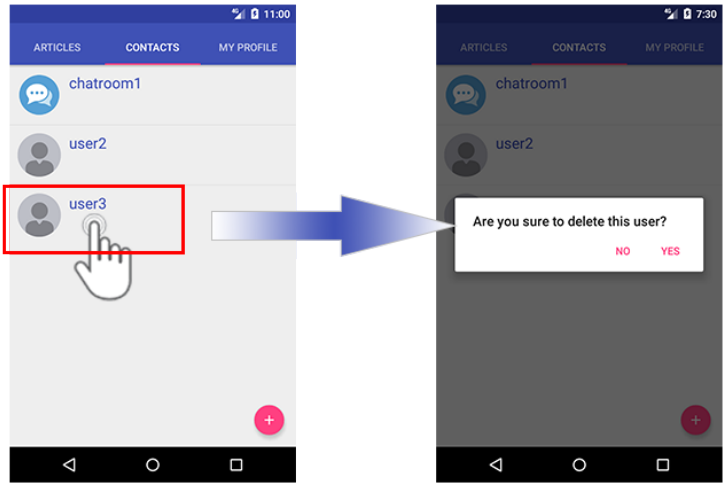

Figure 3.17: Process to remove friends from the list

In the chat room's information page, user can see all information about the chat room, such as room's icon, name, name of the administrator, announcement, members'

list and articles shared by members. The way to modify chat room's information is the same as the way to modify account's information (Figure 3.18). But only the administrator can modify chat room's information. There are two buttons in the bottom of the page, which allow members to invite other users by entering their emails or leave the chat room.

Figure 3.19 demonstrates the way to remove members from the chat room, the user must be the administrator of the chat room, and goes to members list page from the chat room's information page. When the user long clicks on one member's avatar or name, a dialog will appear asking the user to confirm the operation or cancel it.

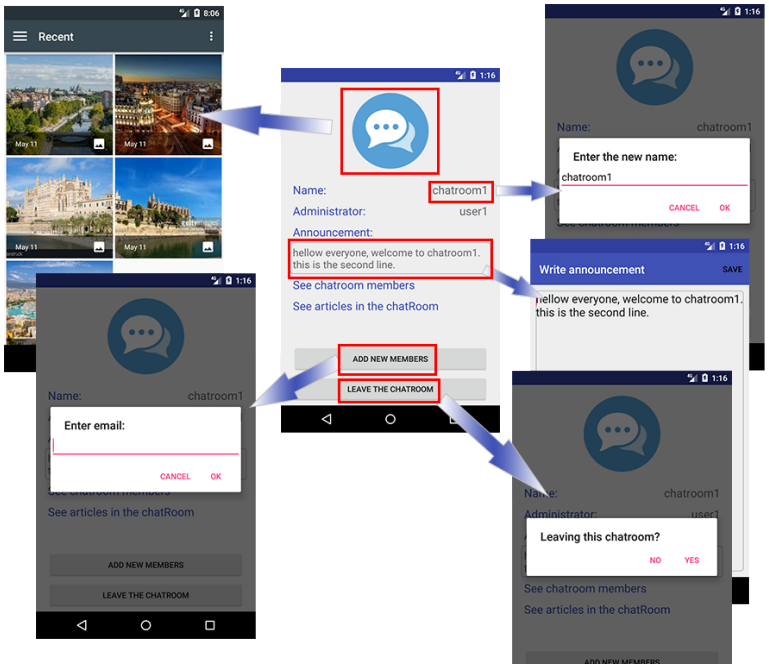

Figure 3.18: Available operations in the chat room's page

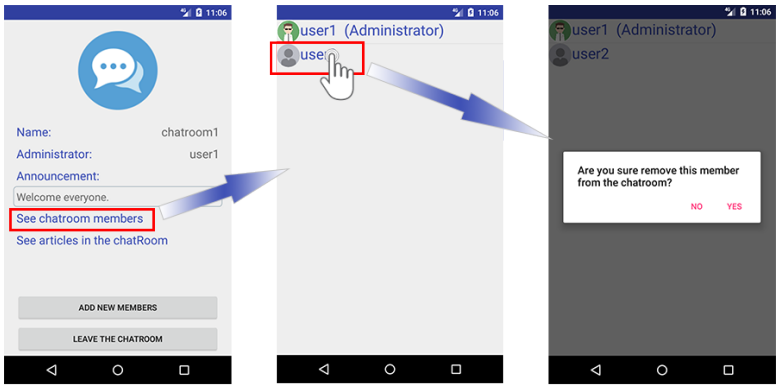

Figure 3.19: Process to remove members from the chat room

In the chat room, members can share their articles with others by adding their articles into the chat room's articles list. The figure 3.20 demonstrates the steps to share

articles in the chat room. At first, the user needs to go to the articles list page from the chat room's information page. In the top right corner of the articles list page, there is an option named "Add", after clicking on this option, the application will display all articles belong to the user. The user needs to choose articles that he/she wants to share, and then click on the "Save" option to add all selected articles into the articles list of the chat room.

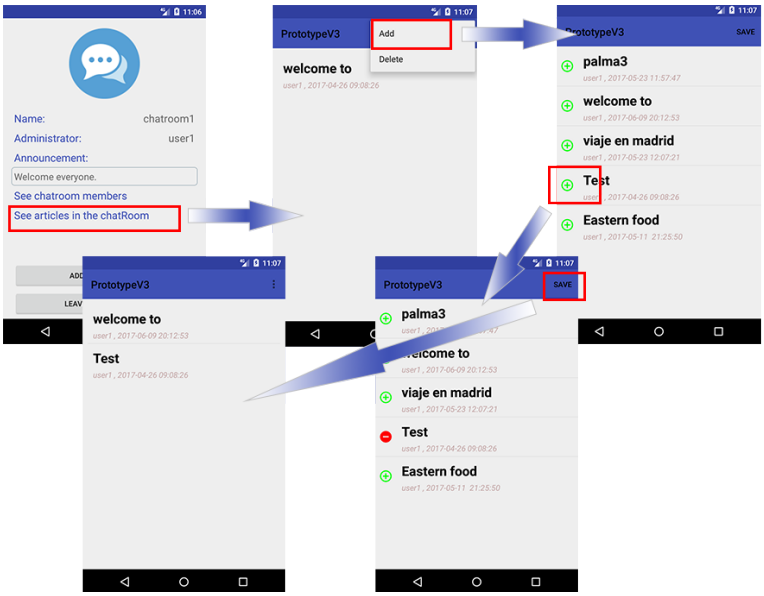

Figure 3.20: Process to share articles in the chat room

The way to remove articles from the article list of the chat room is simple. As we can see in the figure 3.21, there is another option name "Delete" in the chat room's articles page. When the user clicks on it, the application will activate the delete mode, remove icon will appear on the left of each article. After clicking on the remove icon, a dialog will show up in the middle of the screen to ask the user to confirm the operation. The user can deactivate the delete mode by clicking on the "save" option in the toolbar.

Figure 3.21: Process to remove shared articles from the chat room's list

To manage user's articles, the first step is go to the page where displays all articles belong to the user. All articles are displayed in a list, showing the title, author's name and release date. The user can delete the articles by long clicking on them, a dialog will appear asking the user if delete this article or not. In the toolbar of the page, there is an option named "NEW". After clicking on it, the application will open a new blank page, where the user can edit the title and content for the new article. In the right bottom corner of the page, a float button with image icon allows the user to add images stored in the phones to the article (Figure 3.22).

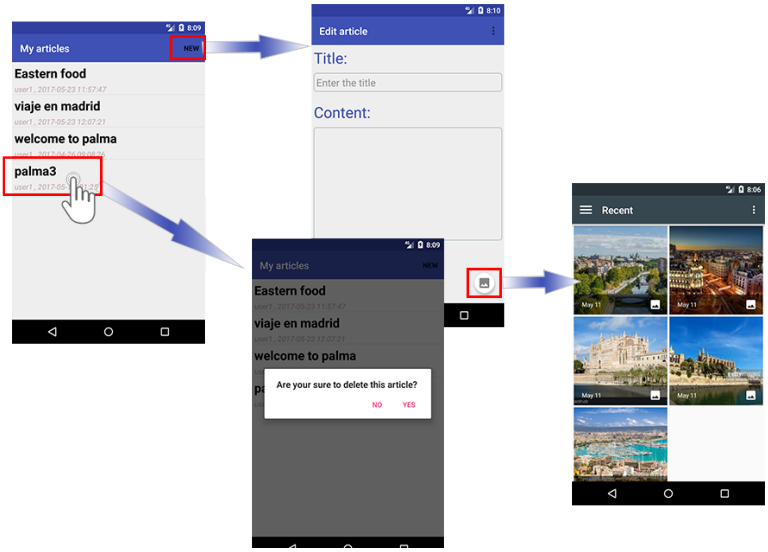

Figure 3.22: Account's articles management

To complement the information of the articles, the user can also edit related maps with markers for his/her articles(See figure 3.23). In the eidt article page, there is an option in the submenu of the toolbar which allows the user to create a related map and add markers on it. In the edit maps page, there is a blank text box where the user can enter the city name and search it on maps or the user can search cities by using his/her finger on the maps. To add a marker, the user needs to long click on the map, a marker will appear in the position of the finger. Each marker has a title and a brief description which can be edited by the user. To remove markers,first the user has to select one marker on the map, then click on the delete icon which is located in the left bottom corner of the page.

In the submenu of the toolbar, there is an option named "set location". This option allows the user to set a location for the article. When the user chooses to search articles on the map, a marker will be displayed in this location on the map with the title of the associated article. When the user clicks on the title, a new page will appear and display the content of this article. The steps to "set location" is very similar than the option "add map", but it only allows the user to set one marker on the map, when the user set a new one, the old one will be removed automatically. Besides, the user does not need to edit the information of the marker, because its information will be the title of the associated article.

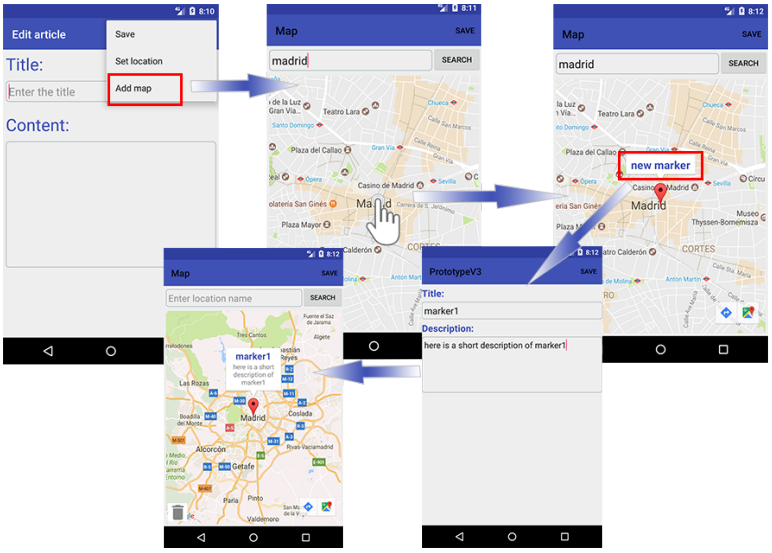

Figure 3.23: Process to edit related maps for articles

In the main menu of the application, there is a tab page where the user can see the latest shared articles. In this page, there is a text box allows the user to enter keywords, a search icon to start searching articles with keywords and a map icon to search articles on map(Figure 3.24). When the user clicks on the map icon, the application will open a map page and all articles will be distributed on this map based on the information of the articles' related maps. In this ways, when the user moves to a city, all articles which are near to this place will be displayed on the map.

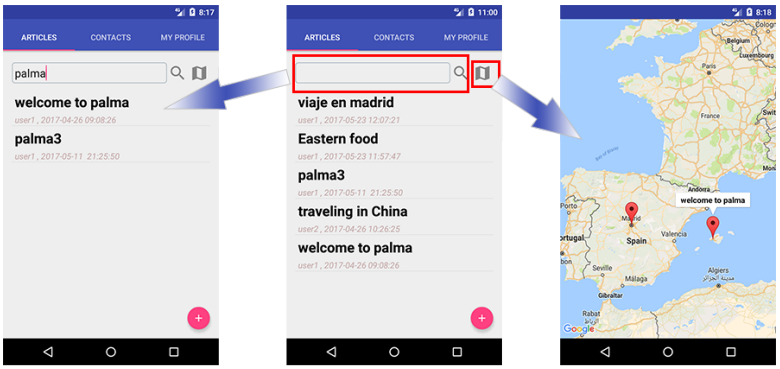

Figure 3.24: Two ways for searching articles

In the page where shows the information of an article, there is a rating bar below the area where displays article's information. The user can see the average rating of this article, and he/she can also choose to make his/her own rating. The user can make a rating from 0.5 star to 5 stars by clicking on the rating bar. After that, the user needs to click on the "submit" button to send his/her rating number to the server. A message

will be displayed under the rating bar, telling the user what is his/her rating (Figure 3.25).

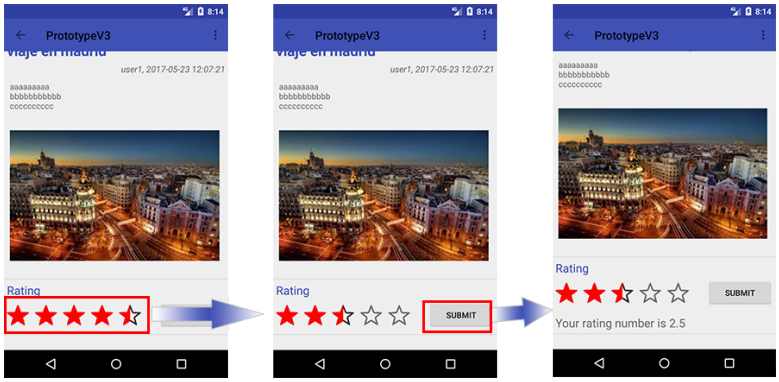

Figure 3.25: Process to rate articles

In addition to making a rating for the article, there are some more options for the user in the submenu of the toolbar(Figure 3.26). The first option is to modify the article, but only the author of this article has the permission to do it. The second option is to see the related map. When the user clicks on the "Maps" option, the article's related map will show up in a new page. The third one is to see the comments of this article. The comments will be displayed in a new page, there is a button in the right bottom corner of the page, this button will lead the user to a new page, where he/she can edit a new comment for the article(Figure 3.27). The fourth is to add tags for this article. A dialog will appear and ask the user to enter new tags.

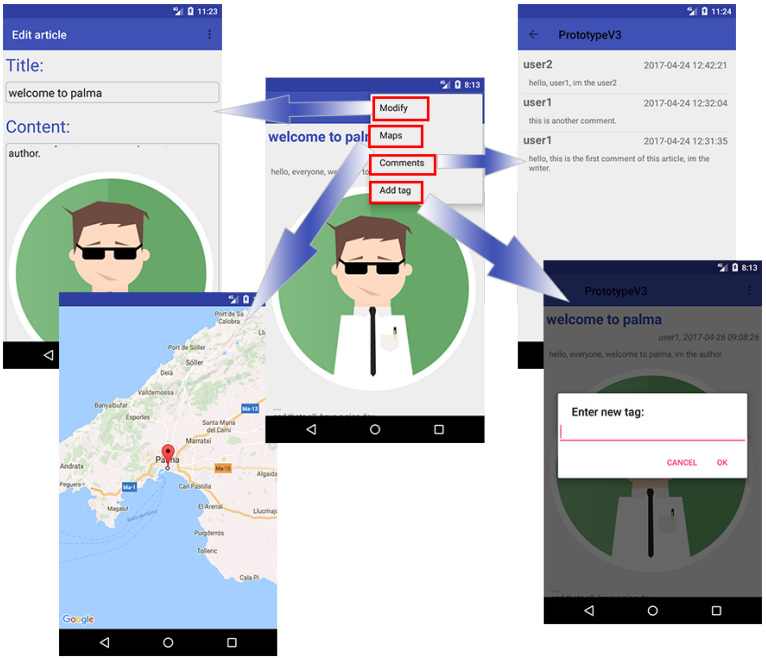

Figure 3.26: Available operations in the article page

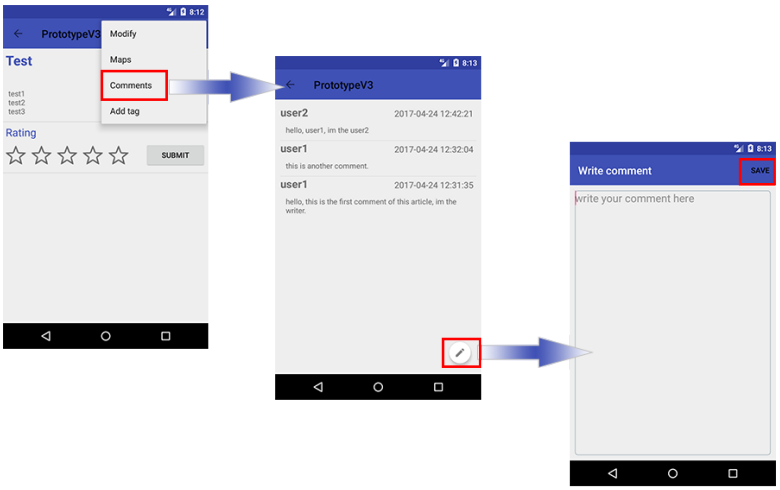

Figure 3.27: Writing comments for an article

There is an area below the rating bar for displaying all tags of the article. The way to delete these tags is the same as other delete operations before. The user only needs to long click on one tag, then a dialog will appear and ask the user if delete this tag or not(Figure 3.28).

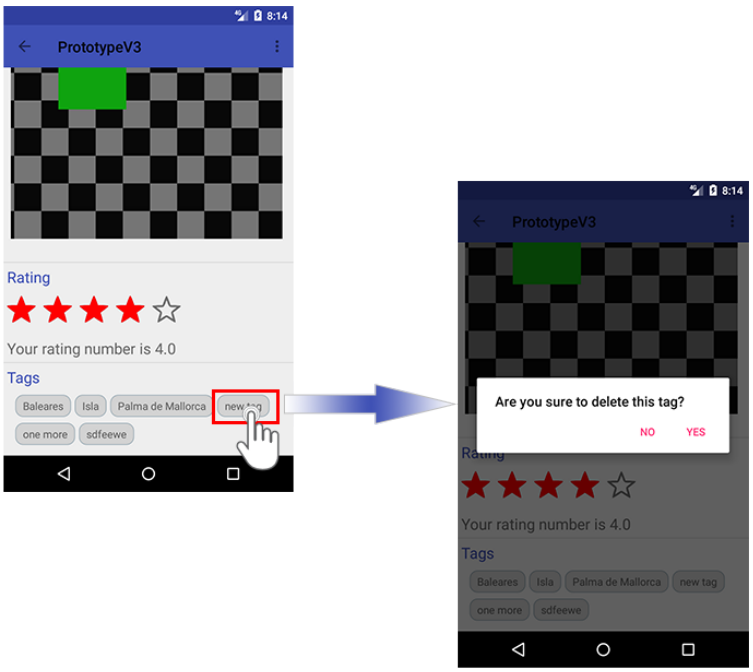

Figure 3.28: Process to delete tags of an article

CHAPTER 4

IMPLEMENTATION

This chapter shows the building of the application with some captures of codes. In the design chapter, I have mentioned that, for building the application, I need to use several services provided by Firebase, such as the authentication service, realtime database and cloud storage. To be able to use these services, at first, I need to install Firebase Plugin in Android Studio and add it into the application project. In Firebase website, a new project will be created which is associated with this application, all services used in the application can be supervised from this website. It is like a console for the Firebase services, I can monitor and manage the status of all services, including the data and files stored in the database and could storage (Figure 4.1 is a capture of the website).

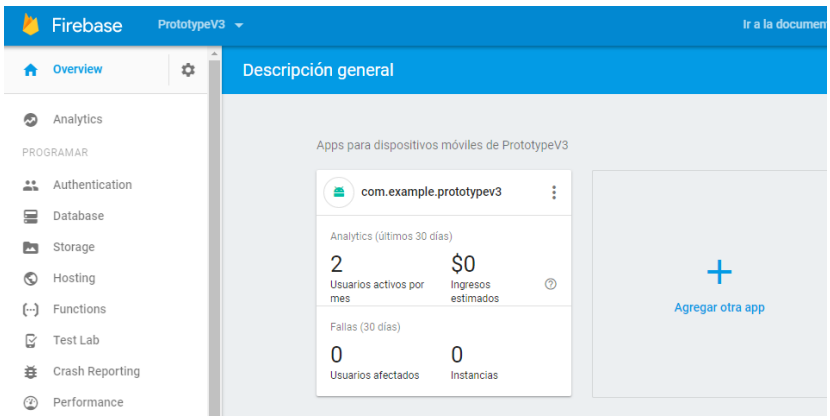

Figure 4.1: Firebase project website

4.1 Authentication System

The authentication system mainly provides three functions for the application: register new accounts, authenticate the password and change the password. Firebase has provided methods for realizing these functions. What I need to do is using these methods in my application code. The methods for registering accounts and authenticating password need the email and password, the application will send requests to Firebase server with these methods, and act based on the response received from the server(Figure 4.2). As for the request of changing password, old password and new password, both are needed. Firebase will check old password first in order to get the permission for replacing old password with the new one(Figure 4.3).

```
//try to create new account with the email and password
auth.createUserWithEmailAndPassword(email, password)
        .addOnCompleteListener(RegisterActivity.this, new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                progressDialog.dismiss();
                if(task.isSuccessful()){
                    auth = FirebaseAuth.getInstance();
                    FirebaseUser user = auth.getCurrentUser();
                    if (user != null) {
                        createNewUser(user.getUid(), email, name);
                        startActivity(new Intent(RegisterActivity.this, MenuActivity.class));
                    }
                } else {
                    //the task will be fail if the email has already been used
                    Toast.makeText(RegisterActivity.this, "Could not register, please try again",T
                }
            }
        });
```

Figure 4.2: Registering new accounts code

```
AuthCredential credential = EmailAuthProvider.getCredential(email, oldPassword);

user.reauthenticate(credential)
        .addOnCompleteListener(new OnCompleteListener<Void>() {
            @Override
            public void onComplete(@NonNull Task<Void> task) {
                if (task.isSuccessful()){
                    String newPassword = editTextNewPassword.getText().toString();
                    user.updatePassword(newPassword)
                            .addOnCompleteListener((task) -> {
                                    if (task.isSuccessful()){
                                        Toast.makeText(getApplicationContext(), "Password has been updat
                                        Intent intent = new Intent(getApplicationContext(), MenuActivity
                                        intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                                        startActivity(intent);
                                        editTextOldPassword.setText("");
                                        editTextNewPassword.setText("");
                                    } else {
                                        Toast.makeText(getApplicationContext(), "Failed to update passwo
                                    }
                                });
                } else {
                    Toast.makeText(getApplicationContext(), "The old password is incorrect.", Toast.LENG
                }
            }
        });
```

Figure 4.3: Changing account's password code

When a new account has been registered in the Firebase server, the application will create a user object (the structure of user object has been explain in the design chapter) with data of the new account, including the email, user's name and avatar(See figure 4.4). The image of the avatar will be the default image in the cloud storage, which can be changed by the user later.

```
private void createNewUser(final String uid, final String email, final String name) {
    storageReference.child("default/avatar.jpg").getDownloadUrl().addOnSuccessListener(new OnSuccessListener
        @Override
        public void onSuccess(Uri uri) {
            String avatarURL = uri.toString();
            Map<String, Object> map = new HashMap<~>();
            map.put("Email", email);
            map.put("UserId",uid);
            map.put("Name", name);
            map.put("Avatar", avatarURL);
            databaseReference.child("Users").child(uid).setValue(map);
        }
    });
}
```

Figure 4.4: Create user object code

4.2 Live Chat System

In the live chat system, the key function is the live chat service between users, the sent messages need to be received immediately. To realize this function, I set up a listener to monitor the status of the database where store user's data. Any change happened in the user's data will trigger the listener. When a new message has been sent by a user, the application will create a message object recording the information of this new message, and store it in the database. Moreover, the sender and receiver's messages list will also be updated by adding the id of this new message(Figure 4.5). This move will trigger the listener of sender and receiver's application since there is a change happened in their data, then the application will read user's data again and implement the new data(Figure 4.6). In this way, the messages displayed by the application will be updated, user will be able to see the new message.

```
Message m = new Message(message_id, user.getUid(), receiver_id, message, strDate);
databaseReference.child("Messages").child(message_id).setValue(m);

Map<String, Object> map = new HashMap<~>();
map.put(message_id, message_id);
if (receiver_type.equals("user")){
    databaseReference.child("Users").child(user.getUid()).child("ListFriends").child(receiver_id).updateChildren(map);
    databaseReference.child("Users").child(receiver_id).child("ListFriends").child(user.getUid()).updateChildren(map);
} else if (receiver_type.equals("room"))  {
    databaseReference.child("ChatRooms").child(receiver_id).child("ListMessages").updateChildren(map);
}
```

Figure 4.5: Sending messages code

The way to manage contact list is the same. The application has set up a listener to monitor user's contact list. Adding new friends means new friends' id will be added to user's contact list. Removing friends is deleting their id from the contact list. Both actions will trigger the listener, make the application to update the contact list displayed on the devices.

4.3 Share Articles System

The way to store articles is the same as the way to store messages, just replacing the message object with article object. The application sets up a listener to monitor articles as well. Any changes happed to the articles will trigger the application to re-read the database for updating the information of articles. Any changes in the articles will be seen by readers immediately, such as modification of article's content, new tags, etc. To

```
databaseReference.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        list.clear();
        FirebaseUser user = auth.getCurrentUser();
        Iterator i = null;
        if (receiver_type.equals("user")){
            if (dataSnapshot.child("Users").child(receiver_id).exists()){
                String title = dataSnapshot.child("Users").child(receiver_id).child("Name").getValue().toString();
                toolbar.setTitle(title);
                i = dataSnapshot.child("Users").child(user.getUid()).child("ListFriends").child(receiver_id).getChildren().iterator();
            }

        } else if (receiver_type.equals("room")) {
            if (dataSnapshot.child("ChatRooms").child(receiver_id).exists()){
                String title = dataSnapshot.child("ChatRooms").child(receiver_id).child("Room_name").getValue().toString();
                toolbar.setTitle(title);
                i = dataSnapshot.child("ChatRooms").child(receiver_id).child("ListMessages").getChildren().iterator();
            }
        }

        if (i != null){
            while (i.hasNext()){
                String message_id = ((DataSnapshot)i.next()).getKey();
                String content = dataSnapshot.child("Messages").child(message_id).child("content").getValue().toString();
                String receiver = dataSnapshot.child("Messages").child(message_id).child("receiver").getValue().toString();
                String sender = dataSnapshot.child("Messages").child(message_id).child("sender").getValue().toString();
                String date = dataSnapshot.child("Messages").child(message_id).child("date").getValue().toString();
                list.add(new Message(message_id, sender, receiver, content, date));
            }
        }

        messagesAdapter.notifyDataSetChanged();
    }
```

Figure 4.6: Receiving messages code

manage images, the application uploads all images files to Firebase could storage first, when the images have been uploaded successfully, the application will send a request to the server to get the images' URLs and save them in the realtime database(Figure 4.7).

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (resultCode == RESULT_OK && data != null && data.getData() != null && requestCode == PICK_IMAGE_REQUEST){
        final Uri filePath = data.getData();
        final String image_name = UUID.randomUUID().toString();
        StorageReference storageRef = storageReference.child("Images/"+image_name+".jpg");
        storageRef.putFile(filePath).addOnSuccessListener((OnSuccessListener) (taskSnapshot) -> {
                storageReference.child("Images/"+image_name+".jpg").getDownloadUrl()
                        .addOnSuccessListener((OnSuccessListener) (uri) -> {
                        String imageURL = uri.toString();
```

Figure 4.7: Uploading images code

In the application, I have used the widgets EditText and TextView for displaying the content of an article. Information displayed with EditText is editable by the user, and it can not be edited in TextView. But there is a problem which is both of them cannot display images directly, and my solution is using SpannableString and ImageSpan. ImageSpan allows application to display images in the TextView and EditText, SpannableString can define a string which will substitute the image when the application tries to get the value of the TextView or EditText(Figure 4.8). In this way, when the content of an article has been saved the database, its form will like "texts + image's URL + texts + image's URL + ...". In order to be able to easily find all URLs out from the content of the article, I have added a prefix and suffix to all URLs, and the format of the string will be like " <img src=' " + URL + " > ".

To display the content of an article, the application needs to find all URLs out from the content and substitute them with the corresponding images. Since all images' URLs have been added a prefix and suffix before been savd in the database, what the application needs to do is to extract all strings which have the same prefix and suffix with a format like " <img " + string + " > "(Figure 4.9). All strings which do not the

```
String imageURL = uri.toString();
InputStream inputStream = null;
try {
    //get the image and put it in the position where the cursor is
    Editable edit_text = inputContent.getEditableText();
    int index = inputContent.getSelectionStart();
    inputStream = getContentResolver().openInputStream(filePath);
    Drawable d = Drawable.createFromStream(inputStream, filePath.toString());
    //String image_code = "<br><img src='" + imageURL + "' style='max-width: 90%; h
    String image_code = "<img src='" + imageURL + "'";
    SpannableString ss = new SpannableString(image_code);
    WindowManager wm = EditArticleActivity.this.getWindowManager();
    int width = wm.getDefaultDisplay().getWidth() - 30;
    int height = width * d.getIntrinsicHeight() / d.getIntrinsicWidth();
    d.setBounds(0, 0, width, height);
    ImageSpan span = new ImageSpan(d, ImageSpan.ALIGN_BASELINE);
    ss.setSpan(span, 0, image_code.length(), Spannable.SPAN_INCLUSIVE_EXCLUSIVE);
    if (index<0 ||index >= edit_text.length()){
        edit_text.append("\n");
        edit_text.append(ss);
    } else {
        edit_text.insert(index, "\n");
        edit_text.insert(index+1, ss);
    }
    //inputContent.append(ss);
} catch (FileNotFoundException e) {
    e.printStackTrace();
}
```

Figure 4.8: Displaying images in EditText code

specified prefix and suffix, will be display on the TextView directly. And the remaining ones are URLs of images, the application needs to remove their prefix and suffix, then convert URLs to images(Figure 4.10).

```
String regex = "<img .*>";
Pattern p = Pattern.compile(regex);
Matcher m = p.matcher(content);
```

Figure 4.9: Extracting URLs from the content code

4.4 Support of multi-languages

Support of multi-languages is a non-functional requirement of the application raised in the design phase, its purpose is to improve the usability of the application allowing users to choose the language they like. To realize this idea, I have created three XML files named "strings" under the "values" directory in the application. These three files have stored all strings which will be displayed in the application interfaces(Figure 4.11). In the codes of the application interfaces, these strings are replaced with the strings' name in the three language files.

When the application is launched, it will detect the language of the OS automatically and choose to use the file with the same language. If the language of the OS is not one of these three languages, the application will use English as the default language of the application.

1. get the URL by removing the prefix and suffix, then convert the URL to a bitmap

```
//check if the string is an URL of an image
if (string.startsWith("<img src")){
    text = string.substring(10, string.length()-41);
    //URL url = new URL(string.substring(10, string.length()-4));
    //remove the prefix and suffix
    URL url = new URL(string.substring(10, string.length()-39).replace("amp;", ""));

    Bitmap bitmap = BitmapFactory.decodeStream(url.openConnection().getInputStream());
    return bitmap;
```

2. convert the bitmap to an image and add it to TextView

```
Drawable drawable = new BitmapDrawable(getResources(), bitmap);
String image_code = list.get(count);
SpannableString ss = new SpannableString(image_code);
WindowManager wm = DisplayArticleActivity.this.getWindowManager();
int width = wm.getDefaultDisplay().getWidth() - 50;
int height = width * drawable.getIntrinsicHeight() / drawable.getIntrinsicWidth();
//drawable.setBounds(0,0,drawable.getIntrinsicWidth(), drawable.getIntrinsicHeight());
drawable.setBounds(0,0,width, height);
ImageSpan span = new ImageSpan(drawable, ImageSpan.ALIGN_BASELINE);
ss.setSpan(span, 0, image_code.length(), Spannable.SPAN_INCLUSIVE_EXCLUSIVE);
articleContent.append(ss);
```

Figure 4.10: Converting URL to image code

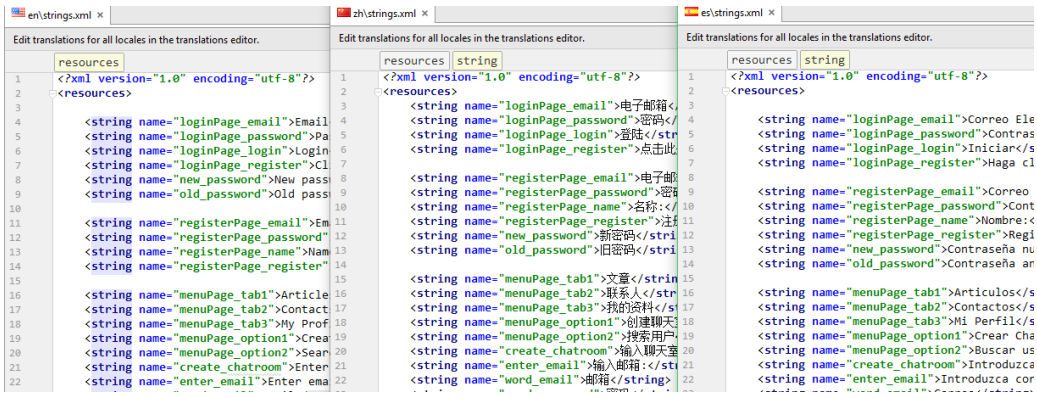

Figure 4.11: Files for supporting different languages

Considering the possibility of using different languages between the application and devices, an option has been added allowing users to change the application's language without changing the language of the devices. The figure 4.12 is the code for changing the application's language. The application sends a request to the system indicating the type of the language with the variable named "lang", if its value is "es" means using the file with strings in Spanish, "en" for English and "zh" for Chinese .

```
//the variable "lang" indicates the language: "es" for spanish, "en" for english and "zh" for chinese
public static void changeLang(Context context, String lang) {
    Locale myLocale = new Locale(lang);
    Locale.setDefault(myLocale);
    android.content.res.Configuration config = new android.content.res.Configuration();
    config.locale = myLocale;
    context.getResources().updateConfiguration(config, context.getResources().getDisplayMetrics());
}
```

Figure 4.12: Changing application's language code

4.5 Database for tags

In the database which stores relationships among tags, all its data is managed by me, the developer of the application. The application accesses this database when it tries to get all tags related to the keywords. Figure 4.13 shows some relationships saved the database. "spain" has two children named "madrid" and "barcelona", this means these two are tags related to "spain".

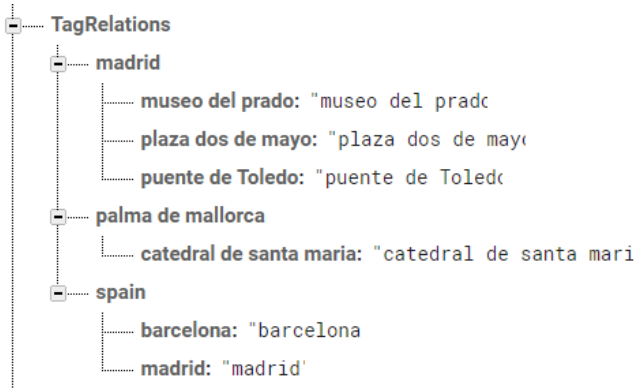

Figure 4.13: Capture of the tags database

Figure 4.14 is the code which tries to search all tags related to the keyword entered by the user. When the keywords do not appear in an article's title or tags, the application will run this code for getting all related tags and then check if any of the tags related to the keywords has appeared in the article's title or tags.

To find all related tags, the application runs with the following steps:

1. Add the keyword to a queue.
2. If the queue is not empty, get the first string in the queue, and then remove it from the queue.
3. Get this string's related tags by reading its children in the database.
4. Add all related tags to the queue and a list which stores all related tags and then back to step 2.
5. When the queue is empty, all found related tags are stored in the list.

For example, if the keywords is "spain", based on the data of previous database Figure 4.13, its related tags will be "barcelona", "madrid" and three tags which are related to "madrid": "museo del prado", "plaza dos de mayo" and "puente de Toledo".

```
final List<String> queue;
queue = new ArrayList<>();
queue.clear();
queue.add(keyWord);
while(queue.size()>0){
    String tag = queue.get(0);
    tagsList.add(tag);
    queue.remove(0);
    if (dataSnapshot.child("TagRelations").child(tag).exists()){
        Iterator iii = dataSnapshot.child("TagRelations").child(tag).getChildren().iterator();
        while(iii.hasNext()){
            String relatedTag = ((DataSnapshot)iii.next()).getKey();
            queue.add(relatedTag);
        }
    }
}
```

Figure 4.14: Search all related tags code

4.6 Levenshtein Algorithm

In addition to creating database for storing relationships among tags, I have also implemented another algorithm for improving the search function. This algorithm allows the keywords contain some mistyped or missing characters, and its name is Levenshtein distance [14]. Levenshtein distance is a algorithm for calculating the difference between two strings. The Levenshtein distance between two strings is the minimum number of single-character edit (insertions, deletions or substitutions) required to change one word into the other. The appliction will try to calculate the Levenshtein distance between the keywords and article's title and tags, if the result distance is short, for example, is less than 3, which means, the characters of two strings are very similar. So there is a possibility that the keywords are mistyped by the user, with some characters in disorder or mistyped. Figure 4.15 is the code in the application for calculating the Levenshtein distance between two string. To guarantee the accuracy of the search result, I have set a criterion for the Levenshtein distance, the value of the Levenshtein distance must be less or equal to 2, which means only two characters in the keywords can be missing, mistyped or disordered.

```
//algorithm of Levenshtein distance. calculate the distance between two strings
public static int distance(String a, String b){
    a = a.toLowerCase();
    b = b.toLowerCase();

    int [] costs = new int [b.length() + 1];
    for (int j = 0; j < costs.length; j++){
        costs[j] = j;
    }

    for (int i = 1; i <= a.length(); i++){
        costs[0] = i;
        int nw = i - 1;
        for (int j = 1; j <= b.length(); j++){
            int cj = Math.min(1 + Math.min(costs[j], costs[j-1]), a.charAt(i - 1) == b.charAt(j - 1)? nw : nw + 1);
            nw = costs[j];
            costs[j] = cj;
        }
    }
    return costs[b.length()];
}
```

Figure 4.15: Levenshtein distance algorithm code

4.7 Protection of data

The data stored in the server can be classified into two types: one is related to the shared articles, such articles' information, comments, etc. Another the messages among users. Considering the former one's feature, it does not need a high level of security, because its essential purpose is to share these information with others. But the latter is different, it stores all messages among users, which are belong to users' privacy. In order to protect the users' privacy, these messages need to be encrypted before being sent to the server from devices, even the developer of the application could not get the messages' information from the server. If these messages have been leaked, the hackers would not be able to know the content of the messages either.

To encrypt the data, I use Cipher class, which is available in JAVA providing a series of functions to transform the data using different encryption algorithms, such as AES, DES, RSA, etc [15]. Figure 4.16 is the code which encrypts the data using AES algorithm, the parameter "rawKey" stores the key which is required in AES algorithm to encrypt and decrypt the data. This key is generated based on the message sender's ID. Another parameter named "original" stores the message's content.

```
private static byte[] encrypt(byte[] rawKey, byte[] original) throws Exception {
    SecretKeySpec skeySpec = new SecretKeySpec(rawKey, "AES");
    Cipher cipher = Cipher.getInstance("AES");
    cipher.init(Cipher.ENCRYPT_MODE, skeySpec, new IvParameterSpec(
            new byte[cipher.getBlockSize()]));
    byte[] encrypted = cipher.doFinal(original);
    return encrypted;
}
```

Figure 4.16: Data encryption code

The decryption of the message need to use the AES algorithm with the encrypted text and the same key generated from the message sender's ID (Figure 4.17). If the key is correct, the function will return the original text of the message.

```
private static byte[] decrypt(byte[] rawKey, byte[] encrypted) throws Exception {
    SecretKeySpec skeySpec = new SecretKeySpec(rawKey, "AES");
    Cipher cipher = Cipher.getInstance("AES");
    cipher.init(Cipher.DECRYPT_MODE, skeySpec, new IvParameterSpec(
            new byte[cipher.getBlockSize()]));
    byte[] decrypted = cipher.doFinal(encrypted);
    return decrypted;
}
```

Figure 4.17: Data decryption code

CHAPTER 5

TIMETABLE AND LEGALIZATION

5.1 Timetable

Table 5.1 shows the duration of major tasks in the project, from the market research to the documentation of the whole project. The tasks are sorted in the order of execution and the duration of each task is calculated in hours.

Task	**Duration (Hour)**
Market Research	10
Analysis of Requirements	10
Design of Use Cases	20
Design of Architecture	50
Design of Database	50
Design of User Interface	70
Building of Authentication System	20
Building of Live Chat System	80
Building of Share Articles System	100
Building of Tags Relationships Database	30
Building of Multi-languages Support	15
Application Tests	20
Documentation	150
Total	625

Table 5.1: Project's Timetable

5.2 Legalization

Before releasing the application officially, it is important to make sure that the application is legal and meets the requirements set by the local government. If the application hasn't been approved by the government, the developer cannot release the application to the market. Here I use the Spanish government as an example. Considering the application has the features about messages between users and upload of images, in Spain, the application must meet the requirements of a law name "Pretección de Datos de Carácter Personal" (LOPD). Below is a list of the major requirements from LOPD [16] [17].

1. Register related files in AEPD (Agencia Española de Protección de Datos).

2. The application can only collect data which is required for processing the operation of the application. The data cannot be used for other unknown purposes.

3. Before collecting data, the application must inform users first and get their permissions.

4. The company must protect the security of data, prevent risks like data loss, alteration or access without permissions. Some special data related to users' ideology, religion, belief must be treated with utmost caution.

5. The company needs to inform its employees and collaborators about their duty to guarantee data privacy.

6. The delivery of data must contain legal purposes and prevent the risk of leak, moreover, the data owners must be informed about the delivery's motive and purpose.

7. If the company needs to give the data to a third party, it must formalize this operation with contract, declaring details such as the process related to the data, the purpose, measures for guaranteeing data's security, etc.

8. The company must be attention to users' rights, offer them adequate ways to exercise their rights.

9. When the company needs to make an international transfer of data, it must make sure that the operation has complied with related regulations and take measures to guarantee data's security.

Among these nine requirements, only requirement 2, 3 and 4 are related to the development of the application, rest of them are not in the scope of the project, they belong to the phase of release and maintenance in the future. About the requirement 2 and 3, in this application, it requires two permissions for its functionality, one is internet access, it allows the application to send and receive data from server. And the second is access devices' image files, when the user tries to add images into his/her articles, the application displays all images stored in devices and updates the selected image to the server. So before being installed in devices, the application has to ask users for these two permissions, the current version of application does not have this functionality, because it is only a prototype for tests. This task is belong to release phase. About the requirement of security, in the application, all data is store in Firebase server. Firebase is a company dedicated to the development of platforms for mobile and web application development, which is belong to Alphabet Inc. Its ability for guaranteeing servers' security is pretty reliable. Moreover, the sensitive data has been encrypted before being store in the database, in this way either the developers of the application or the hackers are able to get the information hidden in the encrypted data. To prevent

users from getting other users' private information, in Firebase database, I have defined rules to check users' id before reading or writing data of a user object (Figure 5.1).

```
// These rules grant access to a node matching the authenticated
// user's ID from the Firebase auth token
{
  "rules": {
    "users": {
      "$uid": {
        ".read": "$uid === auth.uid",
        ".write": "$uid === auth.uid"
      }
    }
  }
}
```

Figure 5.1: Rules in database

CHAPTER 6

CONCLUSION

The objective of this project is to develop a mobile application which facilitates and promotes the communication and exchange of information among tourists. Considering Android large market share in the world, I have decided to make the application working on Android OS. To achieve the goal of the project, the application has three main modules which offer two ways for the communication and exchange of information and an enhancement for the search function. The first one is the Articles Share System for sharing articles about travel experience, feelings and suggestions, the second is the Live Chat System, it allows users to start a live chat with others and all messages will be encrypted to protect users' privacy. The way to enhance search function is establishing relationships among tags, which allows the application be find proper articles even if the keyword does not appear in the title or tags. These three modules have been constructed and implemented successfully to the application with the functions proposed in the design phase, but they are not perfect yet. For example, the live chat service can only support texts, users are not able to send other kind of messages like emojis or images. The Articles Share System also requires a filter for searching articles, but considering the limited space in the screen, I did not add filters in the application. The Authentication System has been done as well, so far, it allows users to register with emails. In the future improvement phase, I will try to make it supports registration with accounts of other social medias, like Facebook, Twitter. As for the application's user interface, since it is my first time to design this, what I have tried to do is to make it simple enough so the users would not need much time to learn how to use the application.

Establishing relationships among tags is just one of the works I have done for enhancing the search function with keywords. This requires the developer of the application to update the database by storing new relationships among tags constantly. It will be a hard work in the phase of maintenance. Another enhancement is using Levenshtein algorithm. With this algorithm, the application is able to calculate the similarity between keywords and articles' titles and tags, reduce the impact of mistyped or missing characters appeared in the keywords.

After testing the application for a while, I think there are two critical problems that

the application needs to face. The first problem is the efficiency of editing articles, especially for an article with large amounts of texts and images. When I was editing an article on the phone, I noticed that the the phone's screen space is very limited, it would be hard to know how is the whole article. When the keyboard appears, it would occupy at least a third space of the screen, and the rest space could only displays a few lines of words or just an image. It would make the editor to be lost easily, and is not able to know how is the article at the moment, if there is any mistakes or missing content in the article. This is also the complaint I have received from my friends when they were using this application. To resolve this problem, I already contemplate an alternative option for users: edit articles on PC. The database and cloud storage, including the authentication service of Firebase could also be accessed from websites, so it is feasible to edit articles on PC and read them on mobile phone. But website programming is not in the scope of this project, this project only dedicate its works to developing the application, so I did not do this work, it will be the task of new project in the future. Another problem is how to attract users to use this application and share their experience, editing an article is a work much harder than reading it. Without the participation of users, the application is nothing. So in my plan, I will add one more module for this application, the tipping system. This system will allow users to give little money from their bank account to the articles they like, and the author of the articles will receive this money, just like the tipping in restaurants. This feature can encourage the authors to publish more valuable articles, and the average quality of the articles will also be improved. But due to the lack of related knowledge, for now, I am not able add this into my application. In addition to these two problems, there are more works that need to do during the improvement and maintenance phase, for example, improve the application's user interfaces by optimizing the size of each component and implementing appropriate colors, reduce required steps for each operation.

This project has been my first contact with Android, and personally, I have learned a lot from it. Developing an application is much more complicated than I thought before. I have to do tasks like researching the market, analyzing possible requirements and use cases of the application , designing the application for achieving these requirements in an adequate way, etc. The programming is just one of the steps toward to the success. Furthermore, using appropriate tools is quite important, it could influence the efficiency of the work strongly and save a lot of time. In this project, several services provided by Firebase truly help me a lot to develop this application and build the modules. For example in this project, thanks to the synchronization feature of Firebase realtime database, the building of live chat service has been quite easier.

BIBLIOGRAPHY

[1] "Trip.com," https://play.google.com/store/apps/details?id=com.gogobot.gogodroid&hl=es, Google Play, Oct 2017. 2

[2] "Minube," https://play.google.com/store/apps/details?id=com.minube.app&hl=es, Google Play, Oct 2017. 2

[3] "Guides by lonely planet," https://play.google.com/store/apps/details?id=com.lonelyplanet.guides&hl=es, Google Play, Oct 2017. 2

[4] "Functional requirement," https://en.wikipedia.org/wiki/Functional_requirement, Wikipedia, Oct 2017. 3.1.1

[5] "Non-functional requirement," https://en.wikipedia.org/wiki/Non-functional_requirement, Wikipedia, Oct 2017. 3.1.2

[6] "Use case," https://en.wikipedia.org/wiki/Use_case, Wikipedia, Oct 2017. 3.2

[7] "Use case template and an example," http://pmblog.accompa.com/2009/10/08/use-case-template-example-requirements-management-basics/, Product Management Insights, Oct 2017. 3.2

[8] "Firebase official website," https://firebase.google.com/, Firebase, Oct 2017. 3.3

[9] "Firebase authentication," https://firebase.google.com/products/auth/, Firebase, Oct 2017. 3.3

[10] "Firebase realtime database," https://firebase.google.com/products/realtime-database/, Firebase, Oct 2017. 3.3, 3.4

[11] "Firebase cloud storage," https://firebase.google.com/products/storage/, Firebase, Oct 2017. 3.3

[12] "Structure your database," https://firebase.google.com/docs/database/android/structure-data, Firebase, Oct 2017. 3.4

[13] "User interface," https://en.wikipedia.org/wiki/User_interface, Wikipedia, Oct 2017. 3.5

[14] Levenshtein, Vladimir I. (February 1966). "Binary codes capable of correcting deletions, insertions, and reversals". Soviet Physics Doklady. 10 (8): 707—710. 4.6

[15] "Cipher," https://developer.android.com/reference/javax/crypto/Cipher.html, Android Developers, Oct 2017. 4.7

[16] “Que hay que hacer para cumplir con la lopd,” http://www.cuidatusdatos.com/infocomocumplirlopd.html, Oct 2017. 5.2

[17] “Ley orgánica 15/1999, de 13 de diciembre, de protección de datos de carácter personal,” https://www.boe.es/buscar/doc.php?id=BOE-A-1999-23750, Gobierno de España. 5.2