POLITECNICO
MILANO 1863

# EventsApp

Software Design Document

Ivan Montalbano

Andrea Zanetti

Politecnico di Milano - A.A. 2020/2021

# 1. Introduction

## 1.1 Document purpose

In this section we will briefly introduce our application idea, features and platforms supported.

This application is developed for the course of “Design and Implementation of Mobile Applications”, thus this document shows the design decisions made during the development of the app in order to accomplish the goals set by this course.

The purpose of this document is to provide a full documentation regarding the requirements for *EventsApp*, core features, data handling, system architecture, external services and libraries, graphics, use cases and class diagrams.

## 1.2 Supported platforms

This app is supported by both Android and iOS, one of the major benefits of developing an app in Flutter. This choice has been made mainly to reach a larger target of users without rewriting code for different systems.
It is also scalable to devices with different resolutions, such as smartphones and tablets.

## 1.3 Application idea

The choice of the application came from an idea to create some sort of interactive events app. This app would include events created by users, where anyone could participate in real life or remotely.
While another type of events belonging to the category of the so called “branded entertainment” is part of the app, usually added by important brands.
By watching a video in the detail page of an event, the user would be rewarded with a coupon to spend on products related to that brand. In this way the application would have a way to receive compensation from the brands while offering a free and useful service to the users.
In future development of this app, the type of events could be extended to Podcasts and audiobooks.

## 1.4 Risk Analysis

At the beginning of the design phase, a list of essential requirements and libraries involved has been delineated.  During the development it was agreed by us that only in case the development would have been ahead of schedule, new features would have been introduced, which happened a few times. In this the risk of not completing the app in time has been reduced to the minimum.

# 2. General Overview

## 2.1 Idea

EventsApp is an application supported by both Android and iOS where users can create and participate in events. There are also special types of events where users can get rewards such as coupons related to the brand which sponsored the event.

Users are not required to be authenticated in order to use the app but without an account, events cannot be created.

It is possible for Users to login using their google account or by creating a new one connected to that app.

The main component of the app is the client side one.

## 2.2 Core features

Here are the main functionalities that our app can offer.

- Login/Register:
  - Lets users login or register to the application in case of its first visit.
  - The registering process can happen either through creating a new account with name, email and password, or by means of a log on using Google, Facebook or..
  - Once logged in, the user will be remembered if they chose so by ticking a checkbox.

- Homescreen:
  - First screen seen by the user after logging in. In the top half it shows highlighted events. While in the bottom half there are 3 tabs for events. “**All**” showing all events available. “**Brand**” showing events sponsoring brands. “**Near**” showing events around the user within a certain radius.
  - At the top bar it’s possible to search for specific events around you.
  - On the top left there’s a sidebar menu that can be opened which lets the user view their account info, create an event, go to settings or logout in order to change the user.

- Event search:
    - Accessible from the home screen, it is possible to search for events around the user.
    - It is shown a list of events including the image, title and brief description for the event.

- Event detail:
    - There are two types of events. A branded event which sponsors a certain brand, showing a video for product, title and description. Once a user has watched the video until the end, it receives a discount or coupon to buy products from that brand’s website. The second type of event shows ...

- Event creation:
    - In the event creation page it is possible to insert the following information about the event: event title, type, description, video url (if necessary), pinpoint the event location and upload an event image.

- Account overview:
    - In this page it is possible to change account profile picture and view the coupon earned so far with their respective activation codes.

- Settings:
    - In the settings page it is possible to change the language to four available ones including: English, Spanish, Chinese, German.
    - It is possible to view the phone number (if provided during registration) and email address.
    - Fingerprint security is available to be enabled in order to open the app only by using the fingerprint, offering an additional layer of protection.
    - Finally it is possible to change the account password by inserting a new one.

## 2.3 Functional requirements

Here are listed the functional requirements that are necessary in order to have the intended user experience when using this app.

General requirements:

- The default language chosen for the app is English in order to be understandable by as many people as possible.
- The application will launch for the first time in a login/register screen in order to let the user create an account. In this way it is possible to offer a better experience by storing data about the user and knowing much about his/her preferences.
- The user must be able to login in the application anonymously
- The application has to provide users the possibility to view events around them
- The application has to provide only subscribed users the possibility to create events
- The application has to provide users the possibility to search for events
- The user must be able to always view the coupons earned so far in a section devoted to their account and coupons

Home requirements:

- The home screen must display the most recent events on the top and on the bottom tabs with All, Branded and Near events
- On the left of the home screen there must be a drawer menu allowing the user to check their profile, coupons gained, create an event, settings, and logout

Event search requirements:

- The search bar should be displayed in the home screen
- The result of the search should display a list of events around the user matching the search keyword
- Events entries in the search results page should include an image, title and brief description

Event detail requirements:

- The event detail page should include the event title, description, image
- The event detail page should include the location displayed on a map
- In case of a branded event, the detail page should include a video
- Once a video included in the detail page has been watched in its entirety, the user should gain a coupon for that product

Event creation requirements:

- The create event page should allow the user to insert a title, description and type of the event
- The user should be able to insert the location of the event
- The user should be able to link a video connected to a branded event
- The create event page should allow the user to insert an image for the event

Authentication requirements:

- The app should allow the user to to create an account by inserting first name, last name, email and password
- The app should allow the user access the app by logging in anonymously
- The app should allow the user to login the app through a logon (google account)
- The app should allow a user to login by inserting their credentials, namely email and password

## 2.4 Non-functional requirements

In this paragraph are listed the non-functional requirements necessary to ensure the intended user experience.

- **Portability**: in order to reach the largest number of users possible, the app is supported by both Android and iOS systems, smartphones and tablets. In the future it is planned to release it as an app for browsers, PWA.

- **Stability**: the system should be always available at any time. Most cases of server side crashes should be avoided since the service is externally provided by Google Firebase.

- **Availability**: the service must always be available and in case of failure it must be restored as soon as possible by an administrator. In order to do so a backup facility must be present.

- **Reliability**: data stored must be reliable, meaning that it must not be corrupted or lost. An external reliable service must be chosen in order to ensure this requirement.

- **Efficiency**: algorithms and data structures must be developed to optimize the consumption of resources.

- **Extensibility**: the application must be developed creating different modules each of which handles a different feature, in this way integrating new features to the app will become much easier.

- **Maintainability**: code must be easily readable in order to be maintained without any issue by future developers.

# 3. Data Handling

## 3.1 Local data structure

All of the events and user data is stored remotely in an external database. The application is structured in such a way to enable access to data only through a Model-View-Controller architecture.
New fresh data is downloaded each time the user opens the application and is temporarily stored as objects during the application's active time in order to load them quickly to provide a better user experience.
The only data actually saved locally is managed by the library shared_preferences which takes care of saving the fingerprint for the app access protection (if enabled by the user) and storing the coupons earned by the user in json format.

## 3.2 Database design & implementation

This paragraph is meant to describe the data structures used in our project, interfacing with the external service offered by Google Firebase in order to store and retrieve data. The storage and database instances are represented by classes included in the Firebase and Firestore libraries and services, both of which are offered by Google.
In our case the only custom data structures created from scratch by us present on the client side were EventEntity and User.

- **<u>EventEntity</u>**:

    1. **ID**: positive integer
    2. **Longitude**: double, representing the longitude of the event location
    3. **Latitude**: double, representing the latitude of the event location
    4. **Title**: string, title of the event
    5. **Description**: string, description of the event
    6. **ImageUrl**: string, url to the event image uploaded on firestore
    7. **VideoUrl**: string, url to the video of the event on youtube

8. **Event Type**: string, type of the event created
9. **Creator**: string, name of the creator of the event

- **User**:
  1. **ID**: string, unique identifier of the user
  2. **Name**: string, first name and last name of the user
  3. **Email**: string, email address used for registration of the user

While the **local database cache** as reported in the previous paragraph is managed by shared_preferences which stores the values for the fingerprint app unlocking and the list of coupons earned by the user.

# 4. Architectures and component level design

## 4.1 System architecture

The main component of the EventsApp is the client-side application. The **Dart code** along with **Flutter libraries**, provides all the functionalities to interact with external services like Google Firebase, which stores information such as events, and at the same time Dart can handle the graphics to be shown to the users. solo client e interazione e server.
The paradigm chosen for the entire system is the Model-View-Controller because it is the most suited for Android and iOS apps development.
Packages have been divided into ones that model the logic to interact with data (Controller) and those that are responsible for viewing the results through a User Interface (View). Data is stored externally using Google Firestore and is modelled locally through some custom classes such as EventsEntity (Model).

## 4.2 Architectural design

In this paragraph we will focus on describing how the client-side is composed in terms of different layers:

- **Data Layer**: include classes that are responsible for local caching of data and local representation of data downloaded from Firestore, such as the representation of Events through the EventsEntity class. It interacts with the Business Layer.

- **Business Layer**: contains mainly classes that query the external Firebase service and interacts at the same time with the Model Layer and Presentation Layer

- **Presentation Layer**: collection of classes responsible for visualizing the User Interface provided data and logic coming from the Business Layer which communicates in turn with the Data Layer.

## 4.3 Security

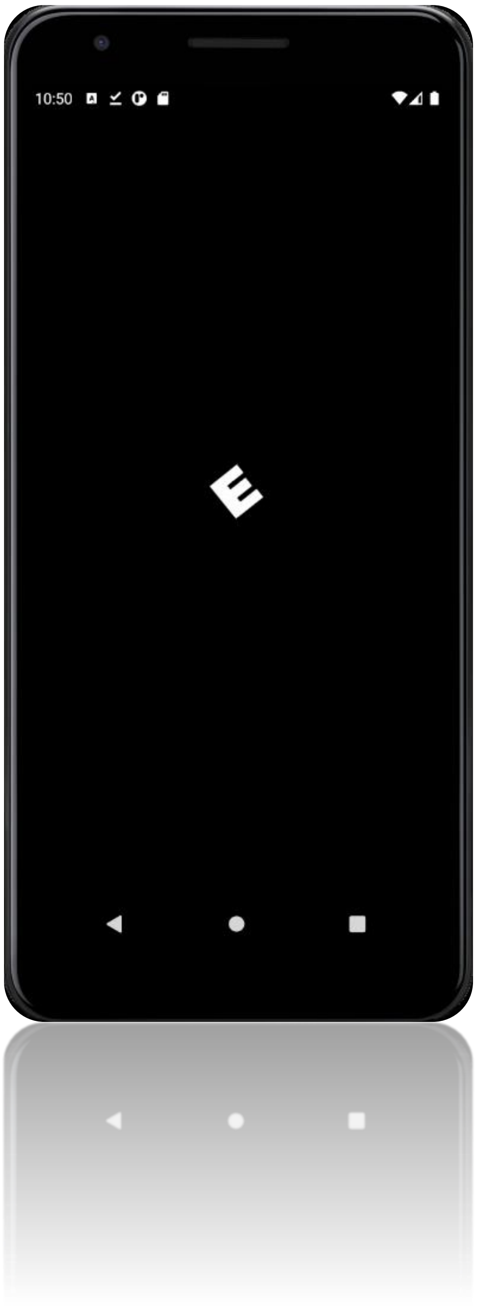
User authentication is handled by the Google Firebase platform, assumed to be secure and well tested against external attacks.
Apart from Users’ emails, there is no other sensible data stored by our app, so even in case of a potential successful attack, the damage would be minimal.

Nevertheless our app also implements **fingerprint recognition** in order to open the app only if the fingerprint of the user is successfully read. In this way an additional layer of security is added.

# 5. User interfaces

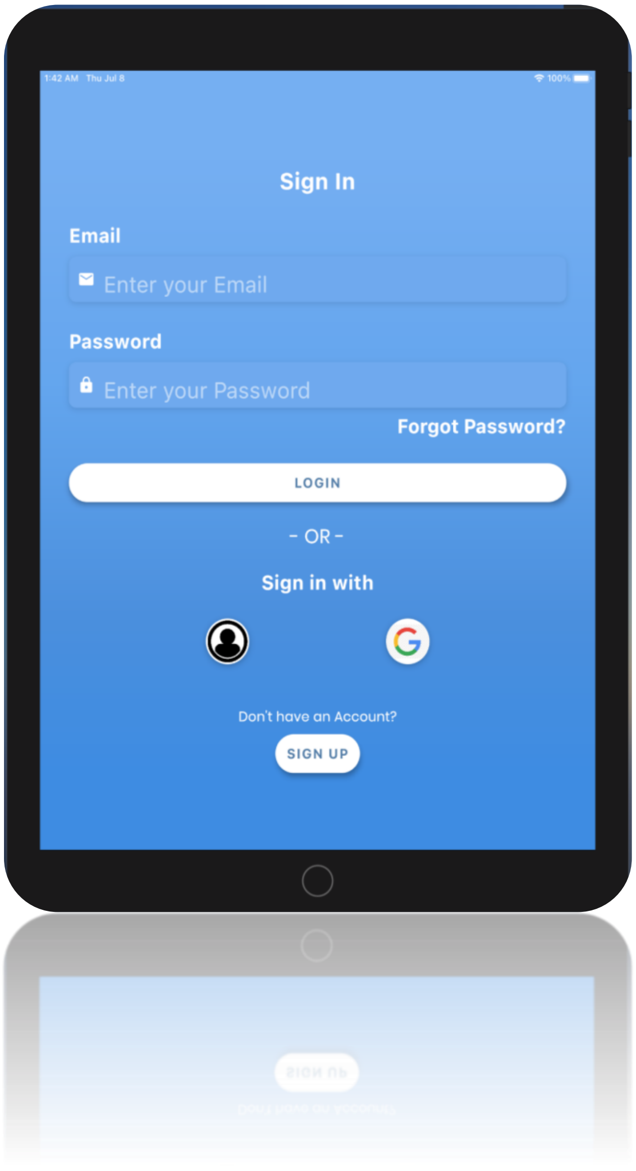
In this section we will provide a description of the main screenshots of the application. Even if our focus during development was the **smartphone's** target, our app is also already scalable to **tablets**, either Android or iOS.

## 5.1 Splash screen



The splash screen is the first screen the users see while the application starts and loads the first resources.

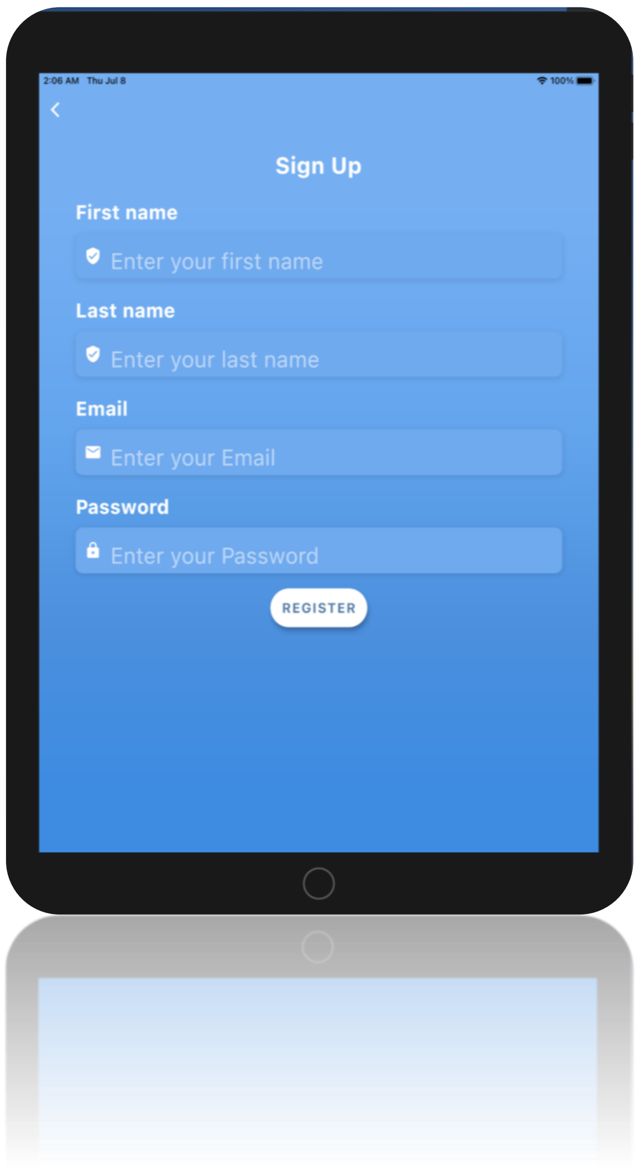
## 5.2 Authentication



This page lets the user register or login if they already have an account. It is also possible to login using a google account.

In case the user doesn't want to create an account, they can also use the app as an anonymous user.

## 5.2 Registration screen


2:06 AM Thu Jul 8 100%

Sign Up

First name

Enter your first name

Last name

Enter your last name

Email

Enter your Email

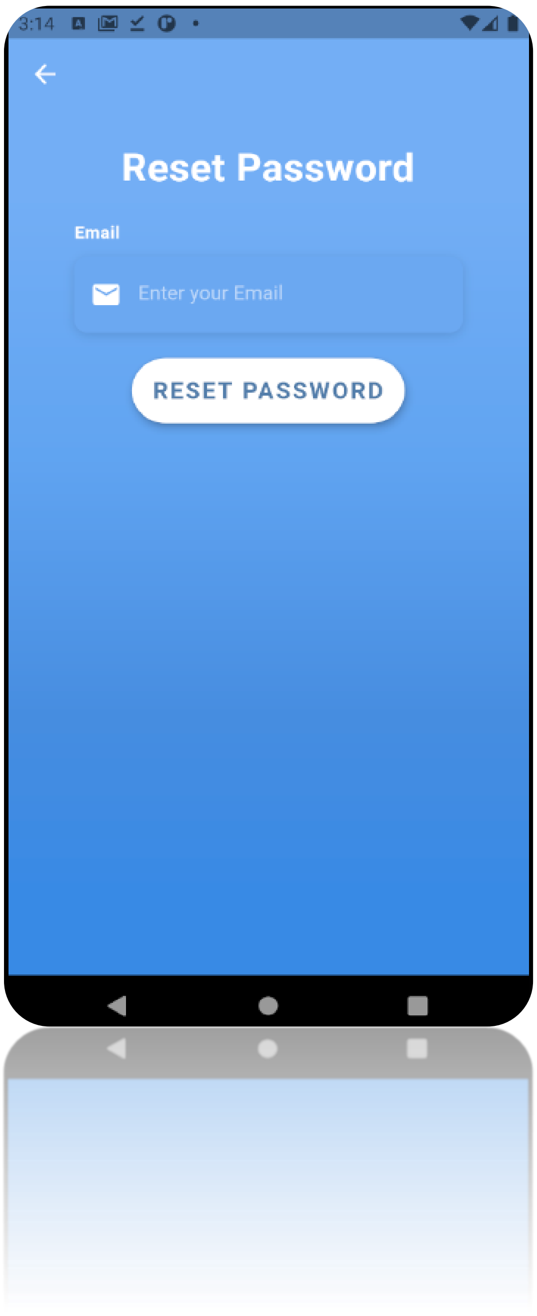
Password

Enter your Password

REGISTER

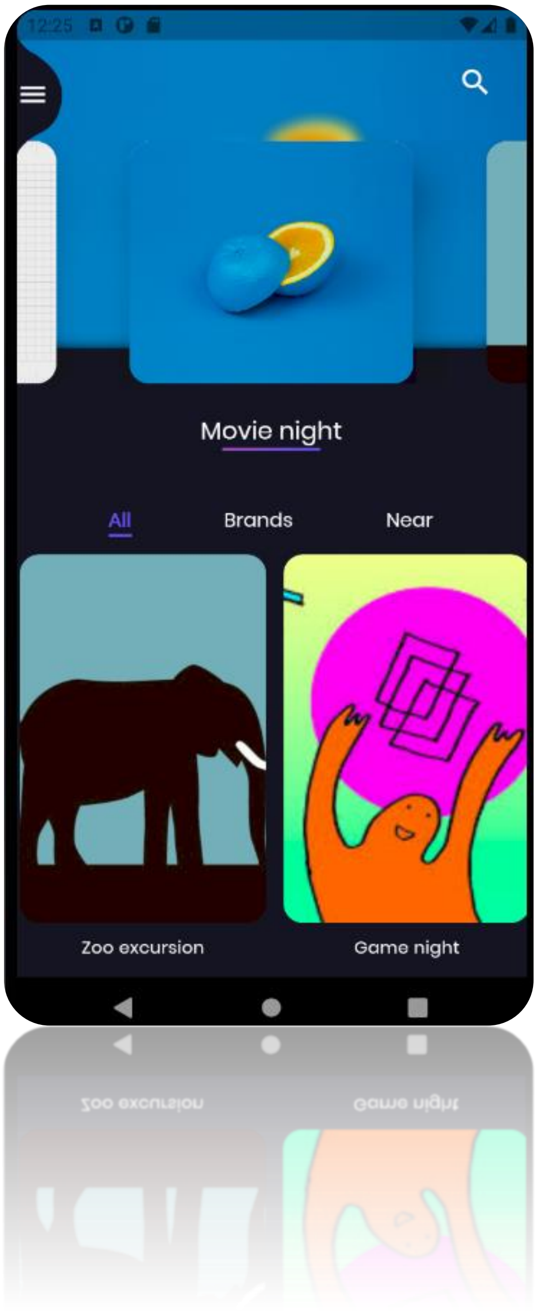
This is the screen that a user who wants to register to the application sees; here the user can insert first name and last name, provide an email address and set the preferred password which will be used to login to the application

## 5.3 Forgot password screen



This screen helps the user to reset his/her password in case the user forgets it; the user should only insert the email address associated with the account and, if the email address exists on the database, an email with a link to reset the password is sent to that email address

## 5.4 Home screen



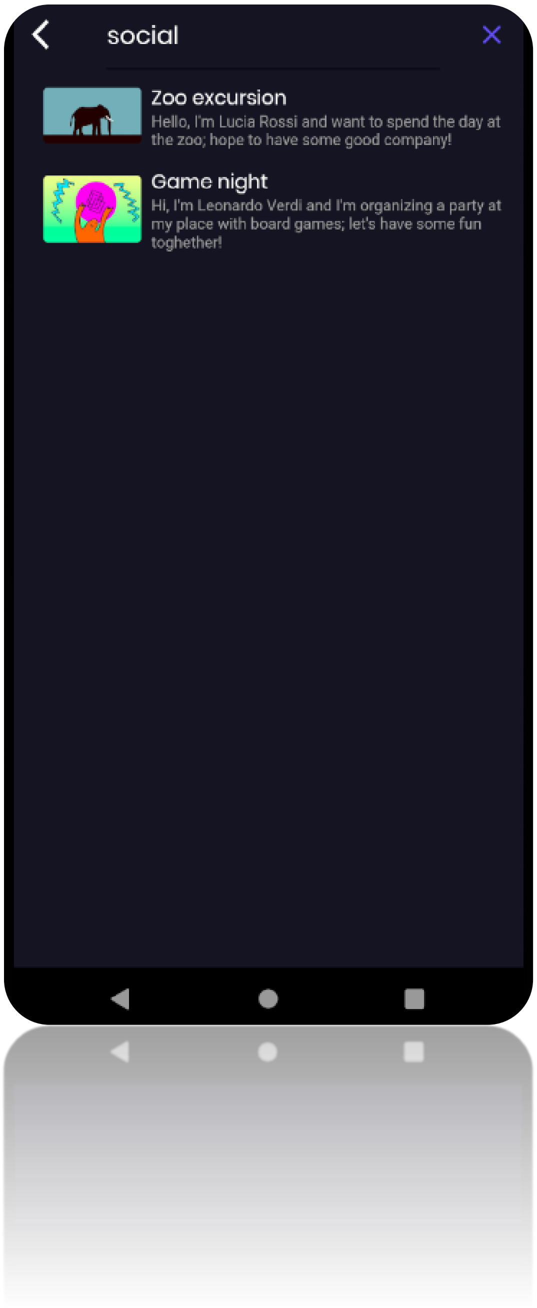
This is the first screen the user sees after login. In the top half the most recent events are shown, while in the bottom half there are three tabs showing All, Brands related or Events near the User.

In the top left corner there’s a button to show the sidebar menu.

While on the top bar the user can search for events around him/her.

By tapping on an event image, the User is directed to the Event Detail page.

## 5.5 Event search


social

Zoo excursion

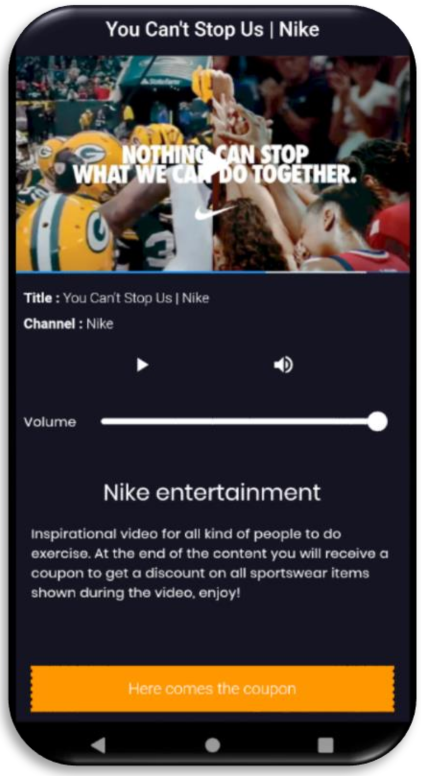
Hello, I'm Lucia Rossi and want to spend the day at the zoo; hope to have some good company!

Game night

Hi, I'm Leonardo Verdi and I'm organizing a party at my place with board games; let's have some fun toghether!

The event search page shows a list with the events matching the keyword typed by the user and that are within a certain radius from their position.
Each entry displayed has a title, image and description.
It is possible to tap on the blue “X” to delete the keyword and type a new one, or to tap the back arrow and go back to the Home screen.
By tapping on an event, the User is redirected to the Event’s Detail page.

## 5.4 Event details

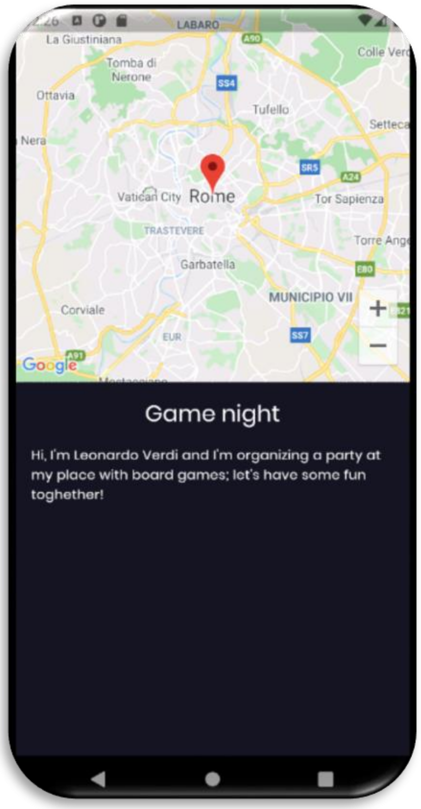


Title : You Can't Stop Us | Nike
Channel : Nike

Nike entertainment

Inspirational video for all kind of people to do exercise. At the end of the content you will receive a coupon to get a discount on all sportswear items shown during the video, enjoy!

The Event Details page shows the title, description and eventually video related to that event. Once the video has been completely watched, it adds a coupon to the user account related to that brand. This coupon may be a discount on products or services offered by that company.

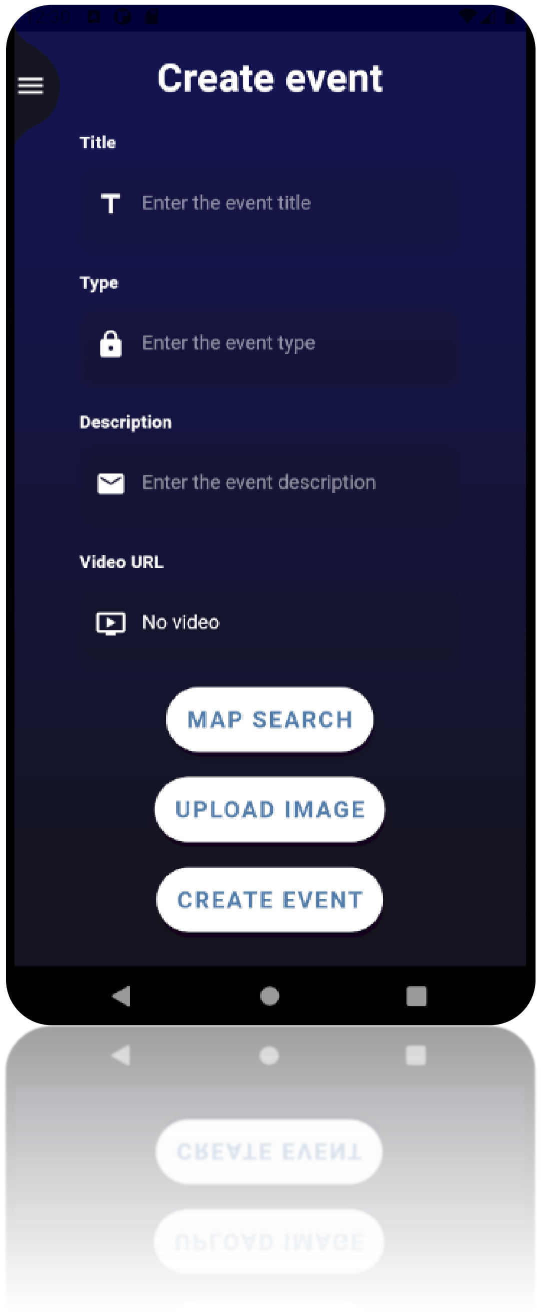


Game night

Hi, I'm Leonardo Verdi and I'm organizing a party at my place with board games; let's have some fun toghether!

In case it is a user event, it shows the location of the event and not a video

## 5.5 Event creation


Create event

Title

Enter the event title

Type

Enter the event type

Description

Enter the event description

Video URL

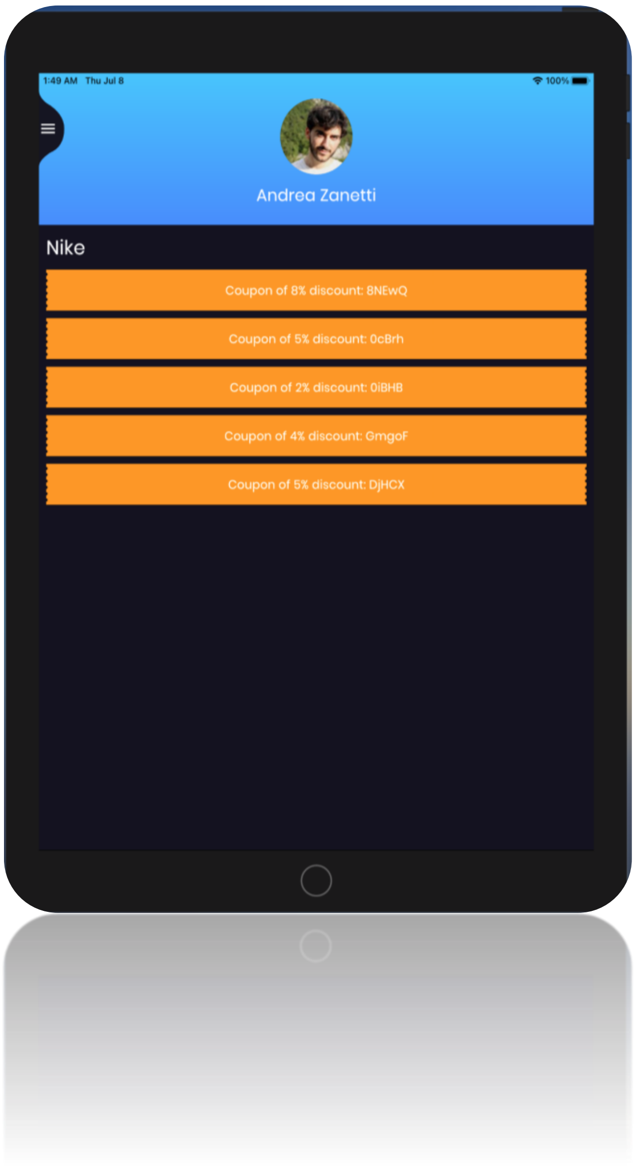
No video

In the Event Creation page it is possible for a registered User to create events.

The information needed to be inserted by the User are the Event Title, Type, Description, and optionally the Video URL, position and Map Location.

When all the useful information has been inserted, by tapping on the “Create Event” button, the event will be generated.

## 5.6 User account & coupons



Nike

Coupon of 8% discount: 8NEwQ

Coupon of 5% discount: 0cBrh

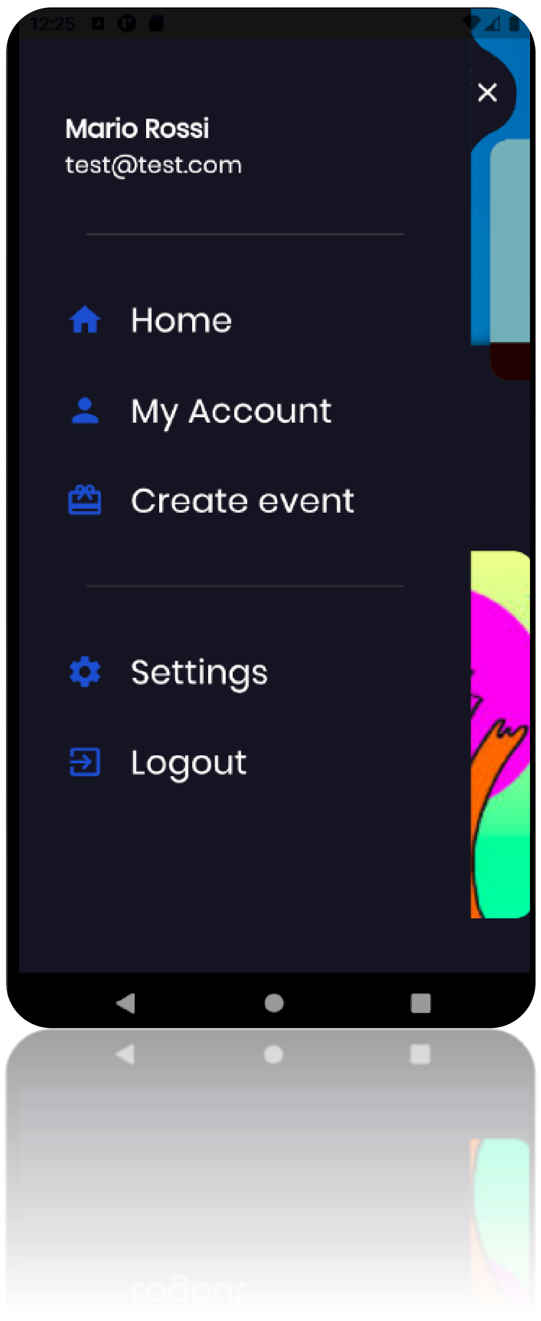
Coupon of 2% discount: 0iBHB

Coupon of 4% discount: GmgoF

Coupon of 5% discount: DjHCX

In the User Account page, the User can see the main account information such as Name, Surname, and profile picture. But most importantly the User can see a list of all the coupons earned by watching branded videos. The sidebar menu is always available on the top left corner.

## 5.7 Sidebar menu

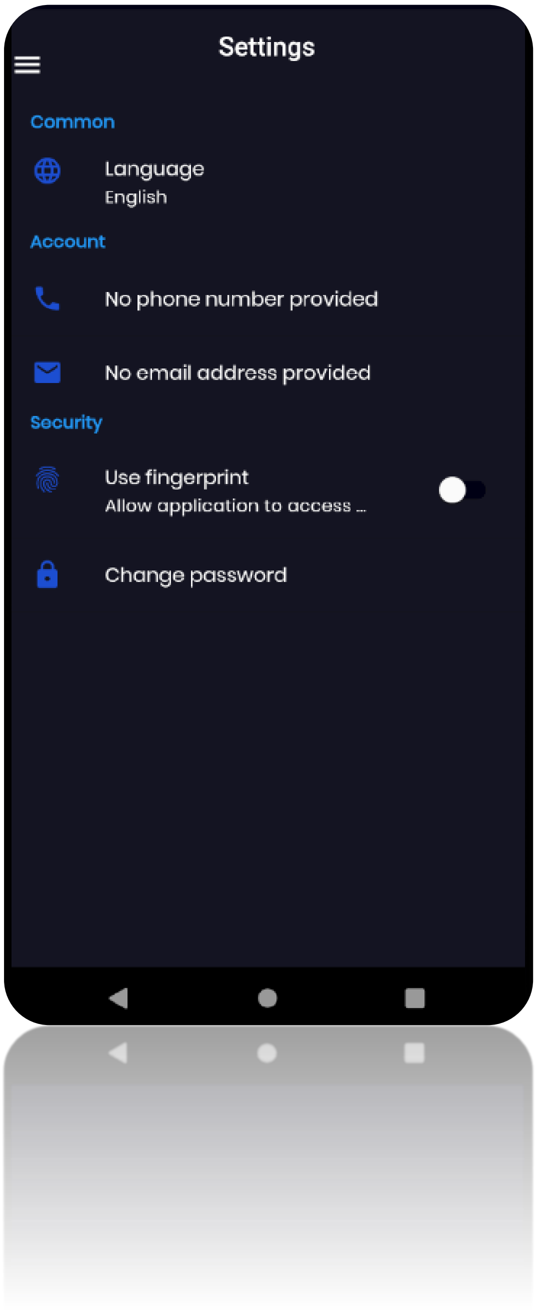


By clicking on the top left icon always available during the use of the app, a sidebar menu is shown.

It shows other screens to which the user can navigate.

These screens include the Home screen, User Account page, Create Event Page, Settings Page, and Logout.

## 5.8 Settings



Settings

Common

Language
English

Account

No phone number provided

No email address provided

Security

Use fingerprint
Allow application to access ...

Change password

In the settings page it is possible to change the app language, change the password of the account and enable the fingerprint protection when opening the application; this is possible if the device over which the application is running has a fingerprint reader.

# 6. External services and libraries

Several libraries supported by Flutter have been used for this application, some of them are necessary for the correct behavior of the application while others are meant to expand the functionalities offered to Users.
Since they are all compatible with Flutter, their integration has been seamless and transparent to the User.

Here are the main external services that we chose to integrate into our app.

## 6.1 Google Firebase

### 6.1.1 Authentication

This service handles all the authentication requests coming from users.
On the client-side it lets the user login either by email and password or by using a Google account. It also lets the User register and an account with credentials will be directly stored by the service.
On the server-side of this service, it checks if the token sent is valid and if so, it proceeds to process the request.

### 6.1.2 Firestore

All the events and coupons created are stored by the Firestore service.
When necessary, a call to this service retrieves the resources needed, so they then need to be converted and their fields are adapted to the custom class created for this purpose, EventEntity.

### 6.1.3 Storage

This service handles all the storage of the resources, like profile picture, image of events and so on.
It provides an API to call that allows to retrieve the resources when needed.

### 6.1.4 Dynamic links

This service handles the opening of links related to the application inside of the application itself.
It allows the application to be brought up as the main screen on the device if already open in background or it opens up the application if a link on the device where the application is installed is clicked.
When the application is released on the Store, the dynamic links will open the Store of the device to download the application if it is not already installed.

## 6.2 Google Play Services

### 6.2.1 Authentication

This service connects with the User's Google profile in order to sign him/her up using the profile or simply logging in in case they're already registered.
It is then used by Google Firebase

### 6.2.2 Location Places

This service is used in the Event Creation page when specifying a location for the event. This service speeds up this process because it lets the User to search for places quickly by implementing an autocomplete feature.

### 6.2.3 Maps

By using this service it is possible to display the events on an interactive map that can eventually be opened in the main Google Maps app and a variety of options such as route suggestions can be exploited by the User.

### 6.2.4 IFrame Player API

This service embeds youtube videos into the app and offers tracking data directly from the YouTube Engine through its API calls. By exploiting them, the app can have additional information about the channel and the target audience to which that video is directed to.

## 6.3 Other minor libraries

- **Geolocator**: library that allows to get to User location. It works across different platforms, both Android and iOS.

- **Local_auth (Fingerprint)**: a library that enables the possibility of unlocking the access to the app only by using the fingerprint of the owner. It works across different platforms, both Android and iOS.

- **Shared_preferences**: library that stores locally data such as the coupons earned by the User and the fingerprint saved to unlock the app. In case the User has no Internet connection, he/she can always access the app in case it’s locked with the fingerprint and retrieve the coupons stored locally. It works across different platforms, both Android and iOS.

- **Youtube_player_flutter**: library that enables the possibility to embed youtube videos into the app and get tracking data from it. For example when a User has entirely watched a video, this library can trigger an event. It works across different platforms, both Android and iOS.

- **Permission_handler**: library that specifically handles permissions requests when they are requested by the app. In our case it was needed to get the current location permissions. When needed, this app triggers the popup prompting the User to grant this permission. It works across different platforms, both Android and iOS.

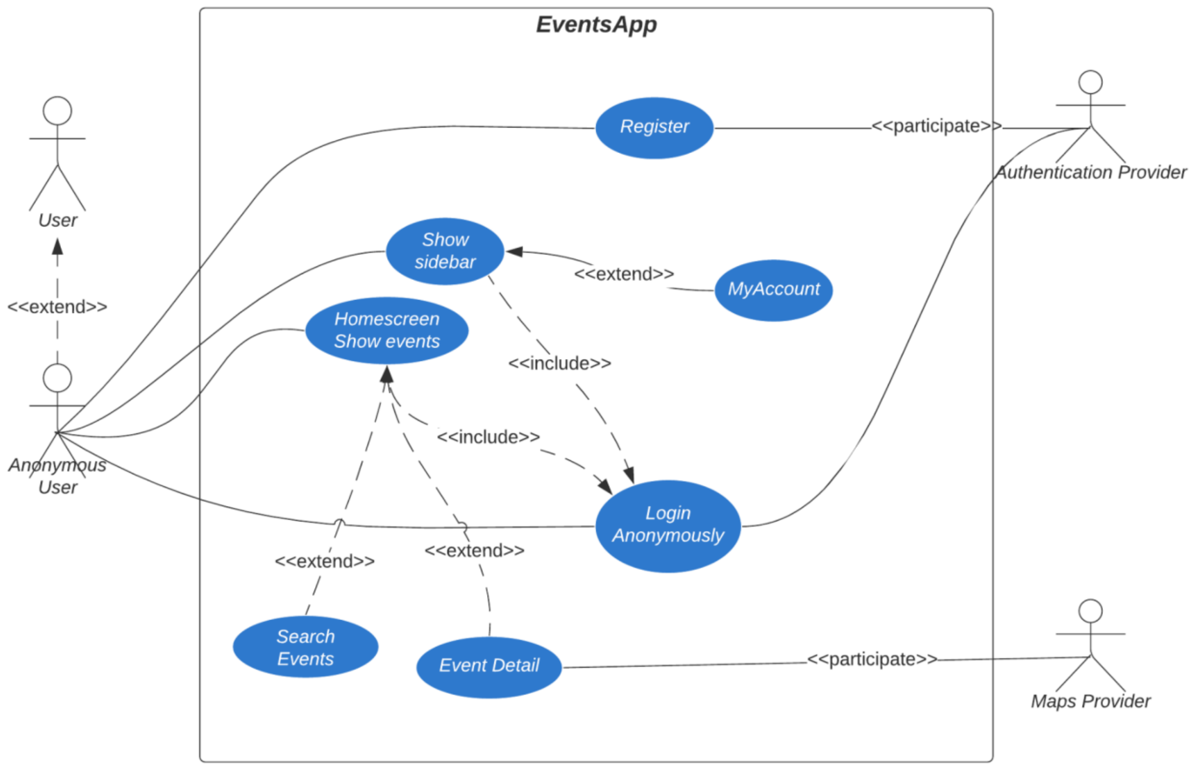
# 7. UML diagrams

In this section we present some diagrams such as use case diagrams and class diagrams in order to describe in a better way the interaction between the user and application or how the packages are divided and main classes involved.

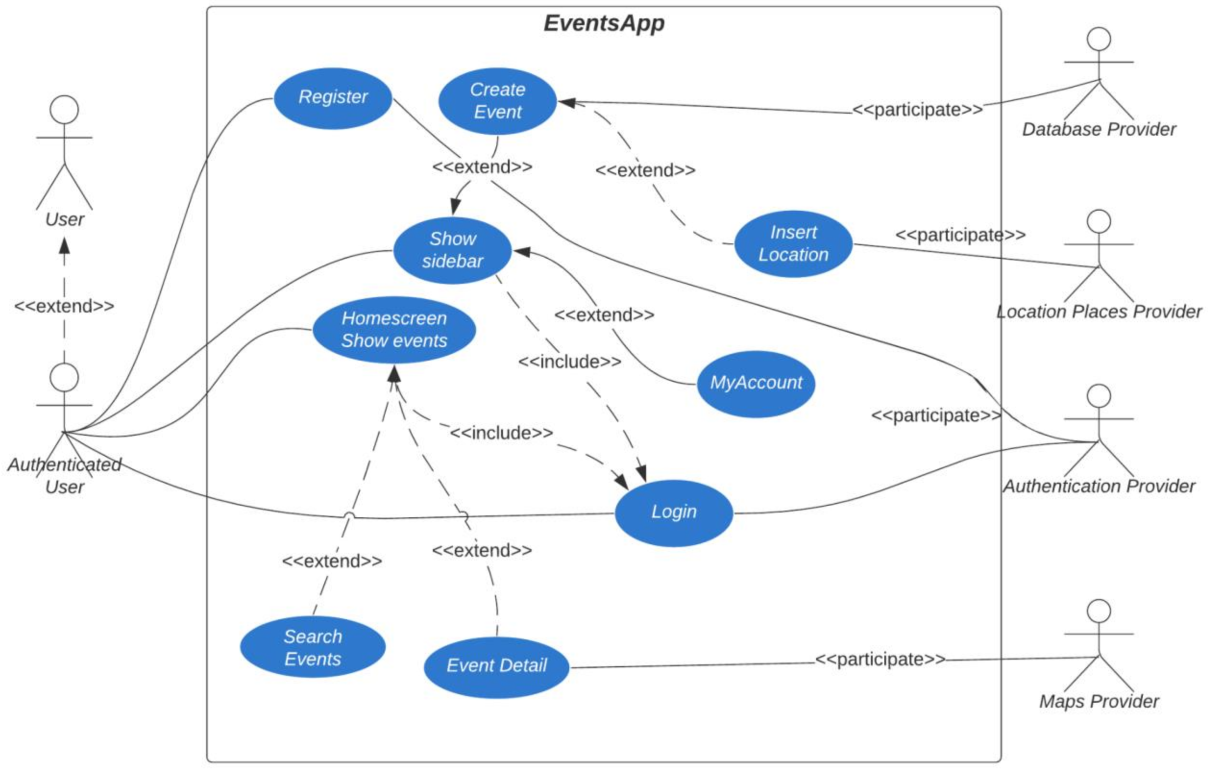
## 7.1 Use Case diagrams

The following use cases show the main action the Users can perform while using the app. In this case Users are the actors that trigger the flow of action.

### 7.1.1 Anonymous user interaction



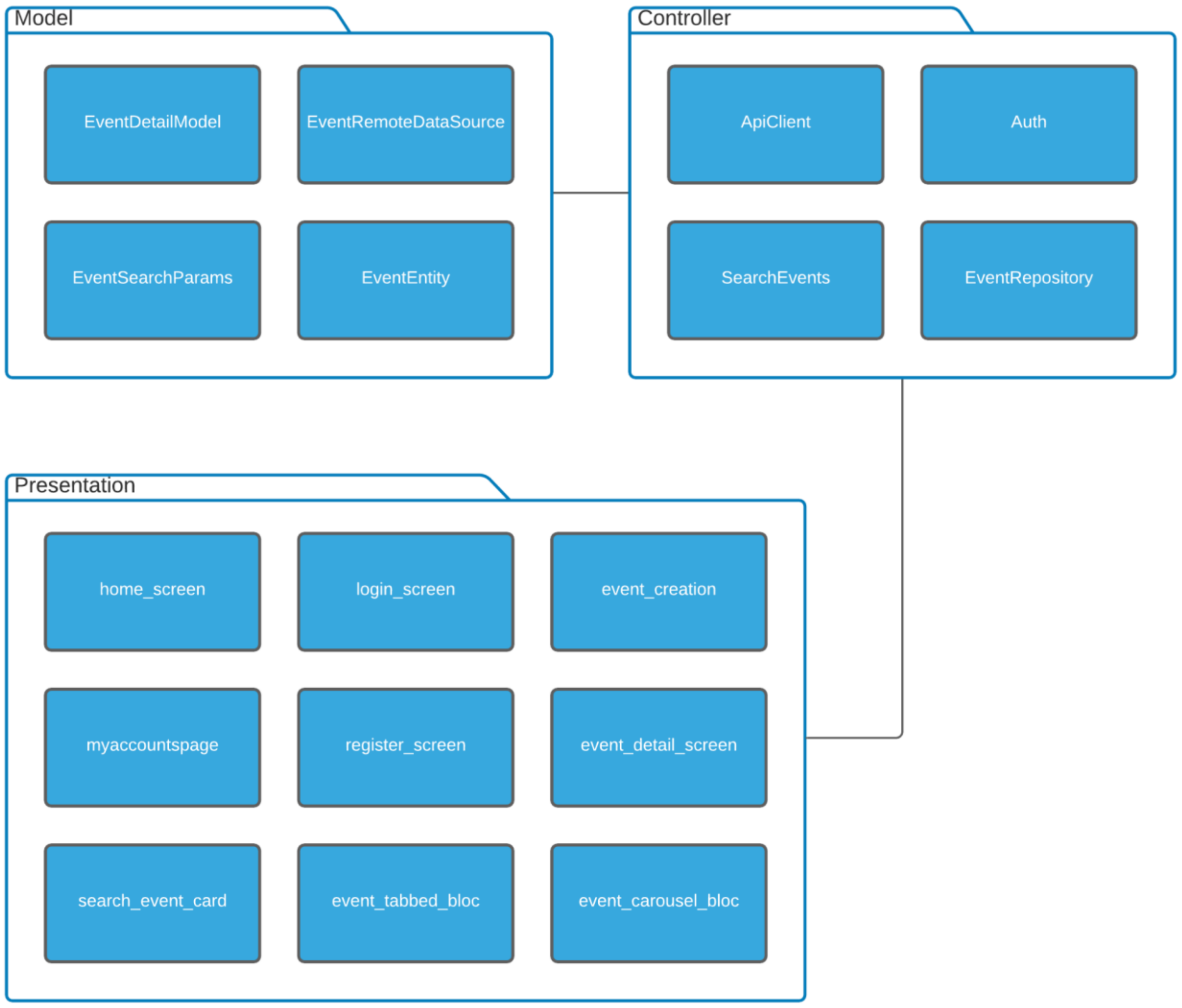
## 7.1.2 Logged user interaction



## 7.2 Class diagrams

In this section are shown the main classes involved grouped into the 3 main packages: Model, Controller, Presentation.

The Controller can interact with both the Model and the Presentation Layer.



# 8. Test cases

This section lists the main tests done on the EventsApp application.

| **Test Case** | **`Navigation: Login screen: sign in is present`** |
|---|---|
| Goal | Text “Sign in” is present |
| Input | None |
| Expected outcome | Text “Sign in” is visible |
| Actual outcome | CORRECT: The text “Sign in” is present in the screen |

| **Test Case** | **`Navigation: Login screen: email is present`** |
|---|---|
| Goal | Text “Email” is present |
| Input | None |
| Expected outcome | Text “Email” is visible |
| Actual outcome | CORRECT: The text “Email” is present in the screen |

| Test Case | `Navigation: Login screen: password is present` |
|---|---|
| Goal | Text “Password” is present |
| Input | None |
| Expected outcome | Text “Password” is visible |
| Actual outcome | CORRECT: The text “Password” is present in the screen |

| Test Case | `Navigation: Login screen: noaccount is present` |
|---|---|
| Goal | Text “Don’t have an Account?” is present |
| Input | None |
| Expected outcome | Text “Don’t have an Account?” is visible |
| Actual outcome | CORRECT: The text “Don’t have an Account?” is present in the screen |

| Test Case | `Navigation: Login screen: forgot password is present` |
|---|---|
| Goal | Text “Forgot Password?” is present |
| Input | None |
| Expected outcome | Text “Forgot Password?” is visible |
| Actual outcome | CORRECT: The text “Forgot Password?” is present in the screen |

| **Test Case** | **`Navigation: Login screen: login is present`** |
|---|---|
| Goal | Text “LOGIN” is present |
| Input | None |
| Expected outcome | Text “LOGIN” is visible inside a button |
| Actual outcome | CORRECT: The text “LOGIN” is present in the screen inside a button |

| **Test Case** | **`Navigation: Login screen: sign up is present`** |
|---|---|
| Goal | Text “SIGN UP” is present |
| Input | None |
| Expected outcome | Text “SIGN UP” is visible inside a button |
| Actual outcome | CORRECT: The text “SIGN UP” is present in the screen inside a button |

| **Test Case** | **`Navigation: Login screen: register screen is reachable`** |
|---|---|
| Goal | Register screen is reachable from login screen |
| Input | Scroll till “SIGN UP” button is visible and then tap it |
| Expected outcome | The register screen shows up |
| Actual outcome | CORRECT: The “SIGN UP” button is present when scrolling down the screen and when tapped it shows the register screen displaying the “REGISTER” text inside a button as well as the “Sign Up” text constituting the title of the screen |

| **Test Case** | **`Navigation: Login screen: forgot password screen is reachable`** |
|---|---|
| Goal | Forgot password screen is reachable from login screen |
| Input | Tap the text “Forgot Password?” just below the password text field of the login screen |
| Expected outcome | The forgot password screen shows up |
| Actual outcome | CORRECT: When the text “Forgot Password?” is tapped it shows the forgot password screen displaying the “RESET PASSWORD” text inside a button |

| **Test Case** | **`Navigation: Login screen: Log in action with email and password`** |
|---|---|
| Goal | Log in to the application |
| Input | Username and password of an existing user in Firebase authentication database |
| Expected outcome | Log in succeeds and the home page screen is displayed |
| Actual outcome | CORRECT: After inserting the username and password in the text fields “Email” and “Password” in the login screen and pressing the “LOGIN” button, it is possible to access the application and the home page screen is displayed showing events defined as “Tabbed” in the lower side of the screen and widgets defined as “Carousel” in the upper side of the screen |

| **Test Case** | **`Navigation: Login screen: Log in action as guest`** |
|---|---|
| Goal | Log in to the application |
| Input | Tap on the guest icon present on the login screen |
| Expected outcome | Log in succeeds and the home page screen is displayed |
| Actual outcome | CORRECT: After tapping the guest icon present on the login screen, it is possible to access the application and the home page screen is displayed showing events defined as “Tabbed” in the lower side of the screen and widgets defined as “Carousel” in the upper side of the screen |

| **Test Case** | **`Navigation: Login screen: Log in action with google account`** |
|---|---|
| Goal | Log in to the application |
| Input | Tap on the Google icon present on the login screen |
| Expected outcome | Log in succeeds and the home page screen is displayed |
| Actual outcome | CORRECT: After tapping the Google icon present on the login screen a system dialog appears asking which Google account to use to login to the application; after selecting a Google account, it is possible to access the application and the home page screen is displayed showing events defined as “Tabbed” in the lower side of the screen and widgets defined as “Carousel” in the upper side of the screen |

| **Test Case** | **`Navigation: Sidebar menu: sidebar elements are visible`** |
|---|---|
| Goal | Sidebar menu items are displayed when sidebar gets opened |
| Input | Tap the clipper element on the upper-left side of the screen to open up the sidebar |
| Expected outcome | The sidebar menu is opened and the elements expected are visible |
| Actual outcome | CORRECT: After tapping the clipper element visible from the home page screen the sidebar menu is opened and there are the elements expected: “Home”, “My Account”, “Create event”, “Settings” and “Logout” |

| **Test Case** | **`Navigation: Sidebar menu: my account item brings up my account screen`** |
|---|---|
| Goal | My account screen is displayed when “My Account” menu item gets tapped |
| Input | Tap “My Account” menu item |
| Expected outcome | My account screen is displayed |
| Actual outcome | CORRECT: After tapping the clipper element to open the sidebar and then tapping “My Account” menu item from the sidebar, the screen “My account” is displayed and the “Avatar” element of the screen is displayed as expected |

| **Test Case** | **`Navigation: Sidebar menu: create event item brings up event creation screen`** |
|---|---|
| Goal | Create event screen is displayed when “Create event” menu item gets tapped |
| Input | Tap “Create event” menu item |
| Expected outcome | Create event screen is displayed |
| Actual outcome | CORRECT: After tapping the clipper element to open the sidebar and then tapping “Create event” menu item from the sidebar, the screen “Create event” is displayed and the title “Create event” of the screen is displayed as expected |

| **Test Case** | **`Navigation: Sidebar menu: settings item brings up setting screen`** |
|---|---|
| Goal | Settings screen is displayed when “Settings” menu item gets tapped |
| Input | Tap “Settings” menu item |
| Expected outcome | Settings screen is displayed |
| Actual outcome | CORRECT: After tapping the clipper element to open the sidebar and then tapping “Settings” menu item from the sidebar, the screen “Settings” is displayed and the title “Settings” of the screen is displayed as expected |

| **Test Case** | **`Navigation: Sidebar menu: home item brings up home page screen`** |
|---|---|
| Goal | Home page screen is displayed when “Home” menu item gets tapped |
| Input | Tap “Home” menu item |
| Expected outcome | Home page screen is displayed |
| Actual outcome | CORRECT: After tapping the clipper element to open the sidebar and then tapping “Home” menu item from the sidebar, the screen “Home page” is displayed and the screen is showing events defined as “Tabbed” in the lower side of the screen and widgets defined as “Carousel” in the upper side of the screen |

| **Test Case** | **`Navigation: Sidebar menu: logout item brings up log in screen and logs out the user`** |
|---|---|
| Goal | Current logged in user gets logged out and the login screen is displayed |
| Input | Tap “Logout” menu item |
| Expected outcome | Log out succeeds and the login screen is displayed |
| Actual outcome | CORRECT: After tapping the clipper element to open the sidebar and then tapping “Logout” menu item from the sidebar, the screen “Login” is displayed and the user is logged out from the application |

| **Test Case** | **`Forgot password widget: forgot password button is present and when tapped brings up forgot password screen`** |
|---|---|
| Goal | **`Forgot password button is present and when tapped brings up forgot password screen`** |
| Input | Tap forgot password text under password text field in login screen |
| Expected outcome | Forgot password screen is displayed |
| Actual outcome | CORRECT: After tapping the forgot password text, the “Forgot password” screen is shown |

| **Test Case** | **`Forgot password widget: reset password title is present`** |
|---|---|
| Goal | The title “Reset Password” of the forgot password screen is visible |
| Input | Tap forgot password text under password text field in login screen |
| Expected outcome | The title “Reset Password” of the forgot password screen is displayed |
| Actual outcome | CORRECT: After tapping the forgot password text, the “Forgot password” screen is shown and the title “Reset Password” gets displayed |

| **Test Case** | **`Forgot password widget: reset password text field is present`** |
|---|---|
| Goal | The text field “Email” of the forgot password screen is visible |
| Input | Tap forgot password text under password text field in login screen |
| Expected outcome | The text field “Email” of the forgot password screen is displayed |
| Actual outcome | CORRECT: After tapping the forgot password text, the “Forgot password” screen is shown and the text field “Email” gets displayed |

| **Test Case** | **`Forgot password widget: reset password button is present`** |
|---|---|
| Goal | The button “RESET PASSWORD” of the forgot password screen is visible |
| Input | Tap forgot password text under password text field in login screen |
| Expected outcome | The button “RESET PASSWORD” of the forgot password screen is displayed |
| Actual outcome | CORRECT: After tapping the forgot password text, the “Forgot password” screen is shown and the button “RESET PASSWORD” gets displayed |

| **Test Case** | **`Forgot password widget: reset password action`** |
|---|---|
| Goal | An email for resetting the password is sent to the email address provided if it is binded with an existing account |
| Input | Insert an email address associated with an existing account in the “Email” text field of the forgot password screen |
| Expected outcome | The email is sent to the provided email address |
| Actual outcome | CORRECT: After inserting the email address of an existing account in the “Email” text field and tapping the “RESET PASSWORD” button, an email is sent with a valid link for resetting the password for the user associated with the provided email address |

| **Test Case** | **`Register widget: register button is present and when tapped brings up register screen`** |
|---|---|
| Goal | “SIGN UP” button is present and when tapped brings up register screen |
| Input | Tap on “SIGN UP” button present in the login screen after scrolling down |
| Expected outcome | The register screen is displayed after the “SIGN UP” button is tapped |
| Actual outcome | CORRECT: The “SIGN UP” button is present when scrolling down the login screen and when tapped it shows the register screen |

| **Test Case** | **`Register widget: first name is present`** |
|---|---|
| Goal | “First name” text is visible on the register screen |
| Input | Tap on “SIGN UP” button present in the login screen after scrolling down |
| Expected outcome | “First name” text is displayed on the register screen |
| Actual outcome | CORRECT: When register screen is displayed, the text “First name” is visible |

| **Test Case** | **`Register widget: last name is present`** |
|---|---|
| Goal | “Last name” text is visible on the register screen |
| Input | Tap on “SIGN UP” button present in the login screen after scrolling down |
| Expected outcome | “Last name” text is displayed on the register screen |
| Actual outcome | CORRECT: When register screen is displayed, the text “Last name” is visible |

| **Test Case** | `Register widget: email is present` |
|---|---|
| Goal | “Email” text is visible on the register screen |
| Input | Tap on “SIGN UP” button present in the login screen after scrolling down |
| Expected outcome | “Email” text is displayed on the register screen |
| Actual outcome | CORRECT: When register screen is displayed, the text “Email” is visible |

| **Test Case** | `Register widget: password is present` |
|---|---|
| Goal | “Password” text is visible on the register screen |
| Input | Tap on “SIGN UP” button present in the login screen after scrolling down |
| Expected outcome | “Password” text is displayed on the register screen |
| Actual outcome | CORRECT: When register screen is displayed, the text “Password” is visible |

| **Test Case** | **`Register widget: Register action`** |
|---|---|
| Goal | A new user gets registered and logged in to the application |
| Input | Insert first name, last name, email and password in the corresponding text fields displayed on the screen and tap the “REGISTER” button |
| Expected outcome | An email to verify the account gets sent to the email address provided, a new user with the details provided in the registration screen is created on the Firebase authentication database and the user is logged in |
| Actual outcome | CORRECT: After inserting the requested information in the text fields displayed on the register screen, a new user is created and logged in to the application, provided that the email address is not already associated with an existing account; also, an email is sent to the email address provided to verify the email address |

| **Test Case** | **`Home screen and event details widgets: carousel widget is present`** |
|---|---|
| Goal | The upper side of the home page screen is filled with the carousel widget |
| Input | Log in to the application |
| Expected outcome | The carousel widget is displayed in the upper side of the screen |
| Actual outcome | CORRECT: After logging in to the application the carousel widget is shown in the upper side of the screen |

| **Test Case** | **`Home screen and event details widgets: tabbed widget is present`** |
|---|---|
| Goal | The lower side of the home page screen is filled with the tabbed widget |
| Input | Log in to the application |
| Expected outcome | The tabbed widget is displayed in the lower side of the screen |
| Actual outcome | CORRECT: After logging in to the application the tabbed widget is shown in the lower side of the screen |

| **Test Case** | **`Home screen and event details widgets: event detail screen for no-brand events is visible and shows the map`** |
|---|---|
| Goal | Opening an event which is not of type “Brand” shows the event detail screen which displays a map |
| Input | Tap a carousel or a tabbed widget not of type “Brand” |
| Expected outcome | The event detail screen is displayed and there is a map shown |
| Actual outcome | CORRECT: After tapping an event not of type “Brand”, the event detail screen is displayed and there is a map showing the location of the event |

| **Test Case** | `Home screen and event details widgets: event detail screen for brand events is visible and shows a video` |
|---|---|
| Goal | Opening an event which is of type “Brand” shows the event detail screen which displays a video |
| Input | Tap a carousel or a tabbed widget of type “Brand” |
| Expected outcome | The event detail screen is displayed and there is a video which starts to play |
| Actual outcome | CORRECT: After tapping an event of type “Brand”, the event detail screen is displayed and there is a video that starts playing |

| **Test Case** | `Home screen and event details widgets: logout` |
|---|---|
| Goal | Log out of the application |
| Input | Open the sidebar menu and tap on “Logout” menu item |
| Expected outcome | The user is logged out from the application and the login screen is displayed |
| Actual outcome | CORRECT: After opening the sidebar through the clipper element and tapping the “Logout” menu item, the user is logged out and is redirected to the login screen |

| **Test Case** | **`Create event: screen title is present`** |
|---|---|
| Goal | The “Create event” text is visible on the create event screen |
| Input | Open the sidebar menu and tap on “Create event” menu item |
| Expected outcome | The “Create event” text is displayed on the create event screen |
| Actual outcome | CORRECT: After opening the sidebar through the clipper element and tapping the “Create event” menu item, the “Create event” text is visible |

| **Test Case** | **`Create event: event title is present`** |
|---|---|
| Goal | The “Title” text is visible on the create event screen |
| Input | Open the sidebar menu and tap on “Create event” menu item |
| Expected outcome | The “Title” text is displayed on the create event screen |
| Actual outcome | CORRECT: After opening the sidebar through the clipper element and tapping the “Create event” menu item, the “Title” text is visible |

| **Test Case** | **`Create event: event type is present`** |
|---|---|
| Goal | The “Type” text is visible on the create event screen |
| Input | Open the sidebar menu and tap on “Create event” menu item |
| Expected outcome | The “Type” text is displayed on the create event screen |
| Actual outcome | CORRECT: After opening the sidebar through the clipper element and tapping the “Create event” menu item, the “Type” text is visible |

| **Test Case** | **`Create event: event description is present`** |
|---|---|
| Goal | The “Description” text is visible on the create event screen |
| Input | Open the sidebar menu and tap on “Create event” menu item |
| Expected outcome | The “Description” text is displayed on the create event screen |
| Actual outcome | CORRECT: After opening the sidebar through the clipper element and tapping the “Create event” menu item, the “Description” text is visible |

| **Test Case** | **`Create event: event video URL is present`** |
|---|---|
| Goal | The “Video URL” text is visible on the create event screen |
| Input | Open the sidebar menu and tap on “Create event” menu item |
| Expected outcome | The “Video URL” text is displayed on the create event screen |
| Actual outcome | CORRECT: After opening the sidebar through the clipper element and tapping the “Create event” menu item, the “Video URL” text is visible |

| **Test Case** | **`Create event: event map search button is present`** |
|---|---|
| Goal | The “MAP SEARCH” text of a button is visible on the create event screen |
| Input | Open the sidebar menu and tap on “Create event” menu item |
| Expected outcome | The “MAP SEARCH” text of a button is displayed on the create event screen |
| Actual outcome | CORRECT: After opening the sidebar through the clipper element and tapping the “Create event” menu item, the “MAP SEARCH” text of a button is visible |

| **Test Case** | **`Create event: event upload image button is present`** |
|---|---|
| Goal | The “UPLOAD IMAGE” text of a button is visible on the create event screen |
| Input | Open the sidebar menu and tap on “Create event” menu item |
| Expected outcome | The “UPLOAD IMAGE” text of a button is displayed on the create event screen |
| Actual outcome | CORRECT: After opening the sidebar through the clipper element and tapping the “Create event” menu item, the “UPLOAD IMAGE” text of a button is visible |

| **Test Case** | **`Create event: event create event button is present`** |
|---|---|
| Goal | The “CREATE EVENT” text of a button is visible on the create event screen |
| Input | Open the sidebar menu and tap on “Create event” menu item |
| Expected outcome | The “CREATE EVENT” text of a button is displayed on the create event screen |
| Actual outcome | CORRECT: After opening the sidebar through the clipper element and tapping the “Create event” menu item, the “CREATE EVENT” text of a button is visible |

| **Test Case** | **`Create event: event creation`** |
|---|---|
| Goal | A new event is created and stored on Firebase database |
| Input | Fill all the text fields of the event creation screen, insert a location through the “MAP SEARCH” button, upload an image to Firestore storage through the “UPLOAD IMAGE” button and tap on the “CREATE EVENT” button |
| Expected outcome | A new event is created and the user is brought to the home page screen |
| Actual outcome | CORRECT: Once filled all the text fields of the event creation screen, inserted a location through the “MAP SEARCH” button which uses Google Places to search for an address, uploaded an image to Firestore storage through the “UPLOAD IMAGE” button and tapped on the “CREATE EVENT” button, the event is created and the user is redirected to the home page screen |

| **Test Case** | **`Settings: change password`** |
|---|---|
| Goal | Change the password of the current user |
| Input | Current password and new password |
| Expected outcome | The user password is updated and the user is transparently re-authenticated with the new password |
| Actual outcome | CORRECT: After inserting the current password and the new password in the change password form, an alert dialog informs the user of the successfulness of the operation and the new password is updated for the account |