POLITECNICO
MILANO 1863

Where am I

Application Design Document

Luca Bianco, Luca Consonni

September 6, 2020

Where Am I, Design Document
Version v1.0, September 2020

Contents

1 Introduction

1.1 Abstract

In this section we want to introduce briefly our application and provide a full description of the design of *Where am I*.

The application is developed for the course of "Design and Implementation of Mobile Applications" at Politecnico di Milano. The goal of the course is to efficiently design and implement a mobile application. This documents illustrates the decisions we made in order to accomplish this goal.

1.2 Main Choices

1.2.1 Choice of the application

We had full choice on the purpose of the application.

After several proposals, our choice fell on a game, a kind of application that we use often in everyday life, with which we could then integrate our experience.

The game consists of being projected to a random location in the world (in a street view panorama) and the user must be able to orient himself through the surrounding environment and understand where he is. This game is inspired by an existing game in the web version called: Geoguessr. Our challenge is to create a more interesting and appealing mobile version for the average user, modifying the original version and adding valuable features.

1.2.2 Choice of the language

We decided to develop our application using React Native.

Main reasons:

- React Native is a new type of language increasingly used among application developers
- React Native is very convenient because it allows you to build an application for both android and iOS
- We both already knew the most common programming languages, so it was our interest to learn a new one

1.3 Timing

We begin to develop our application at the beginning of the second semester. Developing, testing and creating documentation took 6/7 months. Even the server side part was built and maintained during the same time span. We started development at February 2020 and complete it in July 2020. we decided to deliver in September to better review the work during the summer.

2 General overview

2.1 Idea

Where am I is a Game application that provides users the possibility to play alone or in a Real-Time multiplayer mode, challenging other user of the game or their friends. Each matches gives to the player a score, this score is entered in a ranking in which his score is compared with that of the other players, so as to determine the strongest players.

The main components of the project are:

- A server side part, composed by a noSql database that contains user profiles, games in real-time, and the application logic that is in charge of notify.
- A client side part, the application, which tasks are to query the database, show the gathered informations and provide users a point of interaction with the service.

2.2 Game Design

The structure of the game does not change whether it is played in single or multi player mode (online).

- A Game is divided into 5 rounds.
- A Round is divided into 2 main Phases: Moving Phase and Guess Phase.
- At the start of the round, in the Moving Phase, the application shows to the user a Street View Panorama (generated with random coordinates). If user is playing a multiplayer game, both the users will see the same Panorama.
- Moving across the Street View Panorama the user have to get an idea in which part of the world is found himself.
- Moving Phase lasts 60seconds. When he have an idea of the answer or the timer is expired, user go to the next phase of the round.
- In Guess Phase, the application will show to the user a world map. In this world map the user have to put a marker where he think was the panorama shown in the phase before.
- Based on the distance between the answer given and the solution, the application give to the user a score.
- When a user have his Round Score can go the next round. If the user is playing a multiplayer score, before go to the next panorama both the user have to give an answer.
- If a User not give an answer in Guess Phase, the application after 30seconds give for him a Default Answer.
- When an answer (answers in multiplayer mode) or Guess Phase timer expired the round is over.

2.3 Main Task

List of the core functionalities that can be found in our game. We decided to divide them considering his logical purpose, allowing the reader to easily understand them and where they could be found in the application.

- Authentication
  - Allows the user to authenticate using two different ways: a combination of email and password or Facebook account.
  - Choosing Email and Password method a user can be logged or register a new account using a combination of email and password and setting a unique username.
  - Choosing Facebook method, users are logged into the application with own Facebook credential. If it is the first access to application, user have to set a unique username.
  - If a user as already logged in the Game with his credential, these screens are automatically skipped.
- Introduction
  - At the first login of a new account the application briefly show to the user the main kind of game that the application provides.
  - User can consult the Introduction screens or skip them
- Home
  - Home is the core of application, by this point user can move through all the application and application features.
  - User can turn on and turn off the sound.
  - User can start a Single player game.
  - User can start a Multi-player game, a game where he challenges an other random user of the application.
  - User can visit his profile, where he can found all statistics, such his rank position or or total game played, and friends.
  - User can logout by the application.
- Moving Phase
  - User can explore street view Panorama.
  - User has 60seconds to get an idea of the answer and decide to go to Guess Phase.
  - When the timer expires user is directly sent to Guess Phase
- Guess Phase
  - User can move as we want across the world map.
  - User can insert a marker to indicate his answer.
  - User can insert a marker to indicate his answer.
  - When the timer expires user is directly sent to Guess Phase.
  - After send his answer user can see the solution on the same map of his marker.

  - After send his answer the Application show to the user how many kilometers are between his answer and the solution.
  - After send his answer the Application show to the user how many kilometers are between his answer and the solution.

- Profile
  - User can see his statistics such as: number of game played, number online games won, his average and his maximum score.
  - User can see his all badges and which he has unlocked. Badges are a reward system reward, unlockable by user experience on the application.
  - User can consult the rankings: Global Ranking and Friends Ranking.
  - User can send a friend request.
  - User can challenge a friend.
  - User can check his notifications and submit them.
  - User can change his profile picture.
  - User can visit the profile of the other users.

2.4 General qualities

Our application has several characteristics of accessibility and usability. The user interface is easy to use and intuitive, so that the user quickly learns what to do. The design is enthralling and captivating to guarantee the best experience possible.

- Usability: the main actor of the system is the end user. For this reason we decided to make the user interface as easy as possible, but still keeping all the functionalities needed to provide the best user experience.

2.5 Functional requirements

In this section we present the requirements necessary to the correct behavior of the system:

Authentication requirements

- The Application has to start with a Authentication activity.
- Authentication should be accessible only to users not currently logged in.
- Authentication should be skipped if user is currently logged in.
- Authentication should direct the user to Home if the authentication is successful

Email Authentication requirements

- Authentication should delegate registration/access using email to Email Authentication.
- Email Authentication should allow user to register, login in to application.
- Email Authentication should display fields for username, password and email, which might be hidden according to the operation the user is going to perform.
- Email Authentication should direct the user to Home if the authentication is successful

- Email Authentication should direct the user to Authentication if the authentication is successful

Facebook Authentication requirements

- Authentication should delegate registration/access using email to Email Authentication.
- Facebook Authentication should delegate registration/access using Facebook to the respective libraries.
- Facebook Authentication should direct the user to Home if the authentication is successful
- Facebook Authentication should direct the user to Authentication if the authentication is successful

Home requirements

- On first run, Home should introduce the application features to the user.
- On first run, Home should provide to start game music.
- Home should provide to the possibility to turn off and turn on the music.
- After the first run, Home should provide to start or not the music such as the user set it at the last login in.
- Home should provide to the user the possibility to start a Single-Player Game.
- Home should provide to the user the possibility to start an Online Game.
- Home should provide to the user the possibility to visit his profile.
- Home should provide to the user the possibility to logout of the application.

General requirements

- The application has to notify to the user when an other user want to be his friend.
- The application has to notify to the user when a friend when challenge him in an online game.
- The application has to notify to the user when his achieve a goal and unlock a new badge.

Single Player game requirements

- The game functions have to put on wait the user while the game features are loaded.
- The game functions have to provide to the user the possibility to come back on the homepage if the wait is too long.
- The game functions have to load correctly the round informations and create a street view panorama.
- In moving phase the game functions have to set a timer that identify a limit for this phase.
- In Moving phase the game functions should provide to the user the possibility to moving through the street view panorama.

- In Moving phase the game functions should provide to the user the possibility to go to the next phase before the timer expired.
- In Moving phase the game functions should provide to the user the possibility to go to the next phase before the timer expired.
- In Moving phase the game functions have to redirect the user to the next phase when timer expired.
- In Guess phase the game functions have to set a timer that identify a limit for this phase.
- In Guess phase the game functions have to set a default Marker.
- In Guess phase the game functions should provide to the user the possibility to zoom and move on the world map.
- In Guess phase the game functions should provide to the user the possibility to move a the marker on the world map.
- In Guess phase the game functions should provide to the user the possibility to move a the marker on the world map.
- In Guess phase the game functions should provide to the user the possibility to guess the place before the timer expired.
- In Guess phase the game functions have to submit the last marker on the map when the timer expired.
- In Guess phase, when the round is over, the game functions have to show to the user his score.
- In Guess phase, when the round is over, the game functions have to show to the user his score.
- In Guess phase, when the round is over, the game functions should provide to the user the possibility to watch on the world map his answer and the real solution.
- In Guess phase, when the round is over and it is less than five, the game functions should provide to the user the possibility to go to the next round.
- In Guess phase, when the round is over and it was the fifth, the game functions should provide to the user the possibility to finish the game and redirect to home.
- When a game is over game functions have to update user's statistics.

Multiplayer requirements

- Multiplayer requirements include Single Player requirements.
- In Guess phase, when the round is over and it was less than five, the game functions should provide to the user the possibility to get ready for the next round.
- In Guess phase, when both player are ready for the next round , the game functions have to set a timer for the next round start. When the timer expired game functions redirect players to the next round.
- In Guess phase, when both player are ready for the next round , the game functions have to show to both the real time leaderboard of the game.

- In Guess phase, when both player are ready for the next round , the game functions provide to the user the possibility to be redirect to the home page.

Profile requirements

- Profile functions have to show loading page while user statistics are loaded.
- Profile functions should provide to the user the possibility to watch his profile picture.
- Profile functions should provide to the user the possibility to watch what are the badges available on the Game.
- Profile functions should provide to the user the possibility to check his badges.
- Profile functions should provide to the user the possibility to change his profile picture.
- Profile functions should provide to the user the possibility to check the global ranking.
- Profile functions should provide to the user the possibility to check the ranking filtered by his friend.
- Profile functions should provide to the user the possibility to watch others user's profile through the player on the rankings.
- Profile functions should provide to the user the possibility to watch his number of game.
- Profile functions should provide to the user the possibility to watch his number of game won.
- Profile functions should provide to the user the possibility to watch his average score.
- Profile functions should provide to the user the possibility to watch his average score.
- Profile functions should provide to the user the possibility to watch his maximum score ever.
- Profile functions should provide to the user the possibility to watch his friend list ever.
- Profile functions should provide to the user the possibility to add a new friend list ever.
- Profile functions should provide to the user the possibility to watch if a friend his online.
- Profile functions should provide to the user the possibility to challenge if a friend his online.
- Profile functions should provide to the user the possibility to check his friend request.

2.6 Non-functional requirements

These are the non-functional requirements that are necessary to guarantee a functional application.

- **Portability**: we are tested and optimize the application for the android smartphone, but having developed the application with React-Native, we are pretty sure we can extend easily and without much loss time the application on iOS and tablet in the future.
- **Stability**: the system should always be available to be used at any time it is needed. System failures and server side crashes must be avoided in order to guarantee this requirement.
- **Availability**: the services must always be up and running even during a failure period. An administrator must restore the service as soon as possible. A backup facility must be present in order to do so.
- **Reliability**: the data must be reliable, meaning that they are trustworthy and not corrupted by any other. So the remote database must be protected and secure.
- **Efficiency**: the application needs to use as less resource as possible. Algorithms and data structures have been developed to optimize the consumption of resources in the best way possible.
- **Extensibility**: the game was programmed with the idea that further extensions could be added in a simple way without deeply modifying the code, like personalize game with only some part of the world, possibility to play in more than two, and other features.
- **Maintainability**: code is easily readable, in this way it is maintainable by future programmers.

  Like we will see in the next chapters Stability, Avaibility, Reliability and Efficiency are support by Google platform integration, that could be a warranty.

3 System Architectures

3.1 Client-server architecture

Where Am i application is structured like a client-server application but not the traditional one, in fact the game needs data synchronized in real time and with a traditional server client-server application this process is very hard and require very complex and expensive method. So we decied to use Firebase and Google Cloud Platform, cloud platform that allows to build an application without managing the infrastructure.

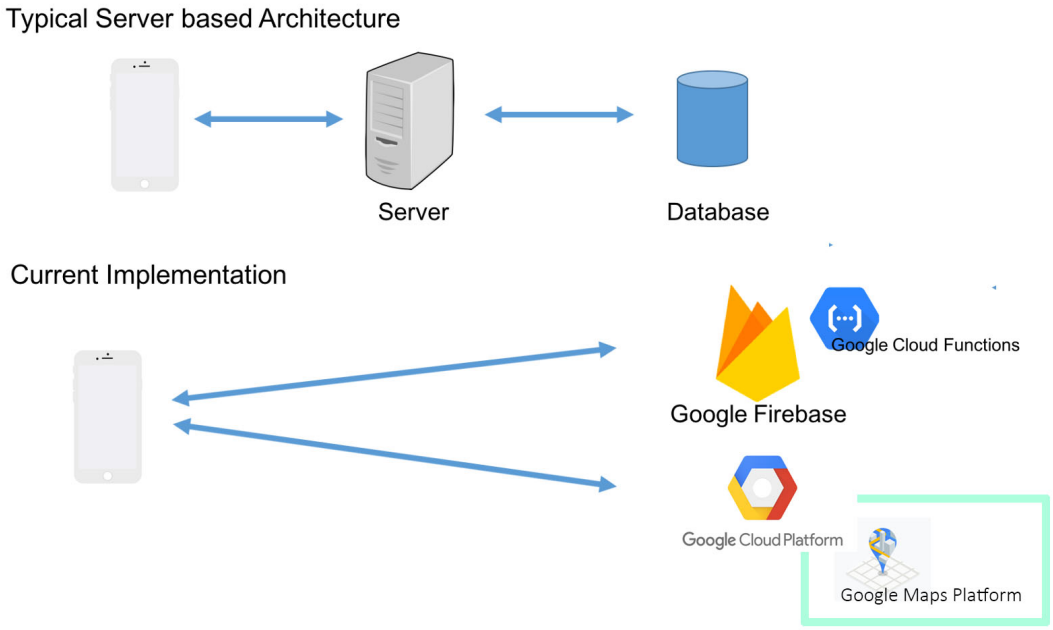


Figure 3.1: Firebase Architecture

3.1.1 Client-side application

Client-side application is written using react-native. It runs in a background process (which interprets the JavaScript written by the developers) directly on the end-device and communicates with the native platform via a serialisation, asynchronous and batched Bridge. React components wrap existing native code and interact with native APIs via React's declarative UI paradigm and JavaScript. This enables native app development for whole new teams of developers, and can let existing native teams work much faster. React Native messages from the JavaScript thread are used to manipulate native views. React Native also allows developers to write native code in languages such as Java for Android and Objective-C or Swift for iOS which make it even more flexible.

3.1.2 Server-side application. Firebase

Firebase is a fully managed platform for building iOS, Android, and web apps that provides automatic data synchronization, authentication services, messaging, file storage, analytics, and more. Starting with Firebase is an efficient way to build or prototype mobile backend services.

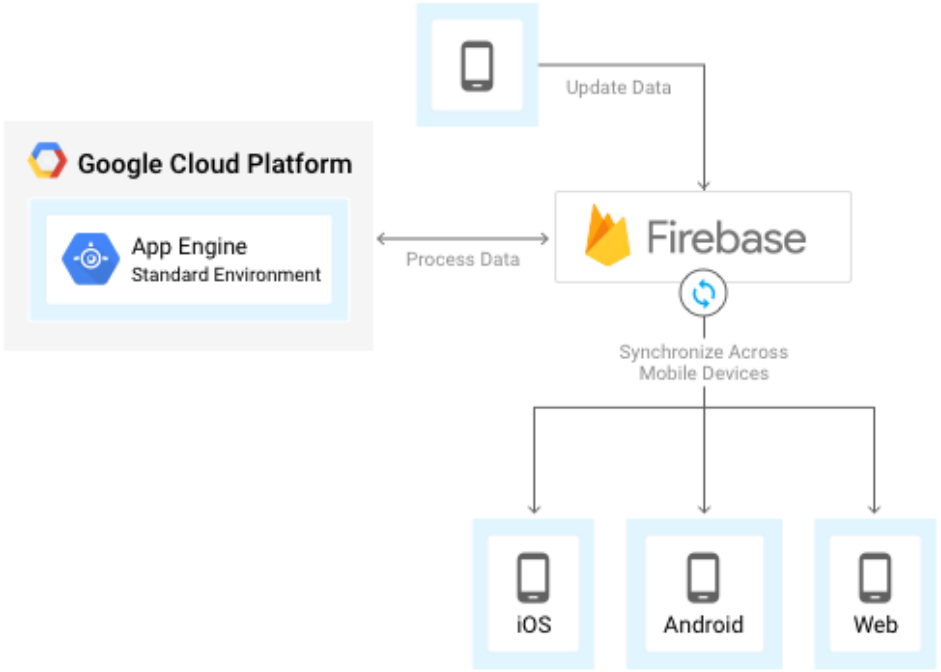


Figure 3.2: Firebase Architecture

App Engine standard environment is an app platform that monitors, updates, and scales the hosting environment; all you need to do is write your mobile backend service code. If your app needs to process user data or orchestrate events, extending Firebase with App Engine standard environment gives you the benefit of automatic real-time data synchronization.

4 External services

We have used several external services for our application, some of them are necessary for the proper behavior of the system, such as, Google Firebase to implement real time database system.

The main advantage in using these kind of services is first of all the ease to not to have to implement specific portion of code in order to achieve the same result that these external services already provide.

Now we present the main external services that we have used and integrated in our application.

4.1 Google Firebase[1]

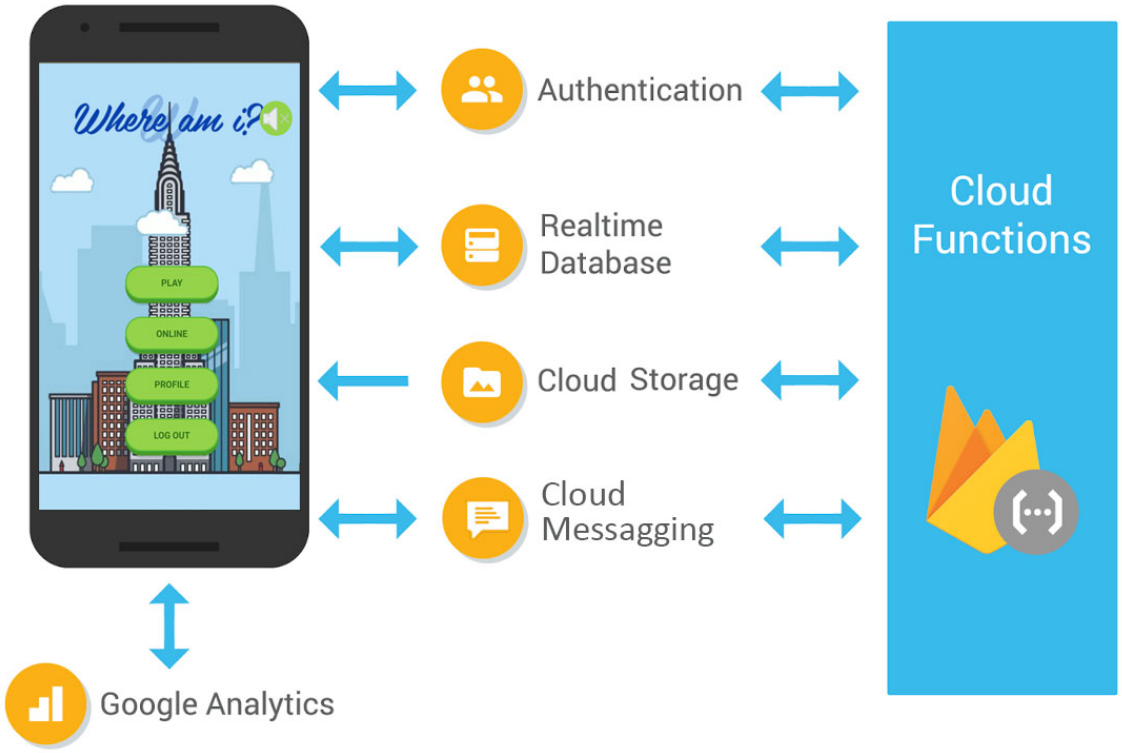


Figure 4.1: Firebase services

[1]Google Firebase, https://firebase.google.com/

4.1.1 Authentication

This service is of foundamental importance since it handles all the authentication procedures for the users. In the client-side part of the application, it is used to allow user to register or login using their preferred method among Facebook or classic email and password. In the server-side of the application, it is used to verify that the user token received in the header of a request are of a valid user. It has been easily integrated in both part of the application.

4.1.2 Realtime database

Store and sync data with our NoSQL cloud database. Data is synced across all clients in realtime, and remains available when your app goes offline. Data is stored as JSON and synchronized in realtime to every connected client. When you build cross-platform apps with our iOS, Android, and JavaScript SDKs, all of your clients share one Realtime Database instance and automatically receive updates with the newest data.

4.1.3 Cloude storage

Cloud Storage is built for app developers who need to store and serve user-generated content, such as photos or videos. Cloud Storage for Firebase is a powerful, simple, and cost-effective object storage service built for Google scale. The Firebase SDKs for Cloud Storage add Google security to file uploads and downloads for your Firebase apps, regardless of network quality. You can use our SDKs to store images, audio, video, or other user-generated content

4.1.4 Cloude Messagging

This service handles the sending of messages from the server to the client. These messages are triggered when a friend request is created on the database. When a friend request is created a cloud function send the message to the corresponding user using the token of the device that is created and updated every time a user access to the app, since a user could change the phone.

4.2 Google Maps Platform[2]

4.2.1 Street View

The Moving Phase through the street view panorama is the central part of the game. So to build it we did it by relying on the services of Google, a leading company in the sector whose platform makes everything easy and accessible. Google Street View Embed street view imagery and high resolution satellite imagery. Obviously the service does not cover the whole world, but it is by far the largest on the market and allows to 'catapult' the user to a different place each time, making the game very beautiful and exciting. The service is the best, 'going around' through the streets is really easy and intuitive, making the experience incredible for the player.

4.2.2 Maps

Maps it is an other real important service, in fact is the way that permit to built the other phase of the game: Guess Phase. Google maps is the best mapping service, it cover 99% of the world and it is constantly updated. Maps is very simple to use for the developer, it

[2]Google Maps Platform, https://cloud.google.com/maps-platform/maps

can easily display maps as images or interactive maps, permit to style them with custom markers, lines, colors, polygons, and images to align with your brand. We think it is the best choice for our game, because in Guess Phase user can easily insert his marker and fast check on the map the result.

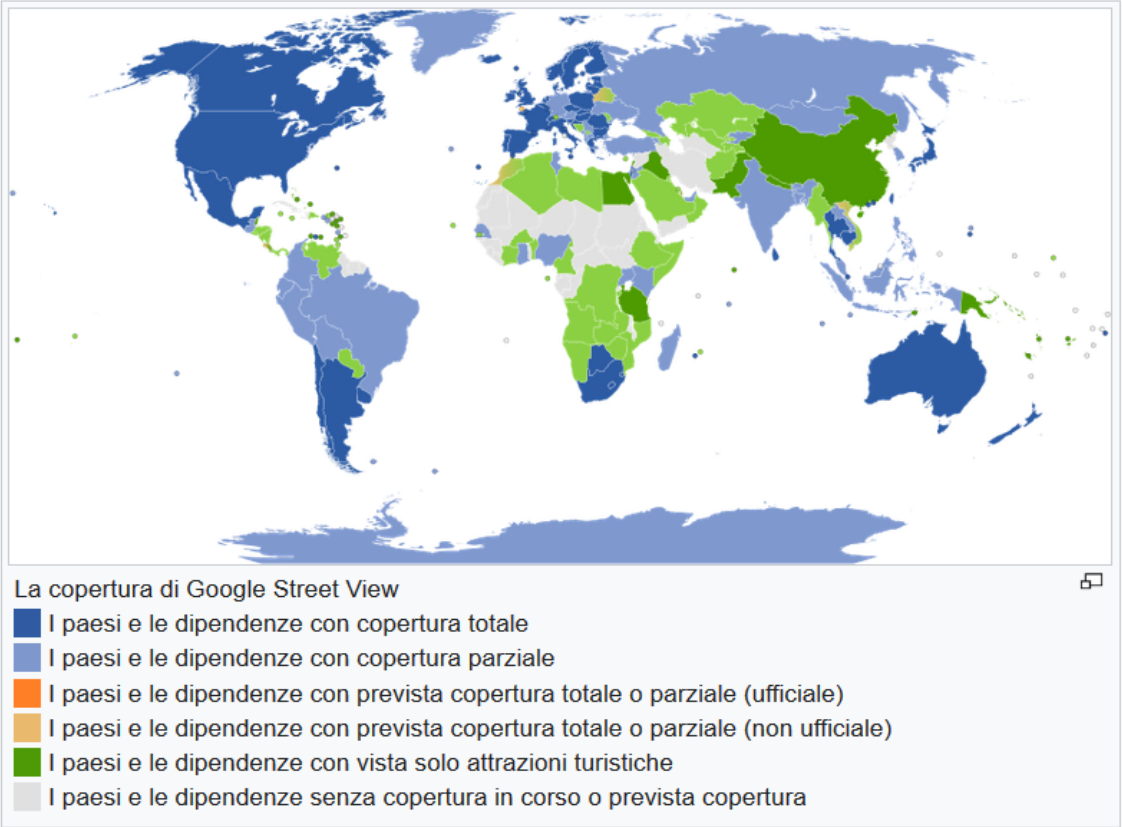


Figure 4.2: Streat View coverage

4.3 Keene Map Polygon/Polyline tool[3]

Each round of a game is generated by two random coordinates. This random coordinates can be each pair of coordinates in the world, so to prevent that the coordinates generated are oceanic coordinates or simply coordinates not mapping by Google service (figure 4.2), we decide to map the world in some polygons. This polygons can contain continents, islands, states or whatever as long as the coordinates content inside the polygon are mapped by google services. To do that we use an online open source tool called: Map Polygon/Polyline Tool developed by Keene state college in USA. This tool permits to developer to draw a polygon on the world map and retrieve the polygon coordinates like a JSON.

[3]Keene Map Polygon/Polyline tool, https://www.keene.edu/campus/maps/tool/

4.4 Facebook[4]

4.4.1 Authentication

The service is used to obtain the Facebook user profile and email, that is then used by Firebase Authentication to handle new user creation or authentication.

[4]Facebook, https://developers.facebook.com/

5 Data design & implementation

As said in last chapter the most quantity of data are stored in Real Time Database of Firebase. Real Time Data Base store and sync data with our NoSQL but using JSON.

5.1 Real Time database

Below are represented the JSONs model in database and their description. Functions work like Triggers, in a sql DB, they are activated when a specific event occurs and they modify the structure of others JSON.

- **users**: This JSON includes users of the application.

  unique string of a user: user has a JSON children, named with a unique string, use to identify each user.

  1. **username**: username chosen by the user, it is a string.
  2. **userpic**: url of image pic, it is a string.
  3. **token**: it is a random unique string.
  4. **online**: boolean that describe if the user is online or not.
  5. **last_access**: date type, the date of the last access of the user.
  6. **statistics**: JSON with some statistics of the user.
     a) **avgScore**: average score between all user game, it is a positive integer.
     b) **day_in_row**: number of games played in a row, it is a positive integer.
     c) **game_in_day**: number of games played by a player in a day, it is a positive integer.
     d) **max_score**: maximum score reach in a game by an user, it is a positive integer.
     e) **nGames**: number of total games played, it is a positive integer.
     f) **nGames_multi**: number of games played in a multi-player mode, it is a positive integer.
     g) **nGames_sing**: number of games played in a single-player mode, it is a positive integer.
     h) **win**: number of games won, it is a positive integer.
     i) **win_in_row**: number of games won in a row, it is a positive integer.
     j) **quit**: number of games quit without come to the end, it is a positive integer.
     k) **badge**: JSON that described badges unlocked.
        i. **center**: boolean, true if the user has unlocked this badge.
        ii. **doppelganger**: boolean, true if the user has unlocked this badge.
        iii. **extrovert**: boolean, true if the user has unlocked this badge.
        iv. **fire**: boolean, true if the user has unlocked this badge.

v. **gamer**: boolean, true if the user has unlocked this badge.

vi. **time**: boolean, true if the user has unlocked this badge.

7. **friend**: Json with a list of the friends of the user.
   a) *unique string of a user*: unique string that identify a user.
      i. **img**: url of friend's image pic, it is a string.
      ii. **name**: nickname of the friend.
8. **request**: Json with a list of the friends request that user are not submit yet.
   a) *unique string of a user*: unique string that identify a user that sent friend request.
      i. **img**: url of friend's image pic, it is a string.
      ii. **name**: nickname of the friend.
9. **playRequest**: Json with a list of the friends request to play together.
   a) *unique string of a user*: unique string that identify a user that sent play request.
      i. **img**: url of friend's image pic, it is a string.
      ii. **name**: nickname of the friend.

- **games**: JSON which contents all game are played in real-time. His children can have some difference if they are single-player game or multi-player game.

  unique string of a user / unique string of the players placed one after the other: this value can change if the game is a single-player or a multi-player game.

  1. **date**: date that the game is started, data format.
  2. **finished**: boolean, true if the game is finished.
  3. **type**: it is a string, could get value "single" or "multiplayer" according to the kind of game.
  4. **playerN**: JSON with the significant values of player N. (N is used in this report to not be redundant, there are 2 children built in the same way, n take value 1 or 2. In a single-player mode, player2 is not initialized).
     a) **score**: it is a positive integer, it is the real time score of the player.
     b) **user**: it is a string, it is the unique string of the user.
     c) **username**: it is a string, it is the nickname of the user.
     d) **badge**: JSON that described badges unlocked during this game by player 1.
        i. **center**: boolean, true if the user has unlocked this badge.
  5. **roundCoordinates**: JSON with the coordinates of the games
     a) **round_n**: JSON with the significant values of round n. (n is used in this report to not be redundant, there are 5 children built in the same way, n take value from 1 to 5).
        i. **latitude**: it is a float, it indicates the latitude used in this round of the game. It can be positive or negative.
        ii. **longitude**: it is float, true it indicates the longitude used in this round of the game. It can be positive or negative.

iii. **playerN_answer**: JSON that describe the answer that the player submit this round (N is used in this report to not be redundant, there are 2 children built in the same way, n take value 1 or 2. In a Single-player mode this node is not initialized).

A. **ready**: it is a boolean, if it is true: it indicates that the user is ready to go to the next round

B. **score**: it is a positive integer, it is the real time score totalize by the player in this round.

C. **coordinates**: it is simply JSON with longitude and latitude insert by the player in this round, in this report to not be redundant we will not expand this JSON.

- **waitingRoom**: JSON which contains the players that are waiting to matchmaking with an other player, in a way to start together the same game.

  unique string of a user: JSON children, named with a unique string, use to identify each user.

  1. **user**: unique string, use to identify each user, it is a string.
  2. **username**: username chosen by the user, it is a string.
  3. **userpic**: url of image pic, it is a string.

Figure 5.1: Extract of Real Time database, one multi-player game

5.2 Firebase functions

Below is briefly described the Firebase Functions that manage the real time database. Functions work like Triggers, in a sql, DB, they are activated when a specific event occurs and they modify the structure of the JSONs.

- **createRandomGame**: this function is triggered when is created a new child of waitingRoom. The function control the number of child in waitingRoom and when the number is more then 2, in pairs, delete children from waitingRoom and it will create a new child of games with the data of user deleted from waitingRoom.
- **addCoordinate**: this function is triggered when is created a new child of games. The function will create the child roundCoordinates (Json number 5 of games) of the game just created.
- **sendFriendRequest**: this function is triggered when a player sent a friend request. This functions create in database the child "request" (number 8) in users.
- **badgeGamer**: this function is triggered on update of "online" (number 4 of users JSON), the function compare some variables such as day_in_row and last_access with standard, and then it could be update "gamer" in badge (k in JSON user).
- **badgeQuit**: this function update count the number of games that a player quit without finish it. Any time a player quit a game (a child of games is deleted and value of user is offline) that is not finished yet (variable 2 of games "finished" set on false), the function will update quit (variable j of statistics in users). So when this count reach a certain value, the function will notify badge quitter or real quitter to the user.
- **badgeExtrovert**: this function is trigger when a new friend is added to a user's friend list (child 7 of users). Function keep the count of the friends of the user each time a new friend is added, so when the friend list reach a certain value, the function will send a user a Extrovert Badge.
- **updateImgProfile**: This function is trigger when userpic (value number 2 in users) is updated. Function will notify Doppleganger Badge to user and will update user's picture in the friend list of the user (child 7 of users).

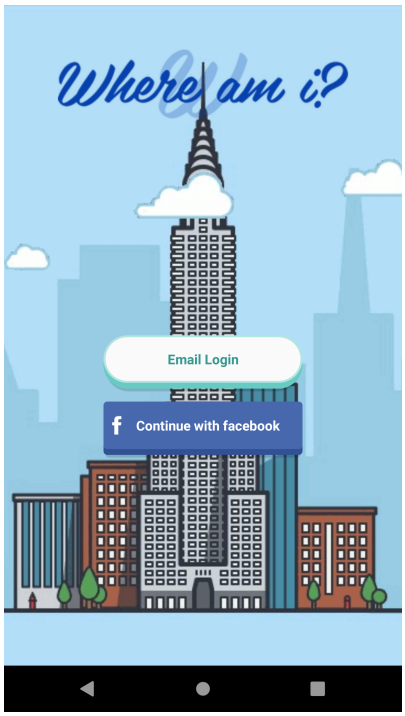
5.3 Media

Where am I application does not require that many user media be saved to a database. The only media the user can upload is their profile photo. This data is saved and managed using google firebase cloud storage.

6 User interfaces

In this section are showed the most important screenshots of the application. With them is provide a briefly description of the screen: which are his main features and how is linked with the other pages.

6.1 Authentication



Authentication page is very simple and just shows the two methods to get access to the application: Email and Facebook. By clicking on "Continue with Facebook", after having granted the permissions requested by Facebook, the user will be directed directly to the application Homepage. If is the first login ever, the application requires to insert a unique nickname.

Figure 6.1: The authentication screen

By clicking on "Login with email", a new page will open, the user will be able to log in with his credentials (email and password) with login button, or, if he is a new user and don't have an account with "Not already register? Sing Up!" he will redirect to Sign Up page. Sign Up page is basically the same page like this but with an email and a password it's required to the user to insert a unique nickname.

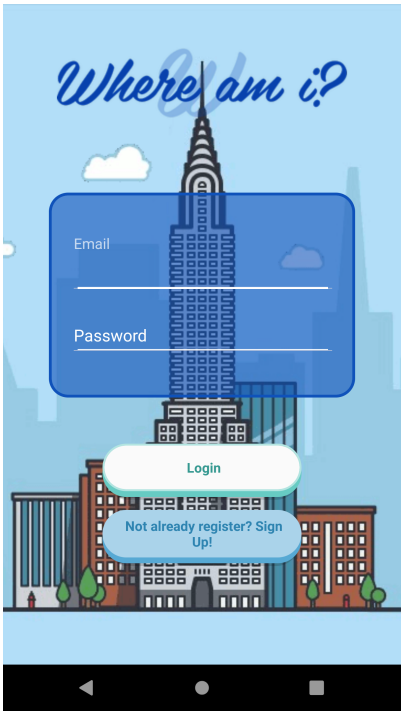


Figure 6.2: The email authentication screen

6.2 Home

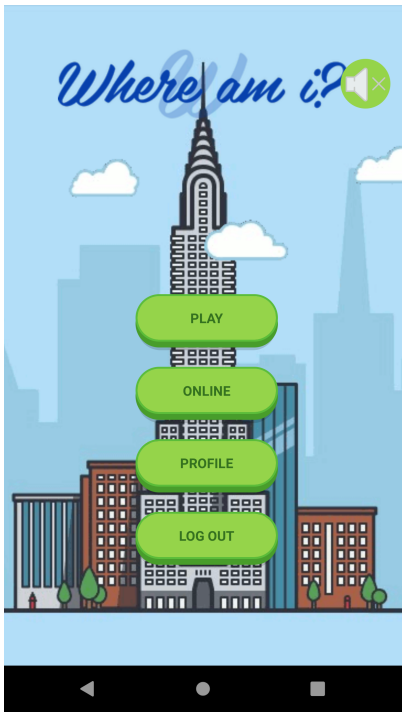


Figure 6.3: The home screen

Home is the central point of the application, from here it is possible to move through the entire application via a menu of buttons. The first two buttons are the core of the application: the game modes. "Play" button allows you to quickly play the single player mode. "Online" instead starts the multiplayer mode. Trivially, "Profile" button leads to the page where manage the profile, while "Logout" allows you to log out of the application. At the top right there is the button to activate and deactivate the volume.

6.3 Game screens

This view is simply compose by only three components: a timer, a button and the Street view panorama. The Street view panorama is obviously the main actor, it is random generated by the application, in way to "Transport" the Player in a different spot of the world each round of the game. The user can use double tap on the street or can use the arrow to move himself across the streets. It's goal is to try to understand in which part of the world he is and to do that he has 60seconds. The green timer at the top right indicates the time available. If a user think to have the answer before the timer expired, he can push top left button "Give Answer" to go to the next phase.

Figure 6.4: Moving Phase

Figure 6.5: Guess Phase

Guess Phase page has basically the same structure of the Moving Phase one, in way to render more comfortable and intuitive the user experience, but instead of have a street view panorama we have a world map. In this phase the user have 30 seconds, always shown by the timer on the top right, to put a marker (his answer) on the map. User can change his mind and the marker will change position each time the user click on the map. The user can confirm his answer before the timer expired clicking on the button "Make The Guess".

After that the user have submit his answer, a red marker corresponding to the Real Solution will appear on the map. A modal will show to the user how many kilometers there are between his answer and the real one and his so his score. In this case if the user is playing a single-player mode, he will directly resend after a few second to the next round (to a new Moving phase) or to home if is finished the round 5. If the user is playing in a multiplayer mode, the user have to wait that hiso pponent insert the answer too. Like in figure 5.6 there will be 3 marker on the map, player one (yellow), player two (blue) and the solution (red).

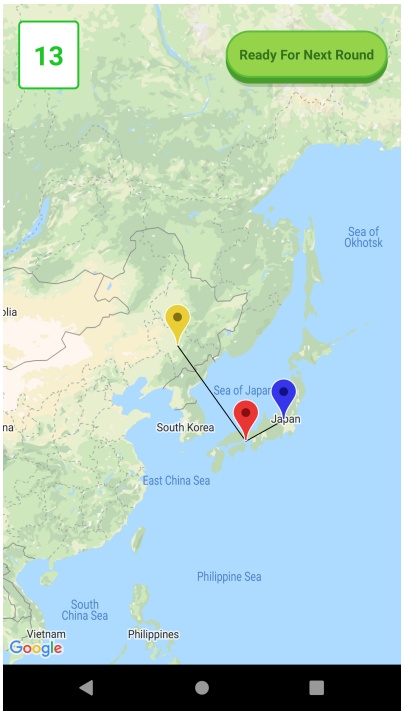


Figure 6.6: Guess Phase

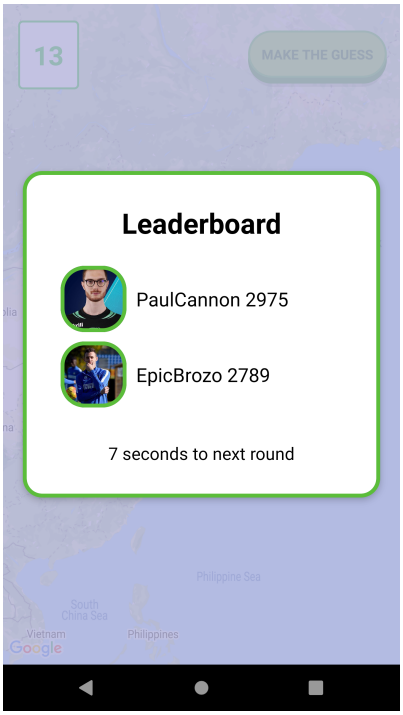


Figure 6.7: Guess Phase

In a multiplayer game both users have to confirm that are ready for the next round, pushing "Ready for the next Round" button in top right. When both have confirmed application will show to the user the leaderboard, and after few seconds will redirect the user to a next round o to the home page if game is finished.

6.4 Profile

The profile screen collects all the statistics and information of the user. The page is divided into three parts. In the first one, we find the user's profile photo, his name and 5 clickable icons, each with a different purpose. The arrow at the top left takes you back to the home page, the pencil allows you to change the profile picture, the bell shows you notifications, the cup allows you to see the rankings and the people icon sends you to friends list.T he central section shows the user statistics, how many games he has played, how many he has won, what is his average score and his maximum score in a game. The last section the are Badges: reward icon that user can unlock playing the game. Badges locked are opaque, while unlocked are full colorized. User can scroll this section and click on each on to see what is it.

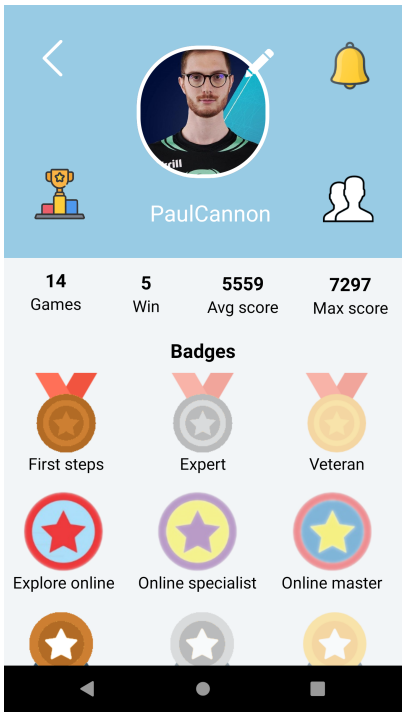


Figure 6.8: Profile screen

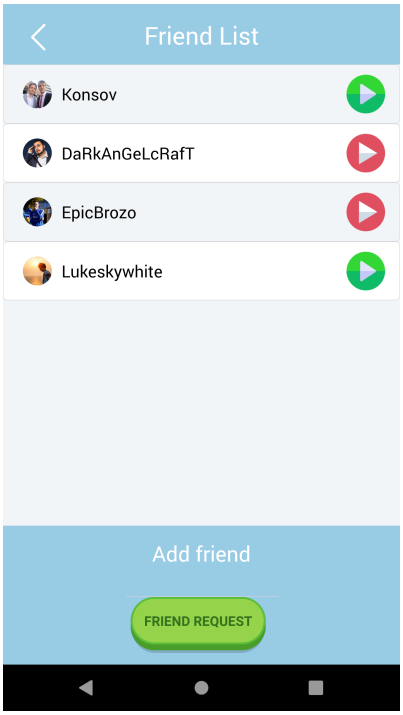


Figure 6.9: Friend list

Friend List is divided in 3 section. The first one with the title of the page and with an arrow allowing to the user to come back to profile. The second section is a scrollable list of friends. For each friend his reported his profile picture, his name and a "play" icon. If the icon is green means the user is online and you can invite a friend to play together with you, if the icon is red means the friend isn't online and you can't challenge him. Pushing on the line of a Friend, application will resend the user to the friend's profile. In the last is composed by a text box and a button, if the user want add a friend to his list, he have to complete text box with friend nickname and send him a friend request pushing the button.

Leaderboard screen is divided in two section. First section is very similar to profile first section, in this the application provide continuity and maintain more linear and comfortable the user experience. On the left of the personal image there is the ranking of the user, while on the right there is his average score, the score that count for the ranking. In this section there is a double button, click on it the user can choose if the application have to show the global ranking or a leaderboard limited to him and his friends. Down this section there is the leaderboard. All users of the game are ranked by their personal average score. By Clicking each line a user can consult the profile of the other players.

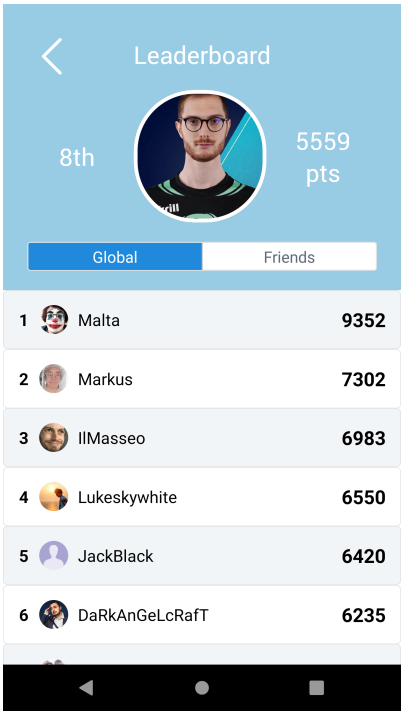


Figure 6.10: Global Ranking

7 UML diagrams

In this section we present some useful diagrams that helps the reader to better understand the interaction between the user and the application. Here we present some of them.

We decided to include only a examples of two diagrams, in order to make the document not too complex and less readable.

7.1 Use Case diagrams

These diagrams show the flow of operation triggered by a specific actor when it tries to perform some task.

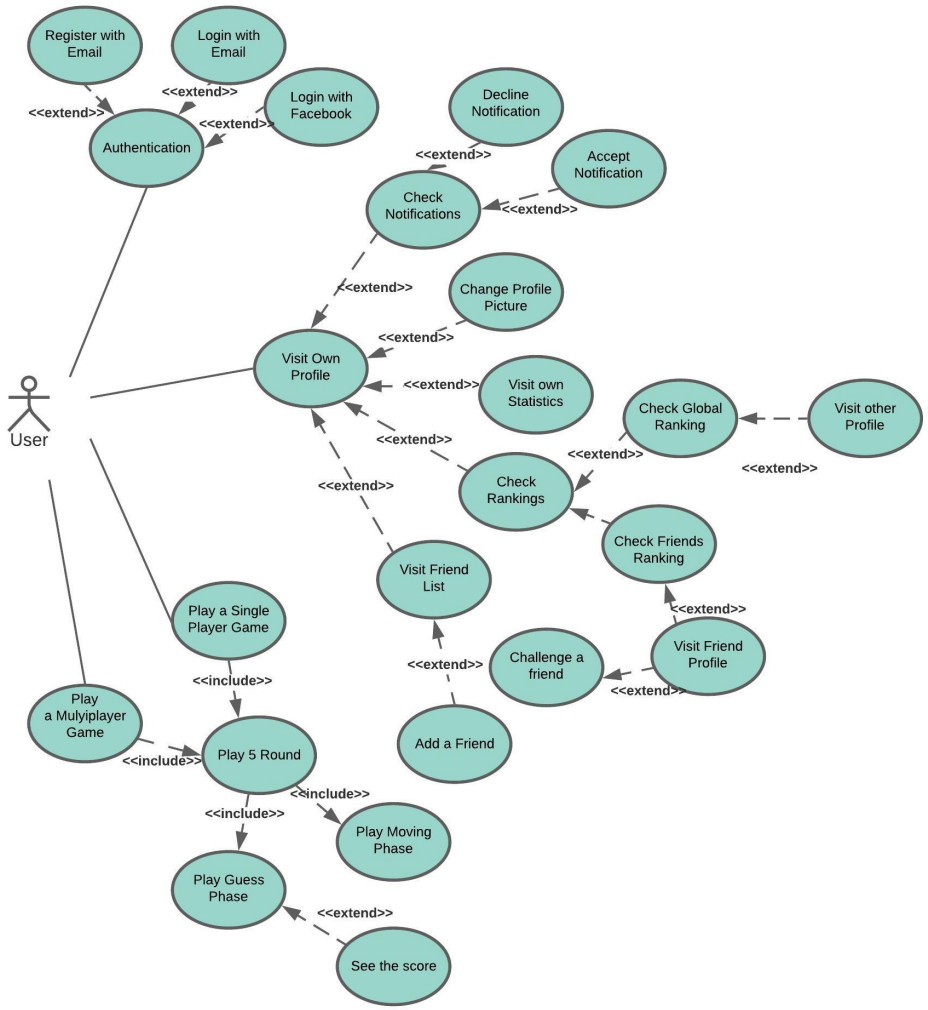


Figure 7.1: Use case

7.2 Sequence diagrams

In this section we provide a sequence diagram concerning the flow of the logical operations performed by our application once a certain operation is executed.

This sequence diagram shows the interactions with pages an actors involved where an online game is started. Player is redirect to play screen after that the server system find an opponent. Each round when the player insert his marker answer the application forwards the response to the database.

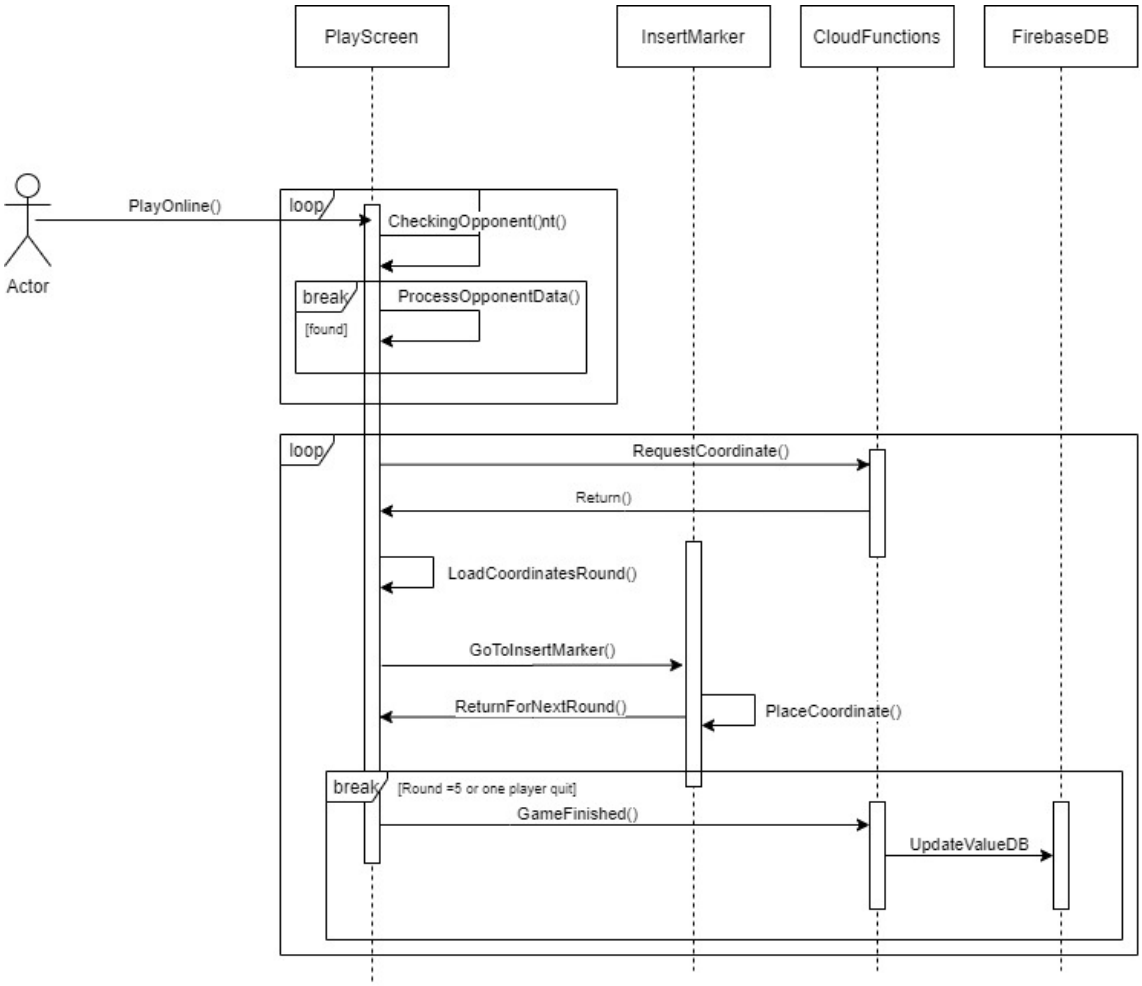


Figure 7.2: Displaying events by area sequence diagram

8 Testing

8.1 Test case

This section describes the results of the main tests done on Where am I application.

Test Case	**Login using Email and Password**
Goal	Login using Email and Password
Input	Click on "Login" in Email Email Sing In Screen
Expected outcome	The user is logged in
Actual outcome	CORRECT: if the user inserts a valid email and password pair, after clicking on the "Login in" button he is authenticated. A network connection is required to perform the operation.

Test Case	**Login using Facebook**
Goal	Login using Facebook
Input	Click on Facebook button in Login Method Screen
Expected outcome	The user is logged in
Actual outcome	CORRECT: after clicking on the button, the user is asked to access using his Facebook account and to grant the application permission to read profile information. If the user accepts, he is successfully authenticated in Where am I. A network connection is required to perform the operation.

Test Case	**Logout**
Goal	Logout from the application
Input	Push "Logout" button in "Home Screen"
Expected outcome	The user is logged out
Actual outcome	CORRECT: The user is correctly logout of the application and redirct to the Login Method Screen. If he want turn to play again he have to log in again.

Test Case	**Start a Single-Player Game**
Goal	Start A Game in Single-Player Mode
Input	Push "Play" button in "Home Screen"
Expected outcome	The user have to "transport" in a Street View Panorama
Actual outcome	CORRECT: After A loading page on the user screen is loaded a Street view Panorama and the game is started. The timer in top of the left is running and the "Give Answer" Button is loaded correctly at top right.

Test Case	**Start a Multi-Player Game**
Goal	Start A Game in Multiplayer Mode
Input	Push "Online" button in "Home Screen"
Expected outcome	The user have to "transport" in a Street View Panorama
Actual outcome	CORRECT: After A loading page on the user screen is loaded a Street view Panorama and the game is started. The timer in top of the left is running and the "Give Answer" Button is loaded correctly at top right.

Test Case	**Go to Guess Phase (1)**
Goal	User choose to go to the Guess Phase
Input	Push "Give Answer" button in "Play screen"
Expected outcome	The user is redirect to world map and can start the Guess Phase
Actual outcome	CORRECT: After pushing "Make Guess" button the user is redirect to "Insert Marker Screen". Guess page is correctly loaded: user marker appears on the world map timer in top left is start and there is "Make Guess" button in top right.

Test Case	**Go to Guess Phase (2)**
Goal	Go to guess phase by timer expiring
Input	User let the Moving Phase's timer expires
Expected outcome	The user is redirect to world map and can start the Guess Phase
Actual outcome	CORRECT: After pushing "Make Guess" button the user is redirect to "Insert Marker Screen". Guess page is correctly loaded: user marker appears on the world map, timer in top left is start and there is "Make Guess" button in top right.

Test Case	**Move Marker multiple times on the map**
Goal	User can move his marker more times on the map
Input	User press on the map where he wants the marker
Expected outcome	The user is redirect to world map and can start the Guess Phase
Actual outcome	CORRECT: After clicking on the map, user marker leave his actual position and it is moved on the place pressed

Test Case	**Guess Place and Finish Round (1)**
Goal	Submit answer, then consult score and compare answer with the right one.
Input	Push "Make Guess" button in "Play screen"
Expected outcome	A modal show to the user his score, and on the world map the user can compare his answer with the right one.
Actual outcome	CORRECT: After pushing "Make Guess" button, the application show to the user a modal with his score and how many kilometers he go far from the right solution. Closing the modal the user can zoom and moving on the map. On the map are inserted his marker and the right solution marker, linked by a black line. The button "Make Guess" is become "Ready for the next round" (test developed with round 4 or less)

Test Case	**Guess Place and Finish Round (2)**
Goal	Submit his answer, then consult score and compare answer with the right one.
Input	User let the Guess Phase's timer expires
Expected outcome	A modal show to the user his score, and on the world map the user can compare his answer with the right one.
Actual outcome	CORRECT: After timer expiring, the application show to the user a modal with his score and how many kilometers he go far from the right solution. Closing the modal the user can zoom and moving on the map. On the map are inserted his marker and the right solution marker, linked by a black line. The button "Make Guess" button is become "Ready for the next round" button. This test is running with round 4 or less, if was the round 5 we expect that the button would become "End" Button.

Test Case	**Go to the next Round Single-Player Mode**
Goal	User have to start a new Moving Phase, if he has played four round or less
Input	Push "Ready for the next round"
Expected outcome	The user have to "transport" in a Street View Panorama different by each one of the round before.
Actual outcome	CORRECT: After pushing "Ready for the next turn" button, the application redirect the user to a new Moving Phase. A new Street View Panorama is loaded correctly and it is different by the one before, the timer in top left is running and in top right there is "Give Answer" Button.

Test Case	**Go to the next Round Multiplayer Mode**
Goal	User have to start a new Moving Phase, if he has played four round or less
Input	Push "Ready for the next round"
Expected outcome	The user have to "transport" in a Street View Panorama different by each one of the round before.
Actual outcome	CORRECT: After both of user pushing "Ready for the next turn" button, the application show a modal to the user. On this modal there is the leaderboard of the game and a timer. The timer allows to synchronize the game between the two player. When the timer expired the user is redirected to a new Moving Phase. A new Street View Panorama is loaded correctly and it is different by the one before, the timer in top left is running and in top right there is "Give Answer" Button.

Test Case	**Finish Game single-player mode**
Goal	After the round 5 the game have to be finished, no new round have to be started
Input	Push "End Game"
Expected outcome	User have to redirect to the "home screen"
Actual outcome	CORRECT: After pushing "End" on the application appear a loading screen to allows to application to send the data of the game to the server. When the loading is over the user is correctly redirected to "Home Screen".

Test Case	**Finish Game Multi-player mode**
Goal	After the round 5 the game have to be finished, no new round have to be started
Input	Push "Go to Hub" button on the modal of the last round
Expected outcome	User have to redirect to the "home screen"
Actual outcome	CORRECT: After the 5 round. The leaderboard not load a timer but a "Go to Hub" button. Pushing this button on the application appear a loading screen to allows to application to send the data of the game to the server. When the loading is over the user is correctly redirected to "Home Screen".

Test Case	**Visit own user Profile**
Goal	Visit his own profile
Input	Push "Profile Button" in "Home Screen"
Expected outcome	User have to redirect to the "Player Profile Screen"
Actual outcome	CORRECT: After that user pushing on "Profile Button" , profile screen is correctly loaded. User can find on the top of the screen a several button icon to improve his social experience. In the middle profile screen loaded correctly his game statistics and on the button the badge section is completely loaded: Badges locked are opaque while badges unlocked are fully colorized.

Test Case	**Visit Global Ranking**
Goal	Vist Global Ranking of all Players of Where am I
Input	Push trophies icon in "Player Profile Screen"
Expected outcome	User have to redirect to "LeaderScreen" and check his position on the global ranking.
Actual outcome	CORRECT: After pushing on Trophies Icon, user have to wait a few seconds that the application load data and then he is correctly redirected to "Leader Screen". The page show to the user his position on the ranking and the position of the other players of the server. Above the leaderboard there is a double bottom, the first part of the bottom, called "Global", is colored in cyan, that means the user his watching the Global Ranking.

Test Case	**Visit Friends Ranking**
Goal	Visit the ranking of the game only between the user and his friends
Input	Press "Friends" in "Leaderboard Screen"
Expected outcome	Global ranking has to change and become a leaderboard where are ranked only the user and his friend
Actual outcome	CORRECT: After pushing on "Friends", the second part of the button switch to cyan and the first part to grey. In the same way laederboard instantly changes, showing ranked in order by score only the user and his friend

Test Case	**Visit an other user profile**
Goal	Visit a profile of an other user of Where am i
Input	Press on the line with the concerned user
Expected outcome	User has to be redirected on concerned user Profile
Actual outcome	CORRECT: After pushing on the interested line, and after conceding a few seconds to application to load the data, the user is correctly send to the profile of user chosen. This page traces the own profile, user can see other one's badges, statistics and profile image, rightly here there aren't pressing icon in the top of the screen, like friends list or ranking.

Test Case	**Visit an Friend's profile (1)**
Goal	Visit a Profile of a friend in Where am I
Input	Press on the line where Friend name appears in Friends Ranking
Expected outcome	User has to be redirected on Friend's Profile
Actual outcome	CORRECT: After pushing on the interested line, and after conceding a few seconds to application to load the data, the user is correctly send to the profile of the friend. This page traces the own profile, user can see friend's badges, statistics and profile image, rightly here there aren't pressing icon in the top of the screen, like friends list or ranking.

Test Case	**Visit an Friend's profile (2)**
Goal	Visit a Profile of a friend in Where am I
Input	Press on the line where Friend name appears in Friends List
Expected outcome	User has to be redirected on Friend's Profile
Actual outcome	CORRECT: After pushing on the interested line, and after conceding a few seconds to application to load the data, the user is correctly send to the profile of the friend. This page traces the own profile, user can see friend's badges, statistics and profile image, rightly here there aren't pressing icon in the top of the screen, like friends list or ranking.

Test Case	**Visit Friend List**
Goal	Visit own Friend list
Input	Push people icon in "Player Profile Screen"
Expected outcome	User has to direct to "Friend Screen"
Actual outcome	CORRECT: After pushing on People icon the application load a the new screen in a few seconds. Here there is a scrollable list of friends, of each friend his possible to see the avatar, the name and his online status. Online status is characterized by a play icon, it will be red if the friend his offline or green if is online.

Test Case	**Invite a Friend to play together**
Goal	Invite a Friend to play together
Input	Press on green play icon on the line of the friend concerned
Expected outcome	A modal should appears on user screen, user could have the possibility to wait the answer of the friend or cancel the request sent.
Actual outcome	CORRECT: On user screen appears a modal with a loading screen. The user is automatically put on wait for the answer of the friend. Below the loading image there is a red button with a cross, user can press it to cancel request sent to his friend. If friend accept the request, a Multiplayer Game is started, if decline, modal disappear and user return on friend list.(tested this option too)

Test Case	**Send a Friend Request**
Goal	Send a request to be friend in Where Am I to an other user
Input	In "Friend List" Insert into a text box at the bottom of the page the name of the friend and push button "ADD FRIEND"
Expected outcome	An alert says us the friend request is correctly sent
Actual outcome	CORRECT: After pushing bottom "ADD FRIEND" an alert inform the user that is request is sent with success. Using two device for this test we can verify that the friend request is really sent and arrived to the destination.

Test Case	**Accept a Friend Request**
Goal	Accept friend request and become Friend to an other user
Input	Push green bottom on the reqeust friend in "Notification Screen"
Expected outcome	The user and the sender of the request become friend in Where am I.
Actual outcome	CORRECT: After pressing green button in notification screen, on the line of the request concerned, notification disappear from the list. Now moving to friend list, user can check that the request is going successful.

8.2 Beta testing

The beta version, is a non-definitive version of the application, but already tested by developers, which is made available to some real users, trusting precisely in their unpredictable actions that could bring to light new bugs or incompatibilities of the software itself. We let try our application to some trusted people (friends and colleagues) for a week, in order to find bugs and receive feedback to improve the user experience.