

POLITECNICO DI MILANO



**POLITECNICO**  
MILANO 1863

DESIGN AND IMPLEMENTATION OF MOBILE  
APPLICATIONS

**iSport**

**Design Document**

*di*

*Gianluigi Oliva*

February 10, 2019

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Purpose . . . . .	3
1.2	Intended Audience . . . . .	3
1.3	Definitions, acronyms, abbreviations . . . . .	3
1.4	Mobile Application Scope . . . . .	4
1.5	Framework . . . . .	5
1.6	Functional Requirements . . . . .	5
1.7	Non Functional Requirements . . . . .	6
1.8	Assumptions, Dependencies and Constraints	
	6	
<b>2</b>	<b>Architecture</b>	<b>7</b>
2.1	Database . . . . .	7
2.2	Client . . . . .	8
<b>3</b>	<b>Use case functional requirements analysis</b>	<b>9</b>
<b>4</b>	<b>Sequence Diagram</b>	<b>31</b>
<b>5</b>	<b>User Interfaces</b>	<b>44</b>
<b>6</b>	<b>External Services and Libraries</b>	<b>53</b>
<b>7</b>	<b>Software System Attribute</b>	<b>59</b>
7.1	Reliability . . . . .	59
7.2	Availability . . . . .	59
7.3	Security . . . . .	59
7.4	Maintainability . . . . .	59
7.5	Usability . . . . .	59
<b>8</b>	<b>Test Cases</b>	<b>60</b>
<b>9</b>	<b>Cost Estimation</b>	<b>61</b>

# 1 Introduction

## 1.1 Purpose

The purpose of this document is to describe the design and prototyping phases used for the realization of the “iSport” mobile application. In detail, the main components, features and user experience will be discussed.

The main aim of iSport application is visualizing information and data related to the sport field. In particular, we will focus on the most relevant daily news, also showing a section with all football final scores. Moreover, to encourage interaction throughout a community discussion, the application will provide a system of live chat to exchange opinions.

This project is the result of the implementation of the knowledge acquired during the course ”Design and Implementation of Mobile Applications” provided by the Milan Polytechnic.

## 1.2 Intended Audience

This document is produced for those who develop, evaluate and use iSport mobile application:

- The engineers who had the idea and developed the application.
- The testers that must verify the effective implementation of all the described components and functions.
- The user who will use the application and take advantage of its functionalities.
- The future contributors who wish to develop new features.

## 1.3 Definitions, acronyms, abbreviations

### Definitions

- **Platform:** The application as a whole.
- **Guest:** A person who can view public contents
- **User:** A guest who already performed the login operation successfully.

## *1 Introduction*

- **Match:** A match between two teams that has already occurred or is in progress
- **Framework:** Reusable set of libraries or classes for a software system.
- **News:** A news related to the world of sport present in some journalism
- **Forecast:** A prediction on the football scores among a class of possible results
- **Odds:** The remuneration value of a prediction relative to a given match
- **REST:** is a way of providing interoperability between computer systems on the Internet.

## **Acronyms**

- **MVC:** Model - View - Controller
- **HTTPS:** HyperText Transfer Protocol Secure
- **IDE:** Integrated Development Environment
- **API:** Application Programming Interface
- **JSON:** JavaScript Object Notation
- **UML:** Unified Modelling Language.
- **UX:** User Experience
- **URL:** Uniform Resource Locator

## **Abbreviations**

- **App:** Mobile Application

## **1.4 Mobile Application Scope**

iSport has been developed for those who love sports, with the aim to unify under one application all the services on the market. In this way we want to give an ongoing service to the end user, without having to browse multiple applications to achieve the same result.

In particular, the application will be divided into three screens:

- **News**
- **Live**
- **Bet**

- **Chat**

In the "News" section there will be the daily sport news displayed with a preview image and a small description. Moreover, by pressing on the single news you can read the complete article.

In the "Live" section there will be all current matches with the final scores if already completed or the current one if still in progress. Pressing on the single game, the user will consult all the related information such as the markers and goal time, cards, training and statistics.

In the "Bet" section there will be the shares related to the daily football matches. By pressing on the single one the user will bet on the winning game composing a ticket; once the process is completed the application will calculate the potential winnings based the bet amount.

In the "Chat" section you will connect to a global room where you can talk to other users who are using the application and exchange comments and opinions.

## **1.5 Framework**

The development of iSport was achieved through the use of native iOS SDKs, in particular by using the Swift programming language. This choice allowed greater control of system resources and access to system services, otherwise not possible if using cross-platform frameworks such as PhoneGap or React Native. The purpose is to implement different functionalities and integration with other sites.

## **1.6 Functional Requirements**

The product provides to users a simple and user-friendly interface to:

1. View news previews
2. Read the complete article
3. View football match results real time
4. Display goal-scores
5. Display booking (caution)
6. Display team playing the match
7. Display statistics
8. Display game share
9. Compose your ticket

10. Display the potential winnings of the ticket
11. Share news on Facebook
12. Save news into favorites section
13. Chat with other users
14. Log in

## 1.7 Non Functional Requirements

The application must be able to:

- Run both on phone and tablet (only if they have an iOS).
- Work without requiring user sensitive data and services that may require a user cost (such as calls or SMS).
- Occupy the entire screen available.
- Keep preferences and status at every start.

## 1.8 Assumptions, Dependencies and Constraints

### Constraints

- **Hardware limitations:** our application runs on every mobile device like smartphones and tablets. Therefore, as the App consumes a low amount of RAM, the only hardware constraint for the users is to have a mid-range device. (for instance iPhone 5 or better).
- **Parallel operations:** the application must be able to handle multiple parallel requests with high reactivity.

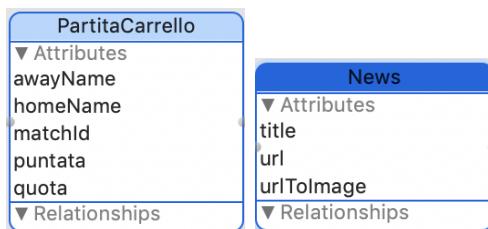
### Assumptions and Dependencies

- **Internet Connection:** the device used by the users dispose of an internet connection and a sufficient bandwidth to use the application.
- **No privileged users:** there are no priviled users or administrators with particular functions.
- **No user connections:** every user is independent from the others.
- **API availability:** the API provided by third part's services are always available.
- **OS Permission Granted:** the user will always grant to his OS's device the permission to access to all the needed services.

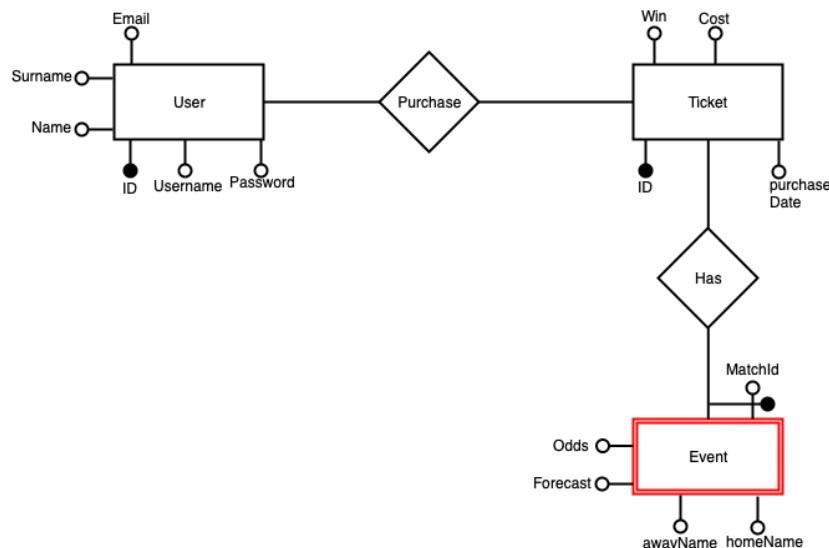
# 2 Architecture

## 2.1 Database

Since the application receives all the necessary data from external services through the API, the only data that need to be saved are the games that make up a ticket. Moreover, since there is no interaction between the different users, the data is saved locally using the Core Data present in the iOS SDK.



In order to purchase the ticket, we used an online database through the Firebase platform. The E / R model used is the follow:



To guarantee privacy, the access to the Database is regulated by the following rules:

```
1 {
2   "rules": {
3     "Schedina": {
4       "$uid": {
5         ".read": "$uid === auth.uid",
6         ".write": "$uid === auth.uid"
7       }
8     }
9   }
10 }
```

## 2.2 Client

For the implementation of the application we have chosen a mobile back-end, that is a client architecture. This choice was made mainly because the application does not interface with other users and because for various services it uses third-party APIs. Communication with third-party services is based on HTTPS REST requests, in particular through GET requests.

The client uses the traditional MVC pattern:

- Model: this package contains all the classes representing data to be shown to the single user, taken by the Controller and published by the View.
- View: this package contains all the components that display data to the user and interact with him.
- Controller: this package contains all the objects in charge to interact between one or more view objects of the application and one or more model objects.

### 3 Use case functional requirements analysis

This section describes how actors can interact with iSport in order to use all the features implemented in the app. The focus of this part is on the front-end and we show the operations that can be performed by the actors without taking care of the system architecture behind the app.

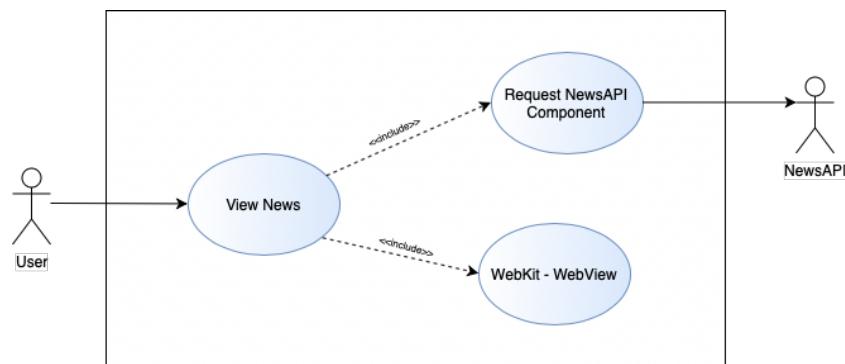


Figure 3.1: Use case related to visualization and handling news

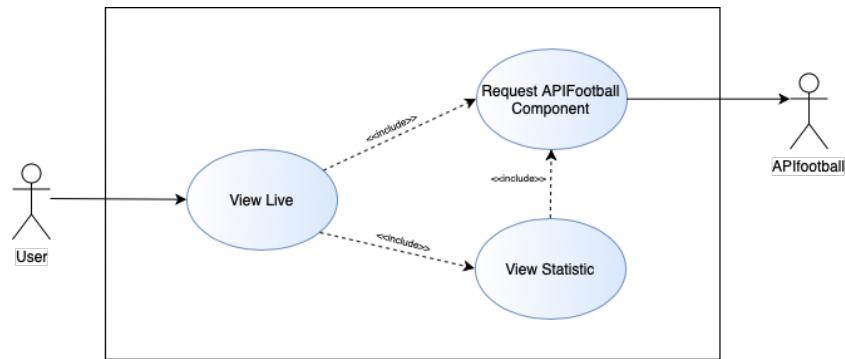


Figure 3.2: Use case related to visualization and handling results

### 3 Use case functional requirements analysis

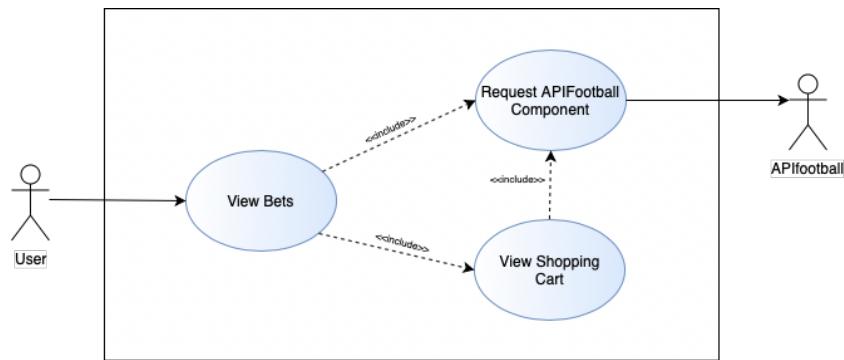


Figure 3.3: Use case related to visualization and handling odds

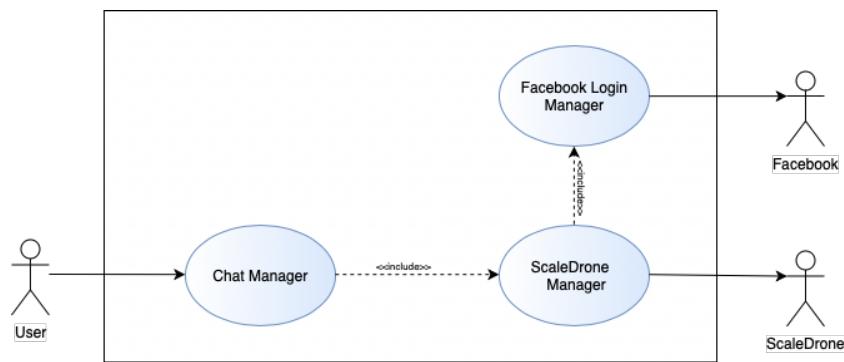


Figure 3.4: Use case related to chat service

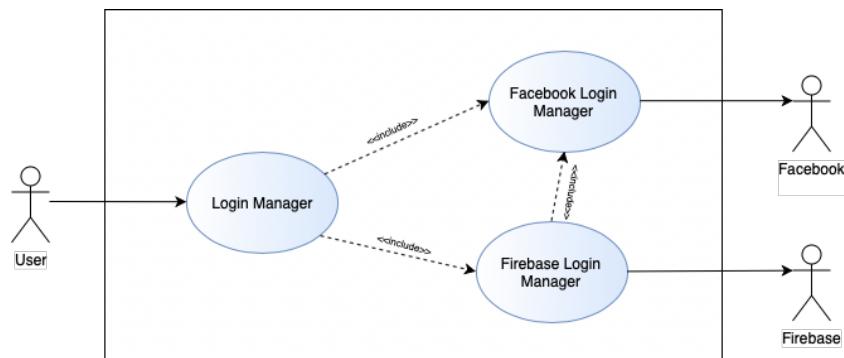


Figure 3.5: Use case related to log in activity

### 3 Use case functional requirements analysis

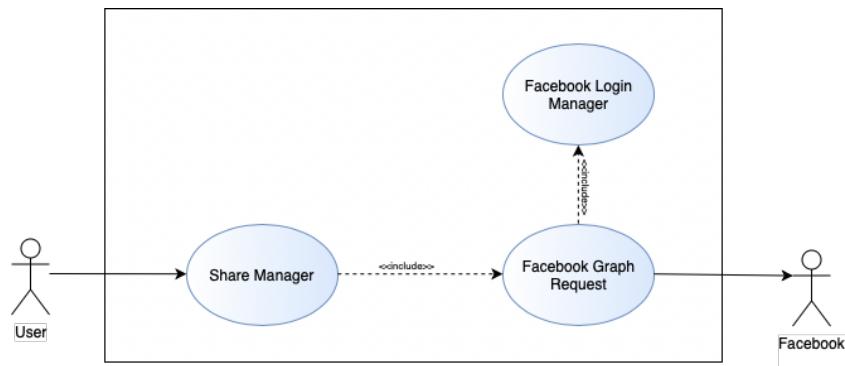


Figure 3.6: Use case related to news sharing

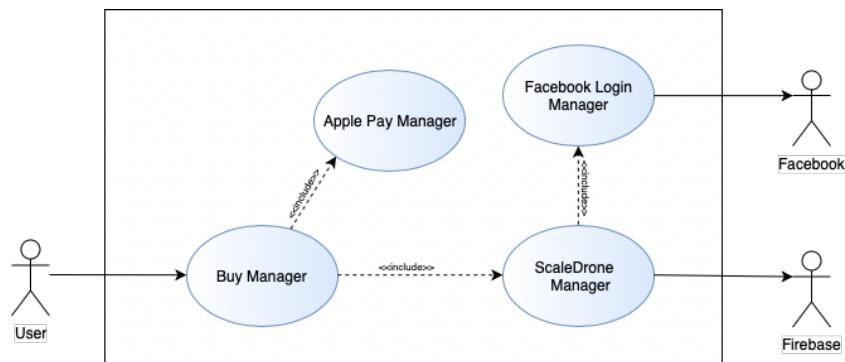


Figure 3.7: Use case related to ticket purchase

### 3 Use case functional requirements analysis

#### Login

<b>Name</b>	Login
<b>Actor</b>	Guest
<b>Entry Condition</b>	Guest with login credentials
<b>Goal</b>	14
<b>Event Flow</b>	<ul style="list-style-type: none"> <li>• The user opens the application</li> <li>• The user presses "Login" located in the "Side Menu"</li> <li>• The user is redirected on Facebook to enter credentials</li> <li>• The app logs in through Facebook</li> <li>• The app login into Firebase using Facebook account</li> </ul>
<b>Exit condition</b>	Actor becomes User
<b>Exceptions</b>	The user is not connected to the network or he hasn't a Facebook account.

### 3 Use case functional requirements analysis

#### Logout

<b>Name</b>	Logout
<b>Actor</b>	User
<b>Entry Condition</b>	Actor is logged in
<b>Goal</b>	14
<b>Event Flow</b>	<ul style="list-style-type: none"><li>• The user presses "Logout" located in the "Side Menu"</li><li>• The app logout from Facebook</li><li>• The app logout from Firebase</li></ul>
<b>Exit condition</b>	Actor is logged out and becomes Guest
<b>Exceptions</b>	The user is not connected to the network.

## View News

<b>Name</b>	View News
<b>Actor</b>	Guest
<b>Entry Condition</b>	The user downloaded the mobile app
<b>Goal</b>	1, 2
<b>Event Flow</b>	<ul style="list-style-type: none"> <li>• The user opens the app</li> <li>• The user presses on the "News" tab located to the "Side Menu"</li> <li>• The app shows a selection of the most important news</li> <li>• The user presses the news to read the whole article</li> <li>• The user presses "Done" to finish reading</li> </ul>
<b>Exit condition</b>	The user read the news of interest.
<b>Exceptions</b>	The user is not connected to the network so he cannot send a request to read the article. The news has been released from the origin website, but not from the database.

### 3 Use case functional requirements analysis

#### Share News

<b>Name</b>	Share News
<b>Actor</b>	User
<b>Entry Condition</b>	The user logged in correctly
<b>Goal</b>	11
<b>Event Flow</b>	<ul style="list-style-type: none"> <li>• The user opens the app</li> <li>• The user presses the "News" tab located in the "Side Menu"</li> <li>• The app shows a selection of the most important news</li> <li>• The user presses on the "Share" function</li> <li>• The user writes the message he wants to attach to the shared post</li> <li>• The user presses "Publish"</li> </ul>
<b>Exit condition</b>	The user shares on Facebook the news of interest.
<b>Exceptions</b>	The user is not connected to the network so he cannot send a request to share the article. The news has been released from the origin website, but not from the database or the connection is weak.

### 3 Use case functional requirements analysis

#### Save News

<b>Name</b>	Save News
<b>Actor</b>	User
<b>Entry Condition</b>	The user downloaded the mobile app
<b>Goal</b>	12
<b>Event Flow</b>	<ul style="list-style-type: none"> <li>• The user opens the app</li> <li>• The user presses the "News" tab located in the "Side Menu"</li> <li>• The app shows a selection of the most important news</li> <li>• The user presses on "Bookmark" to save the news</li> <li>• The app saves the information in the local database</li> </ul>
<b>Exit condition</b>	The user saves the news for a following reading.
<b>Exceptions</b>	The user is not connected to the network, so he cannot send a request to save the article. The news has been released from the origin website, but not from the database.

### 3 Use case functional requirements analysis

#### See Saved News

<b>Name</b>	See Saved News
<b>Actor</b>	User
<b>Entry Condition</b>	The user saved the news
<b>Goal</b>	1, 2, 12
<b>Event Flow</b>	<ul style="list-style-type: none"> <li>• The user opens the app</li> <li>• The user presses the "News" tab located in the "Side Menu"</li> <li>• The user presses on "Bookmark" in the Navigation Bar to see saved news</li> <li>• The app shows a list of the saved news</li> </ul>
<b>Exit condition</b>	The user read the news of interest.
<b>Exceptions</b>	The user is not connected to the network so he cannot send a request to read the article. The news has been released from the origin website, but not from the database.

## See Matches

<b>Name</b>	See Matches
<b>Actor</b>	Guest
<b>Entry Condition</b>	The user downloaded the app
<b>Goal</b>	3
<b>Event Flow</b>	<ul style="list-style-type: none"> <li>• The user opens the app</li> <li>• The user presses the "Live" tab located in the "Side Menu"</li> <li>• The app shows a list of matches</li> </ul>
<b>Exit condition</b>	The user saw scores.
<b>Exceptions</b>	The user is not connected to the network, so he cannot send a request to obtain the final scores.

## View Lineup

<b>Name</b>	View Lineup
<b>Actor</b>	Guest
<b>Entry Condition</b>	The user downloaded the app
<b>Goal</b>	3, 6
<b>Event Flow</b>	<ul style="list-style-type: none"> <li>• The user opens the app</li> <li>• The user presses the "Live" tab located in the "Side Menu"</li> <li>• The app shows a list of daily matches</li> <li>• The user chooses the match to obtain the requested information</li> <li>• The user presses the button showing the game field</li> </ul>
<b>Exit condition</b>	The user saw lineups for the requested match
<b>Exceptions</b>	The user is not connected to the network, so he cannot send a request to obtain results or the connection could be weak.

## View Goal Scorer

<b>Name</b>	View Goal Scorer
<b>Actor</b>	Guest
<b>Entry Condition</b>	The user downloaded the app
<b>Goal</b>	3, 4
<b>Event Flow</b>	<ul style="list-style-type: none"> <li>• The user opens the app</li> <li>• The user presses the "Live" tab located in the "Side Menu"</li> <li>• The app shows a list of daily matches</li> <li>• The user chooses the match to obtain the requested information</li> <li>• The user presses the button showing a ball</li> </ul>
<b>Exit condition</b>	The user saw the goal scorers of the requested match.
<b>Exceptions</b>	The user is not connected to the network, so he cannot send a request to obtain results or the connection could be weak.

## View Match Statistics

<b>Name</b>	View Match Statistics
<b>Actor</b>	Guest
<b>Entry Condition</b>	The user downloaded the app
<b>Goal</b>	3, 7
<b>Event Flow</b>	<ul style="list-style-type: none"> <li>• The user opens the app</li> <li>• The user presses the "Live" tab located in the "Side Menu"</li> <li>• The app shows a list of daily matches</li> <li>• The user chooses the match to obtain the requested information</li> <li>• The user presses the button showing a chart</li> </ul>
<b>Exit condition</b>	The user saw the match statistics.
<b>Exceptions</b>	The user is not connected to the network, so he cannot send a request to obtain results or the connection could be weak.

## **View Match Booking (Cautions)**

<b>Name</b>	View Match Booking (Cautions)
<b>Actor</b>	Guest
<b>Entry Condition</b>	The user downloaded the app
<b>Goal</b>	3, 5
<b>Event Flow</b>	<ul style="list-style-type: none"> <li>• The user opens the app</li> <li>• The user presses the "Live" tab located in the "Side Menu"</li> <li>• The app shows a list of daily matches</li> <li>• The user chooses the match to obtain the requested information</li> <li>• The user presses the button showing a card</li> </ul>
<b>Exit condition</b>	The user saw booked and expelled players.
<b>Exceptions</b>	The user is not connected to the network, so he cannot send a request to obtain results or the connection could be weak.

### 3 Use case functional requirements analysis

#### **View Odds**

<b>Name</b>	View Odds
<b>Actor</b>	Guest
<b>Entry Condition</b>	The user downloaded the app
<b>Goal</b>	8
<b>Event Flow</b>	<ul style="list-style-type: none"> <li>• The user opens the app</li> <li>• The user presses the “Bet” tab located in the ”Side Menu”</li> <li>• The app shows a list of principle odds of the daily matches</li> </ul>
<b>Exit condition</b>	The user saw the odds of the daily matches.
<b>Exceptions</b>	The user is not connected to the network, so he cannot send a request to obtain results or the connection could be weak.

### 3 Use case functional requirements analysis

#### View Ticket

<b>Name</b>	View Ticket
<b>Actor</b>	Guest
<b>Entry Condition</b>	The user downloaded the app
<b>Goal</b>	9
<b>Event Flow</b>	<ul style="list-style-type: none"><li>• The user opens the app</li><li>• The user presses the “Bet” tab located in the ”Side Menu”</li><li>• The user presses the button showing a cart</li></ul>
<b>Exit condition</b>	The user sees the ticket.
<b>Exceptions</b>	The user is not connected to the network, so he cannot send a request to obtain results or the connection could be weak.

## Add Match To The Ticket

<b>Name</b>	Add Match To The Ticket
<b>Actor</b>	Guest
<b>Entry Condition</b>	The user downloaded the app
<b>Goal</b>	9
<b>Event Flow</b>	<ul style="list-style-type: none"> <li>• The user opens the app</li> <li>• The user presses the “Bet” tab located in the ”Side Menu”</li> <li>• The app shows a list of principle odds of the daily matches</li> <li>• The user presses on the odd and add the forecast to the ticket</li> </ul>
<b>Exit condition</b>	The user added a forecast to the ticket.
<b>Exceptions</b>	The user is not connected to the network, so he cannot send a request to obtain results or the connection could be weak.

### 3 Use case functional requirements analysis

#### Delete a Match

<b>Name</b>	Delete a Match
<b>Actor</b>	Guest
<b>Entry Condition</b>	The user downloaded the app
<b>Goal</b>	9
<b>Event Flow</b>	<ul style="list-style-type: none"> <li>• The user opens the app</li> <li>• The user presses the “Bet” tab located in the ”Side Menu”</li> <li>• The user presses the button showing a cart</li> <li>• The user swipes to the left to delete the match</li> </ul>
<b>Exit condition</b>	The user deleted a forecast from the ticket
<b>Exceptions</b>	None

## Calculate The Potential Winning

<b>Name</b>	Calculate The Potential Winning
<b>Actor</b>	Guest
<b>Entry Condition</b>	The user downloaded the app
<b>Goal</b>	10
<b>Event Flow</b>	<ul style="list-style-type: none"> <li>• The user opens the app</li> <li>• The user presses the “Bet” tab located in the ”Side Menu”</li> <li>• The user presses the button showing a cart</li> <li>• The user selects the amount of money he wants to bet</li> <li>• The user presses on “Done”</li> </ul>
<b>Exit condition</b>	The user sees the potential winning
<b>Exceptions</b>	None

## Purchase ticket

<b>Name</b>	Purchase ticket
<b>Actor</b>	User
<b>Entry Condition</b>	The user logged in correctly
<b>Goal</b>	9, 10
<b>Event Flow</b>	<ul style="list-style-type: none"> <li>• The user opens the app</li> <li>• The user presses the “Bet” tab located in the ”Side Menu”</li> <li>• The user presses the button showing a cart</li> <li>• The user selects the amount of money he wants to bet</li> <li>• The user presses on ”Buy”</li> </ul>
<b>Exit condition</b>	The user completes the purchase.
<b>Exceptions</b>	Absent connection to the network, the user cannot complete the purchase. The user didn't log correctly or he didn't add matches to the ticket.

## View Purchased Tickets

<b>Name</b>	View Purchased Tickets
<b>Actor</b>	User
<b>Entry Condition</b>	The user logged in correctly
<b>Goal</b>	9, 10
<b>Event Flow</b>	<ul style="list-style-type: none"> <li>• The user opens the app</li> <li>• The user presses the “Bet” tab located in the ”Side Menu”</li> <li>• The user presses on “chronology” in the NavBar</li> <li>• The app downloads the information about all purchased tickets</li> <li>• The user presses on the corresponding cell of the ticket he wants to see</li> <li>• The app loads the ticket and the user sees all the details</li> </ul>
<b>Exit condition</b>	The user sees a purchased ticket previously.
<b>Exceptions</b>	The user is not connected to the network, so he cannot send a request to obtain results or the connection could be weak. The user didn't log correctly or he didn't buy a ticket.

### 3 Use case functional requirements analysis

#### Chat

<b>Name</b>	Chat
<b>Actor</b>	User
<b>Entry Condition</b>	The user logged in correctly
<b>Goal</b>	13
<b>Event Flow</b>	<ul style="list-style-type: none"> <li>• The user opens the app</li> <li>• The user presses the “Chat” tab located in the ”Side Menu”</li> <li>• The app logs into Scaledrone’s room using Facebook credentials</li> <li>• The user writes messages in the text field</li> <li>• The user presses send</li> <li>• The app loads all the messages of the room</li> </ul>
<b>Exit condition</b>	The user interacts with other people.
<b>Exceptions</b>	The user is not connected or he has not a Facebook account.

# 4 Sequence Diagram

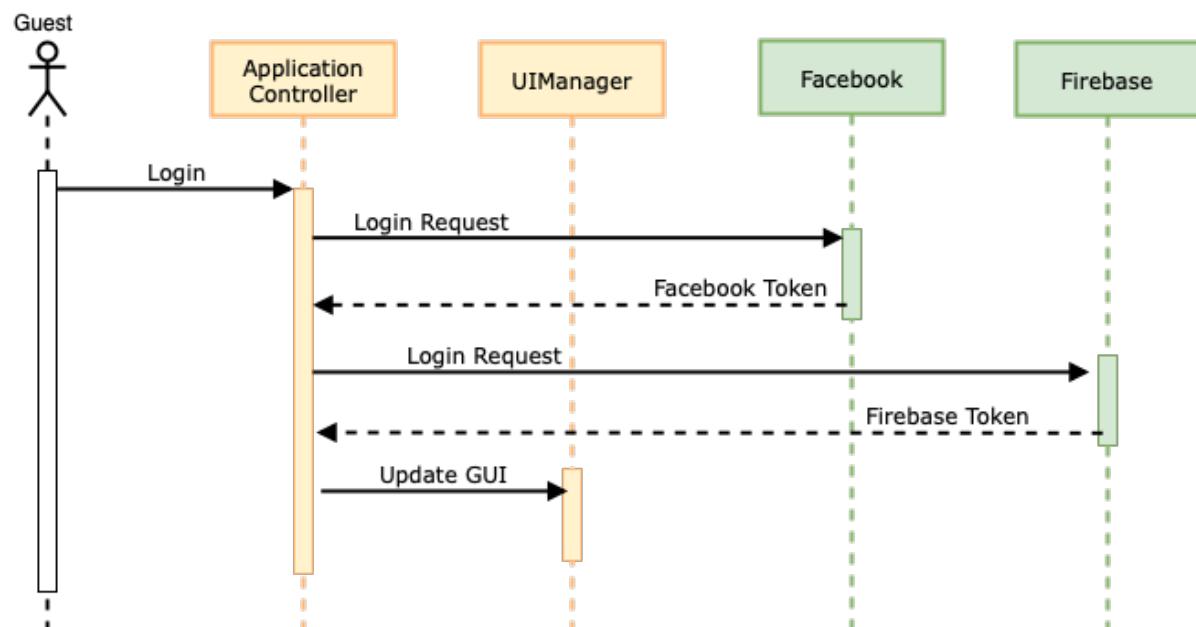
In order to explain our work and facilitate further implementations we have decided to show the logical flows aimed to create some features. The sequence diagrams will describe the interactions between the different parts of the system and the user.

## Login

The "Login" procedure starts when the user presses the button on the Side Menu. Once pressed, the user is redirected to the Facebook site to complete the registration. Immediately after logging on Facebook the application will log into the Firebase Database.

Once this procedure is completed, the system will update the GUI by enabling the sections available only to subscribers.

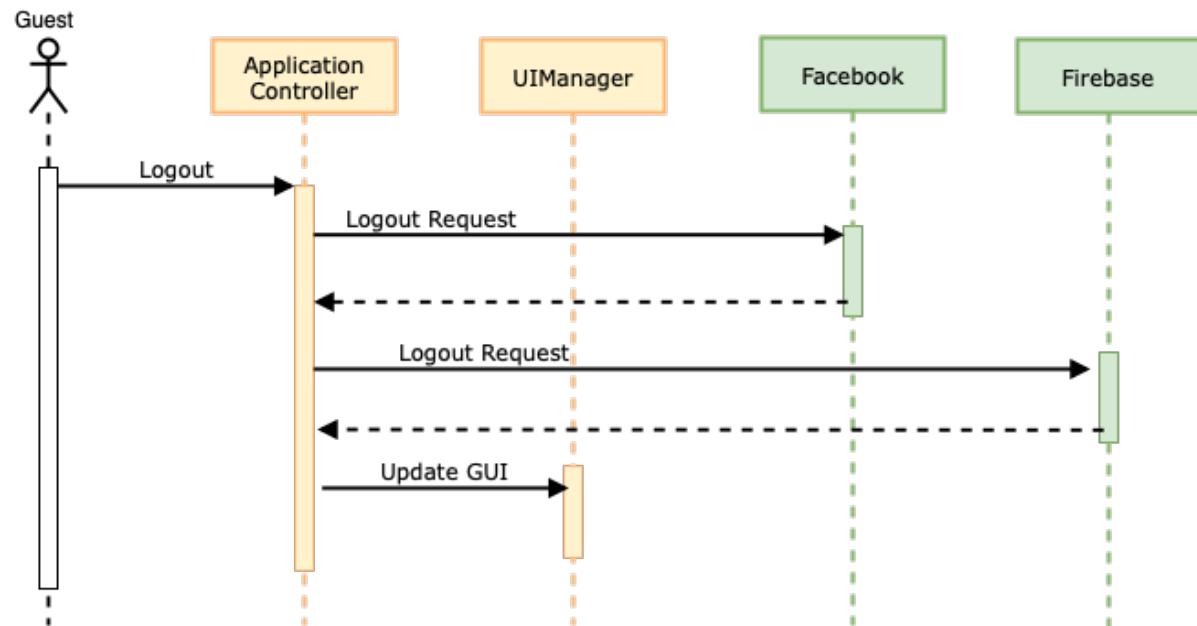
In the event of an error, the system will show an error message.



## Logout

The "Logout" procedure starts when the user presses the button on the Side Menu. Once pressed, a confirmation message will be displayed. In case of approval, the logout from Facebook and Firebase will be effective.

Now that this procedure is completed, the system will update the GUI by enabling sections not available for guests.



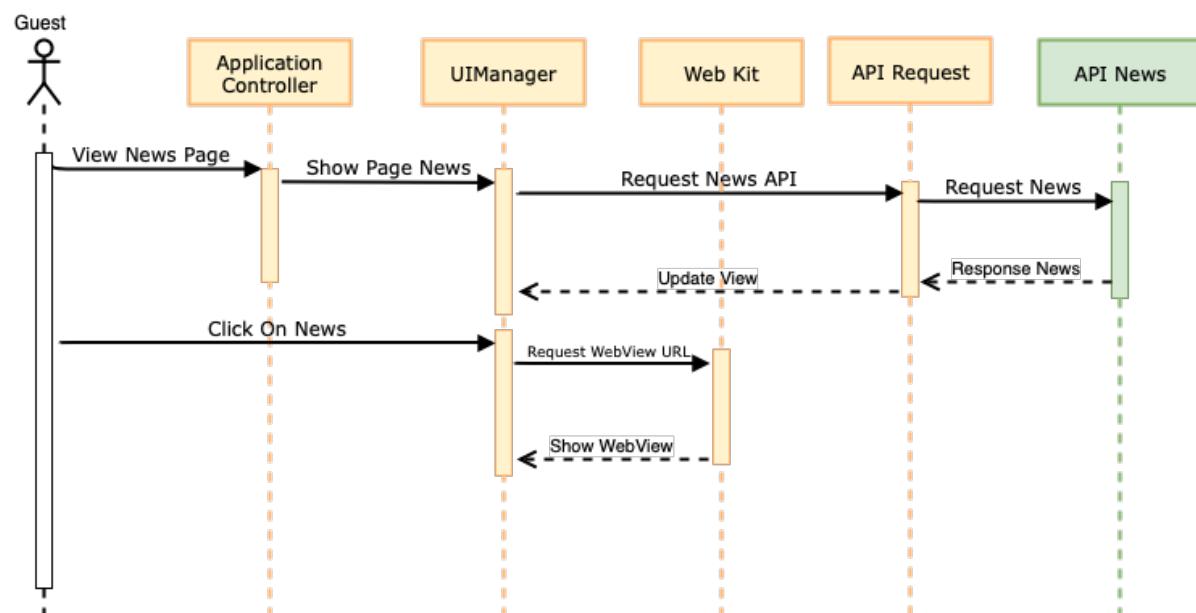
## View News

The "View News" procedure starts when the user opens the application or presses the "News" section in the Tab Navigator. The sequence diagram shows the normal procedure.

After the user activates the News section, the application will send a request to the "NewsAPI" service in order to obtain all the information related to the most important daily news.

Once this information is obtained, the Controller will create a UITableViewCell for each news and insert them into the TableView.

Furthermore and asynchronously, the application will download the preview images to allow the user to read the news headlines. When a cell is pressed the Controller will open a WebView addressed to the URL of the news in order to allow the user to read the complete article.

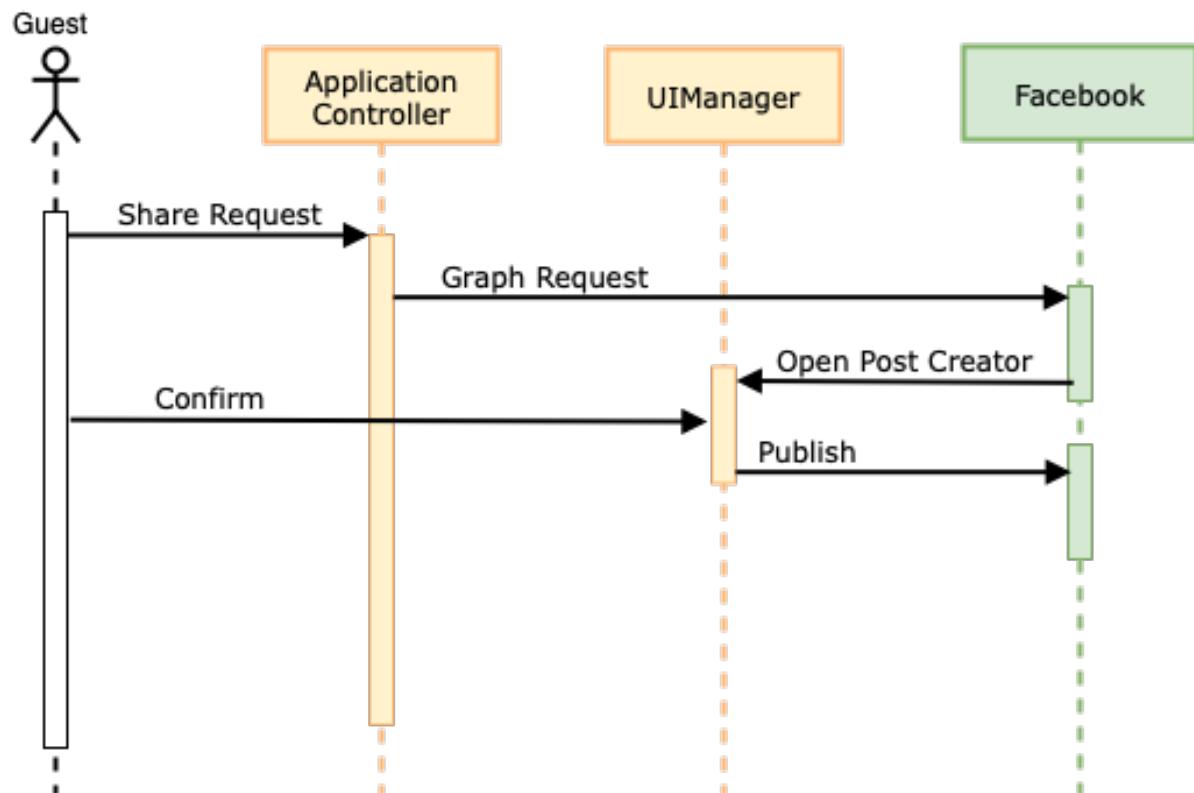


## Share News

The "Share News" procedure starts when the user opens the application or presses "News" on the Side Menu. The sequence diagram shows the normal procedure.

After the normal "View News" procedure the user should press the share button. At that point the application will make a request to the "Graph" API of Facebook to create the post with the message entered by the user.

To make this operation possible is necessary that the user is already authenticated otherwise the system will show an error notice.



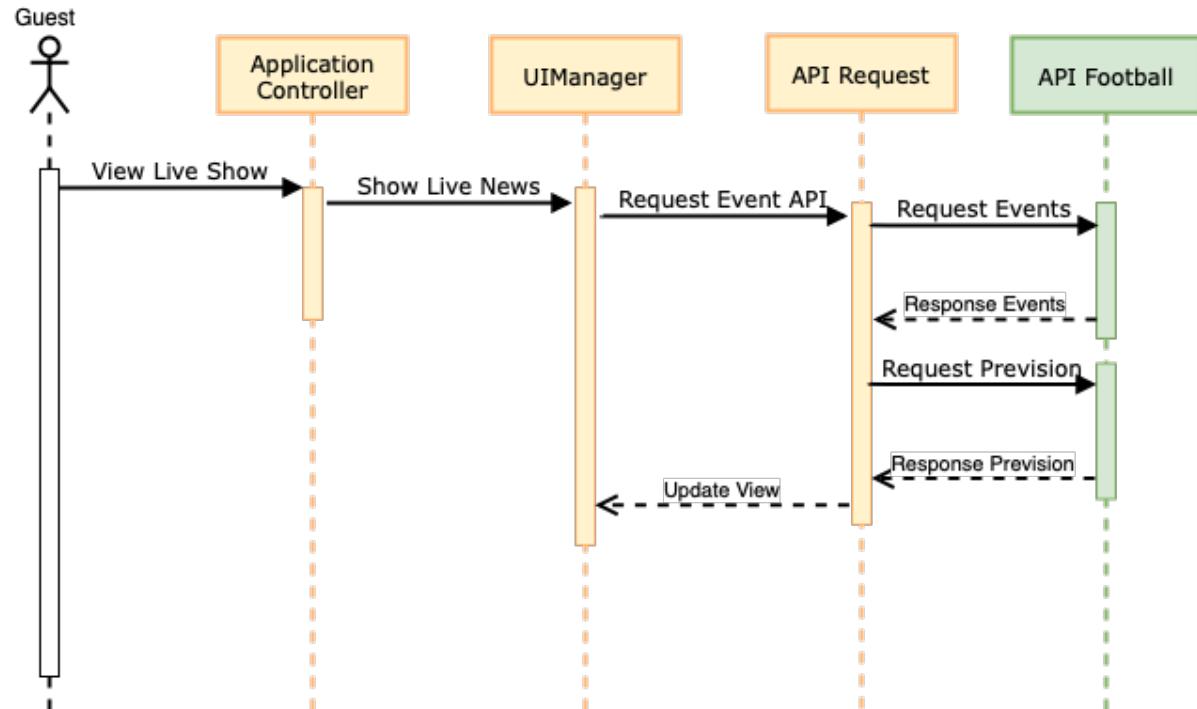
## View Match

The "View Match" procedure starts when the user presses the "Live" section of the Tab Bar Navigator.

After activating the "Live" section, the application will send a request to the "API-Football" service to obtain information about the daily matches and their forecasts.

When this information has been reached, the Controller will create a UITableViewCell for all the matches and insert them into the TableView.

At regular intervals the Controller will repeat these requests to keep the matches updated.

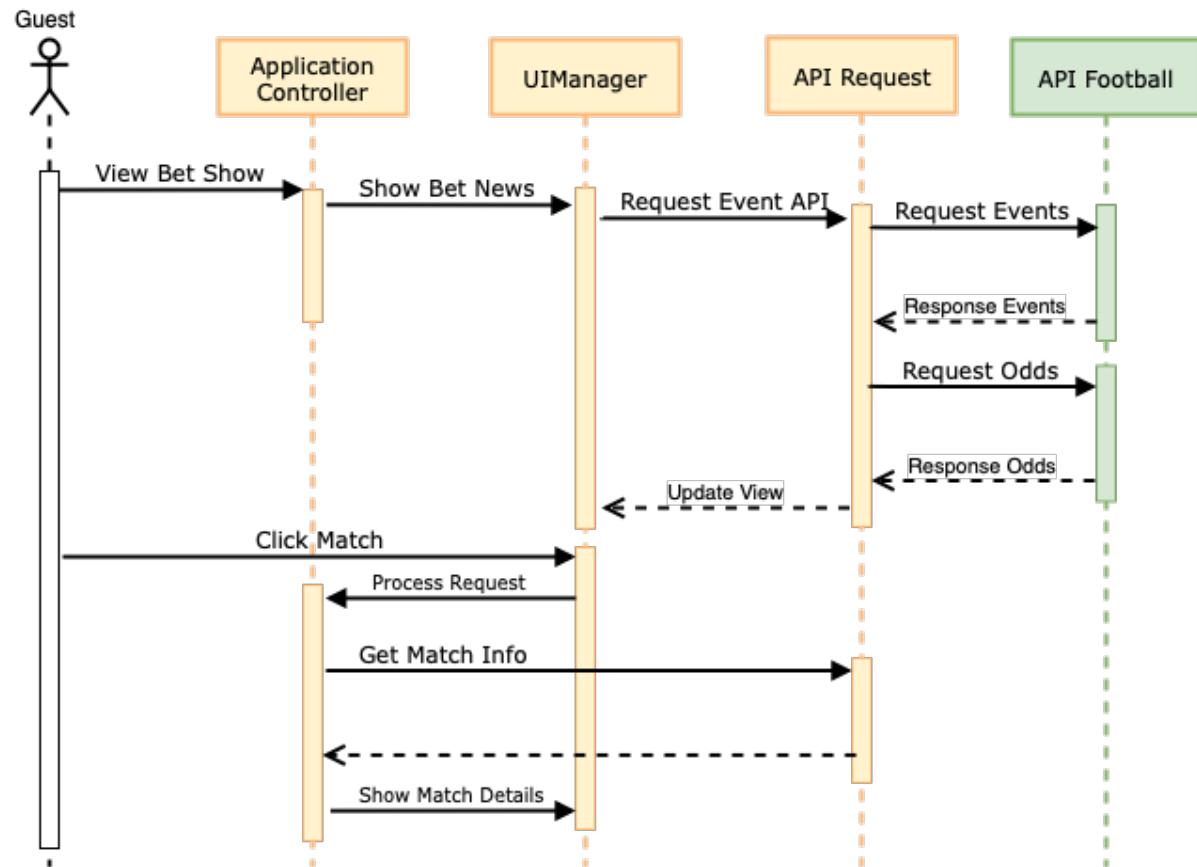


## View Match Details

The "View Match" procedure starts when the user is in the "Live" section and presses on a single match to know the details.

Now the Controller has access to the data for the specific match, previously downloaded from the "APIFootball" service and creates Views for every information.

After opening the detail page, the user can change the desired information through the relative buttons.

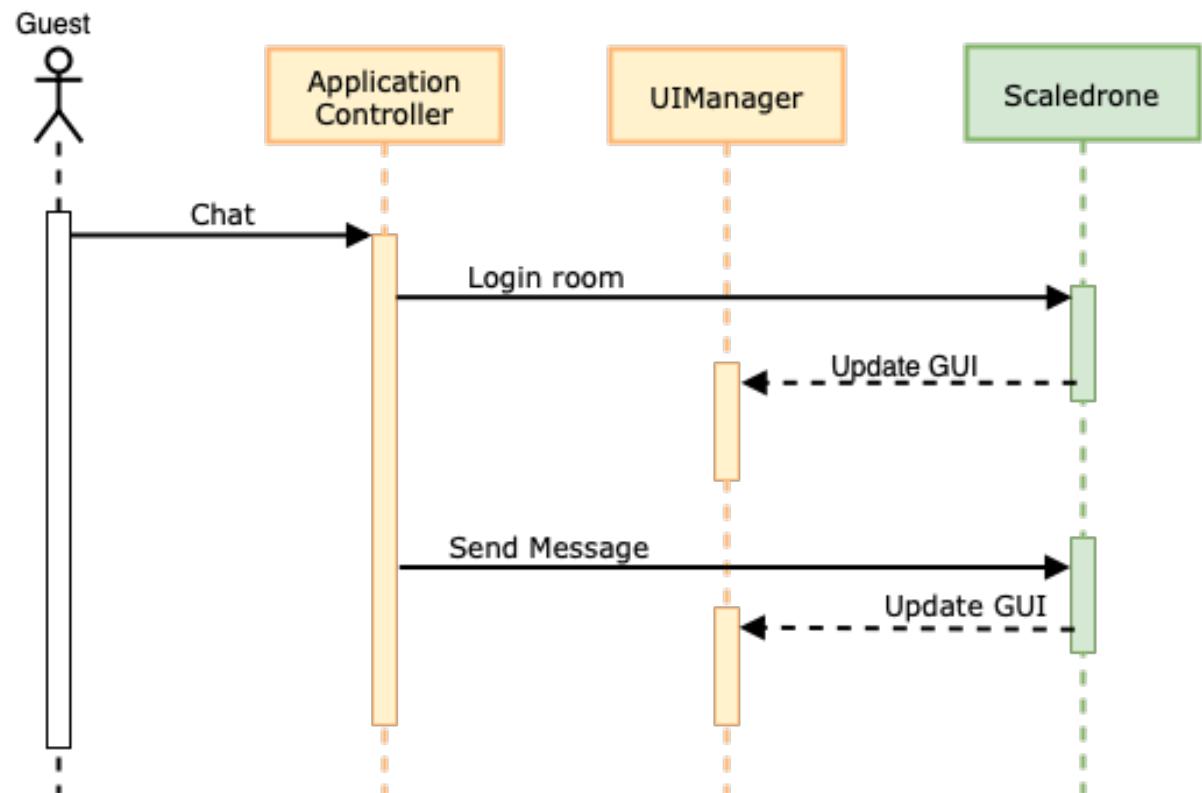


## Chat

The "Chat" procedure starts when the user presses the button on the Side Menu. Once the screen is loaded, the application will show all messages sent by other users in real time. The user can then write and send his message.

The application will also show Facebook profile images of other users next to the message sent.

To make this operation possible is necessary that the user is already authenticated otherwise the system will show an error notice.



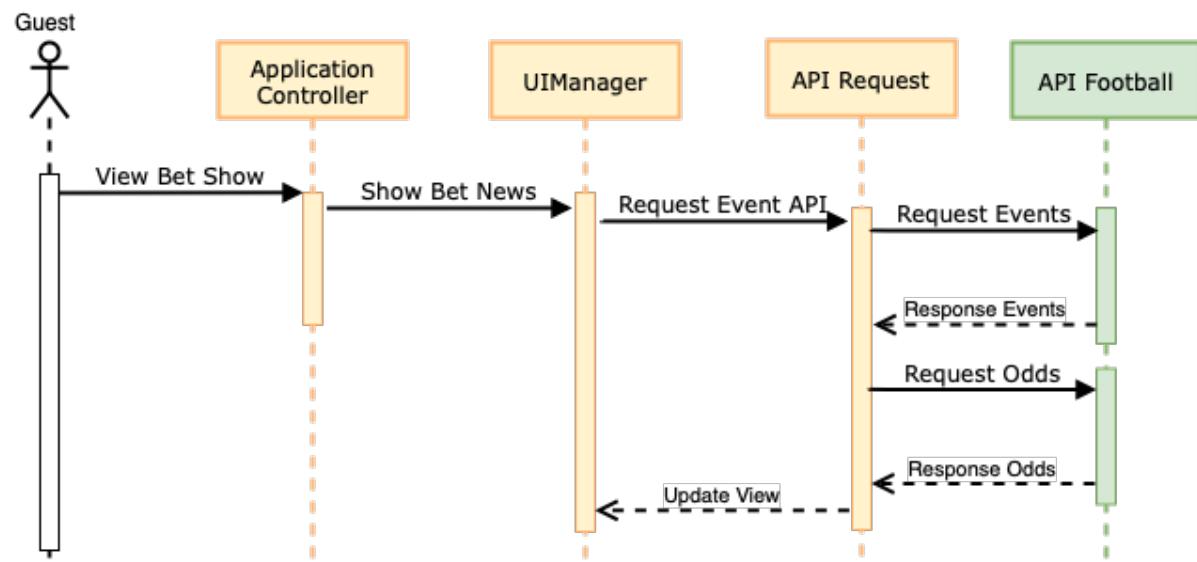
## View Odds

The "View Odds" procedure starts when the user presses the "Bet" section on the Tab Bar Navigator.

Activated the "Bet" section, the application will send a request to the "APIFootball" service to obtain information about the daily matches and the relative odds.

At this point, the Controller will create a UITableViewCell for each news and insert them into the TableView. In particular, we inserted three buttons (one for each prediction) with the relative odd indicated.

At regular intervals the Controller will repeat these requests to keep the matches updated.



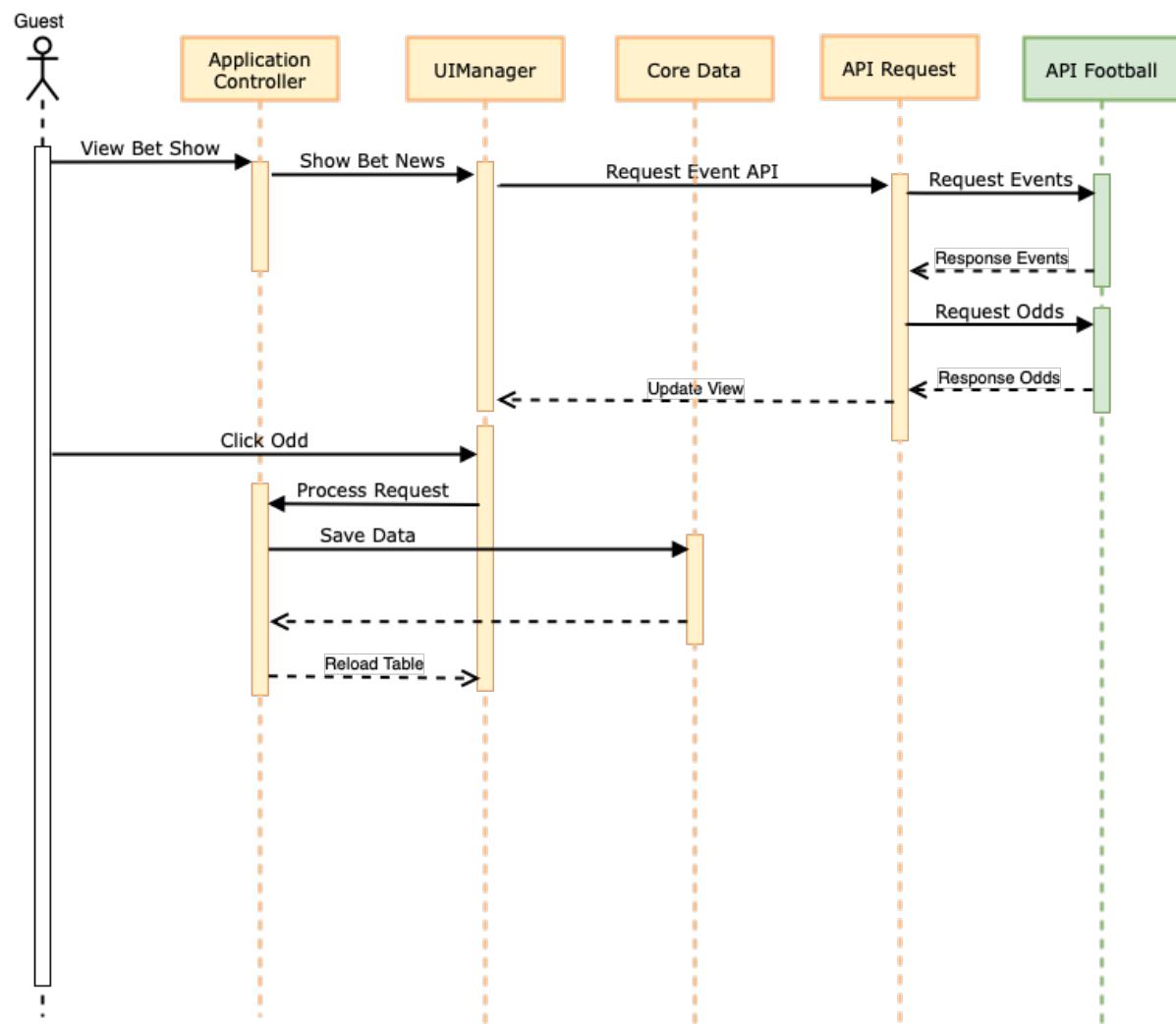
## Add Odd

The "Add Odd" procedure starts when the user is in the "Bet" section and presses on the prediction of the match he wants to bet on.

At that point the Controller has access to the data for the specific match, previously downloaded from the "APIFootball" service and create the data structure that will be saved in the Database.

Once the data structure is created, the Controller will check if the match is already present in the database; If so, it will update the element with the new forecast, otherwise it will create a new record.

After updating the DB the Controller will update the current View to show the change in a visual feedback.

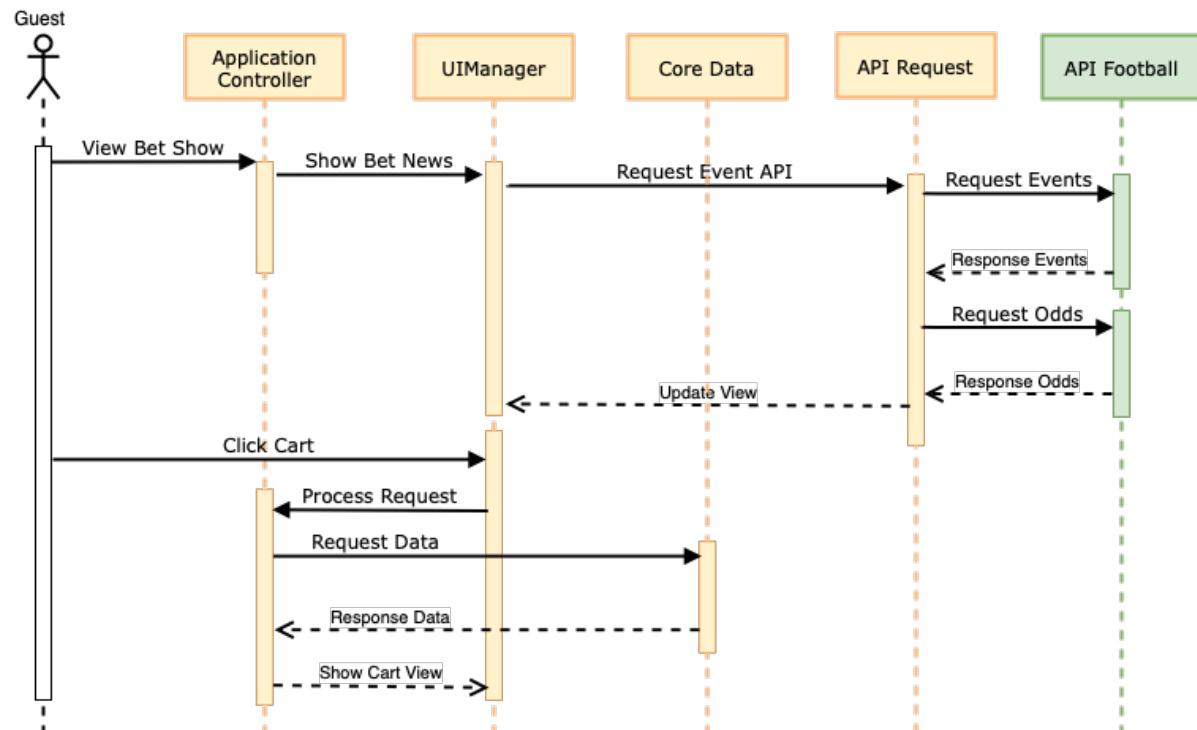


## View Cart

The "View Cart" procedure starts when the user is in the "Bet" section and presses the button showing a cart in the Navigation bar.

At that point, the Controller has access to the data in the Database and for each of them it will create a UITableViewCell to be inserted in the TableView.

The user can change the amount of money gambled by using the corresponding input. Once the amount has been updated, the Controller will calculate the potential winnings by multiplying all the odds with the amount played.

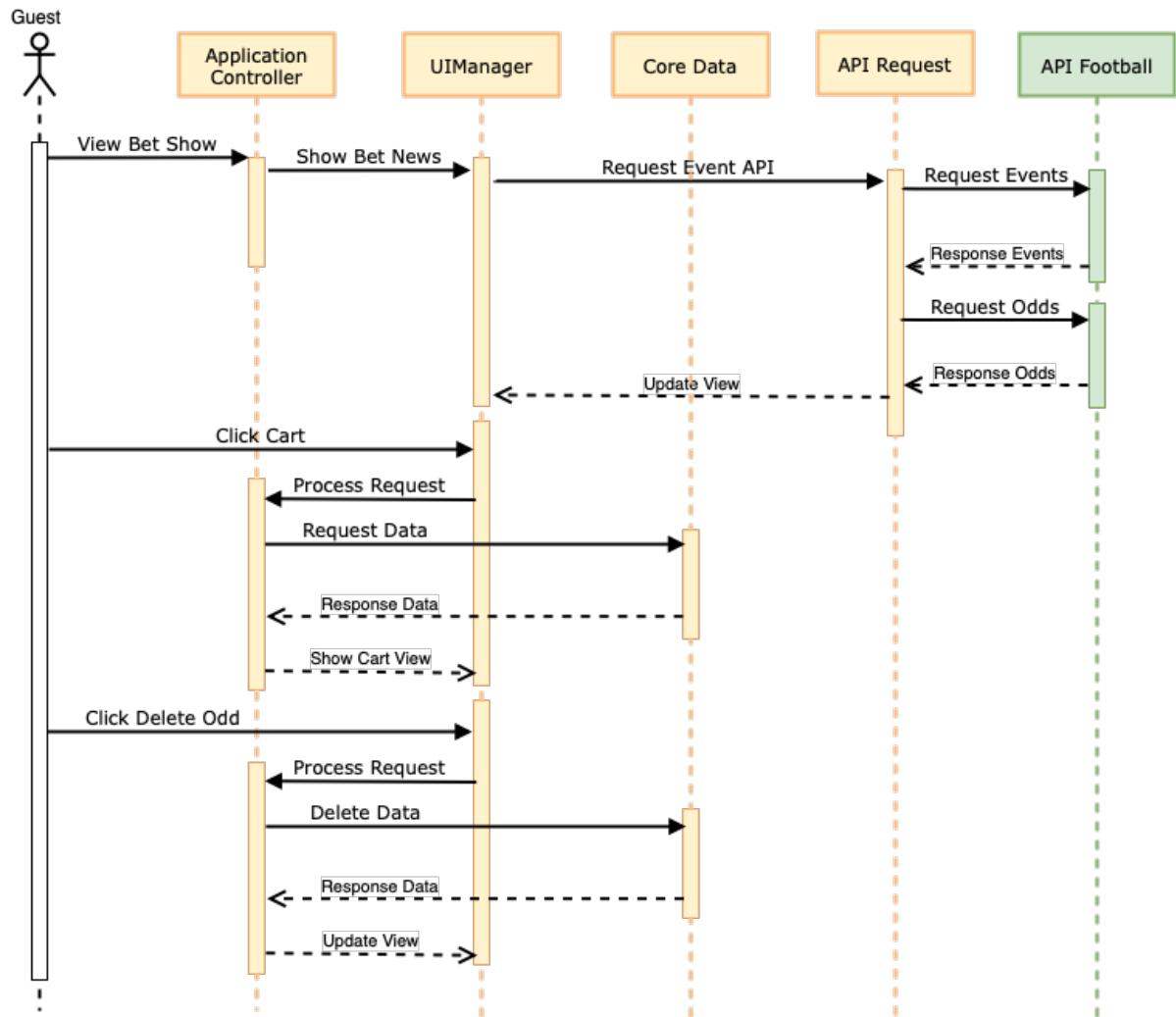


## Remove Odd

The procedure of "Remove Odd" begins when the user is in the "Bet" section and he is checking the cart.

In this section the user can delete each bet by swiping to the left with his finger.

At that point the Controller will delete the corresponding record from the Database and update the contents of the shopping cart also recalculating the new potential winnings.

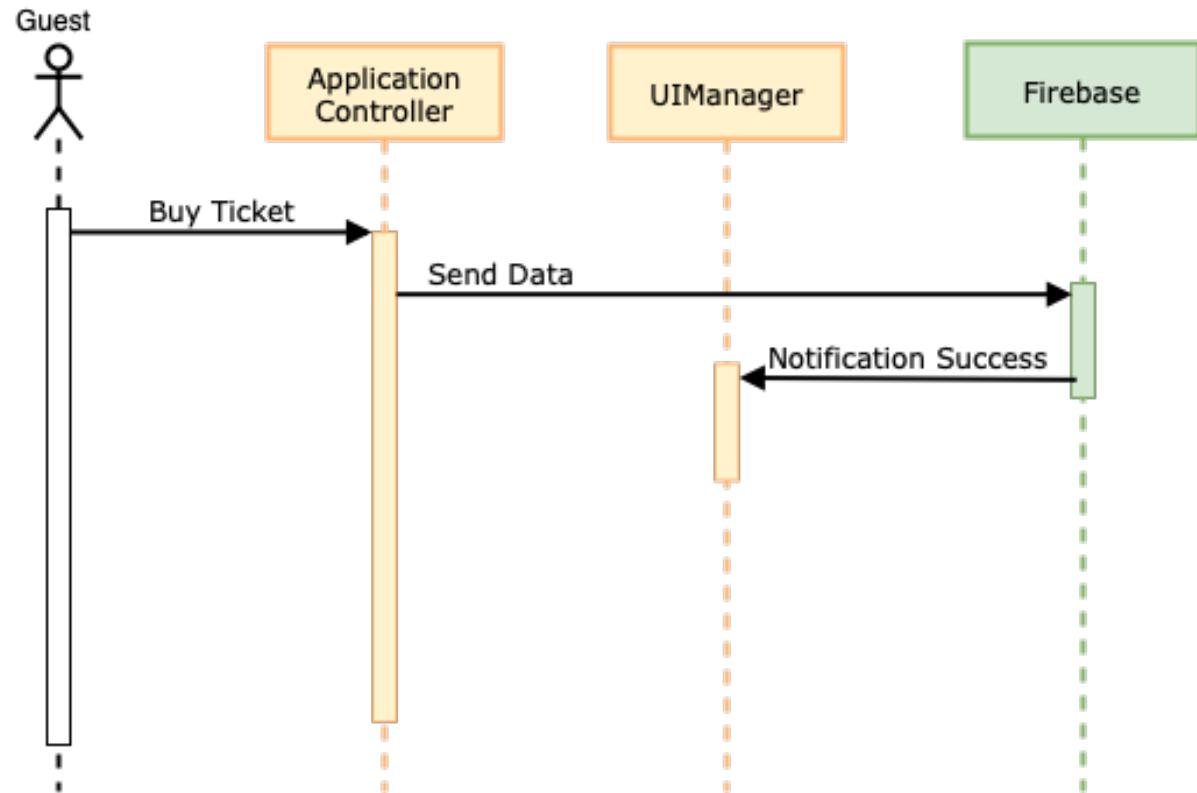


## Buy Ticket

When the user completes the composition of his ticket, the "Buy Ticket" procedure may begin.

By pressing the buy button, the application will check the correctness of the ticket and proceed with the purchase.

To make this operation possible is necessary that the user is already authenticated otherwise the system will show an error notice.

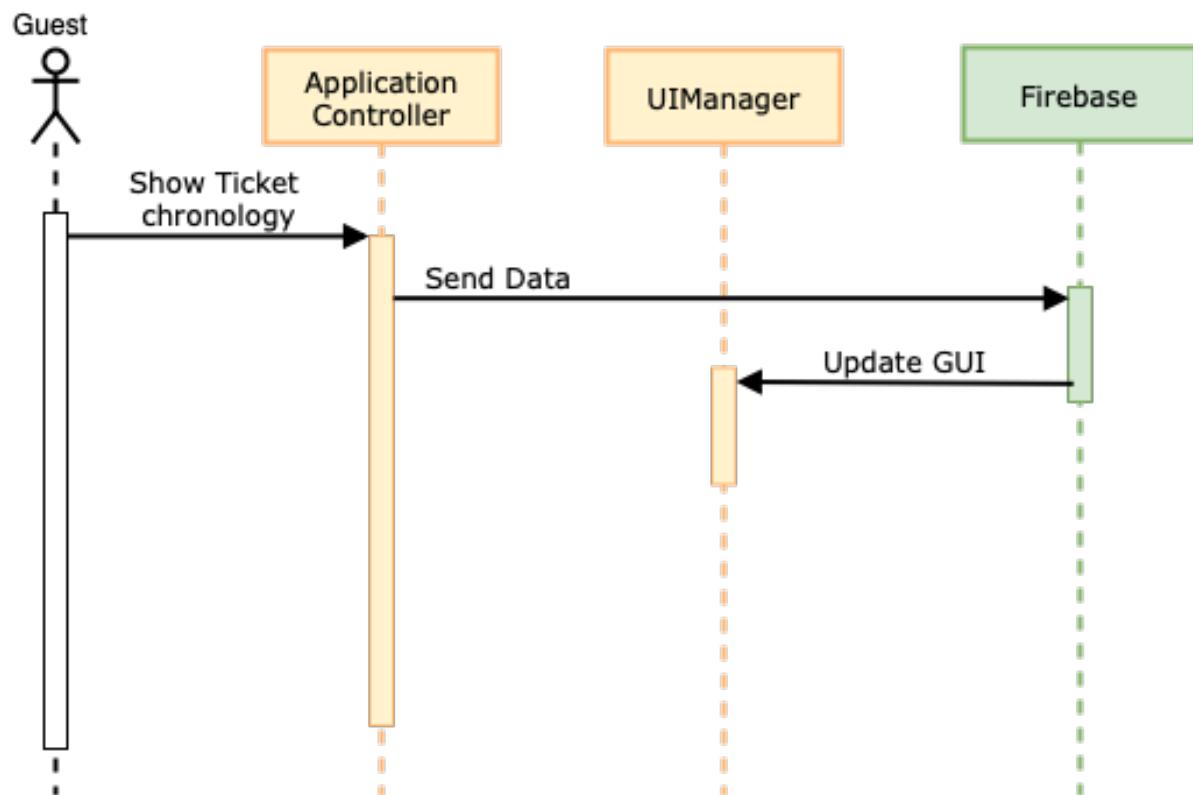


## Ticket Chronology

The "Ticket Chronology" procedure starts when the user is in the "Bet" section by pressing the button in the Side Menu.

Moreover, by pressing "History" on the NavBar the application access Firebase database showing all the tickets purchased previously by the user.

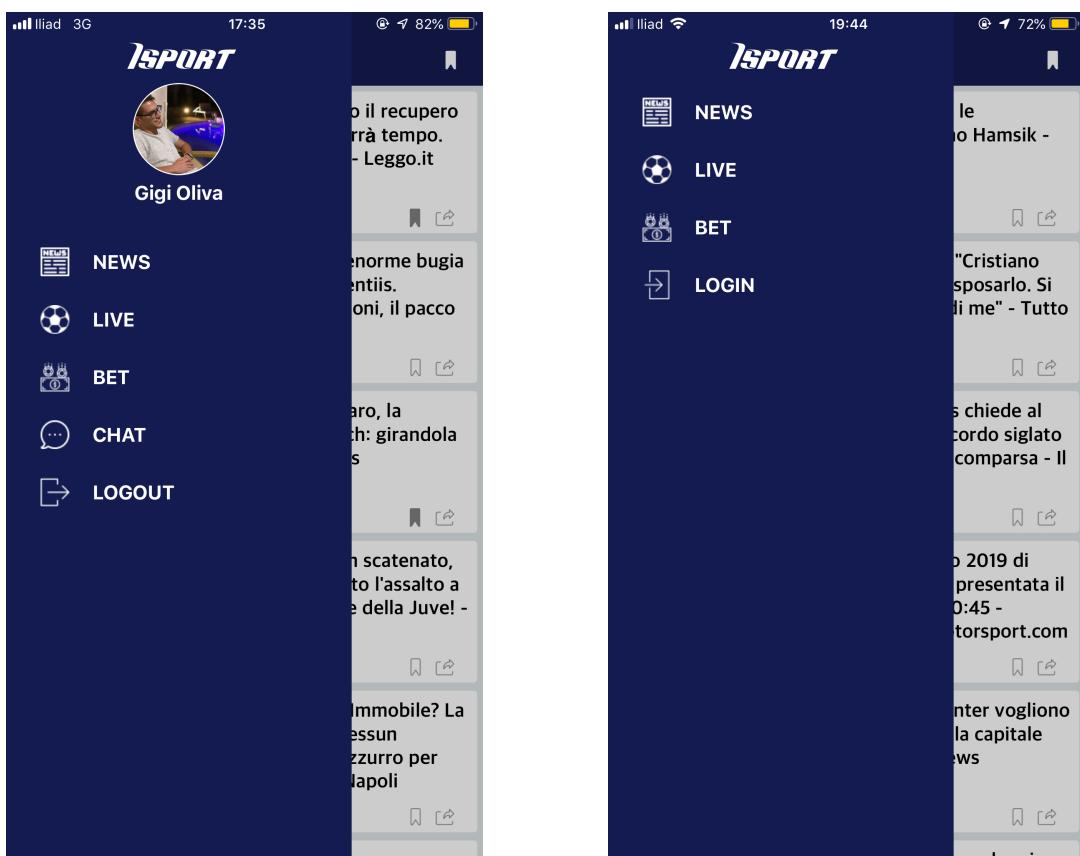
To make this procedure possible is necessary that the user is already authenticated otherwise the system will show an error notice.



# 5 User Interfaces

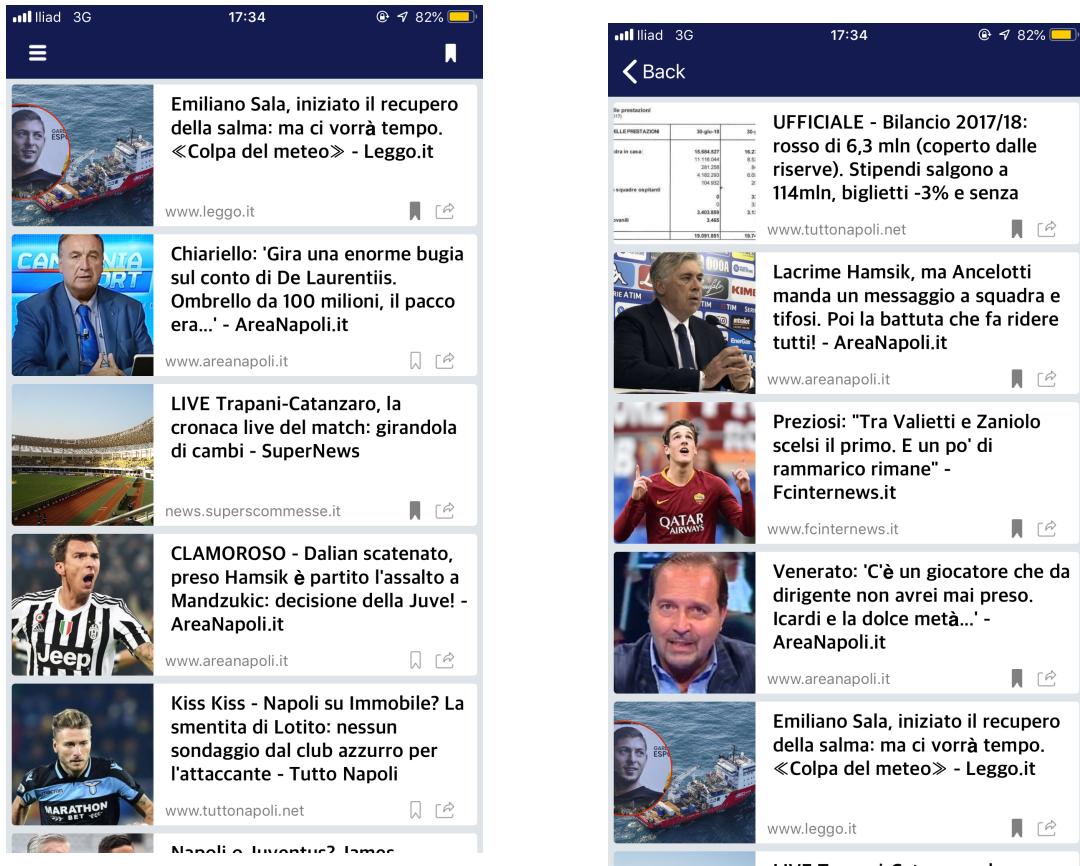
In this chapter will be discussed and displayed the user interface of each page highlighting the functionalities of the iSport application.

## Side Menu



- (a) This screen shows the Side Menu of an authenticated user  
(b) This screen shows the Side Menu of an unauthenticated user

## News



(a) This screen shows all the news along with the possibility to save or publish them

(b) This screen shows all the saved news

## 5 User Interfaces



(a) This screen shows the News sharing

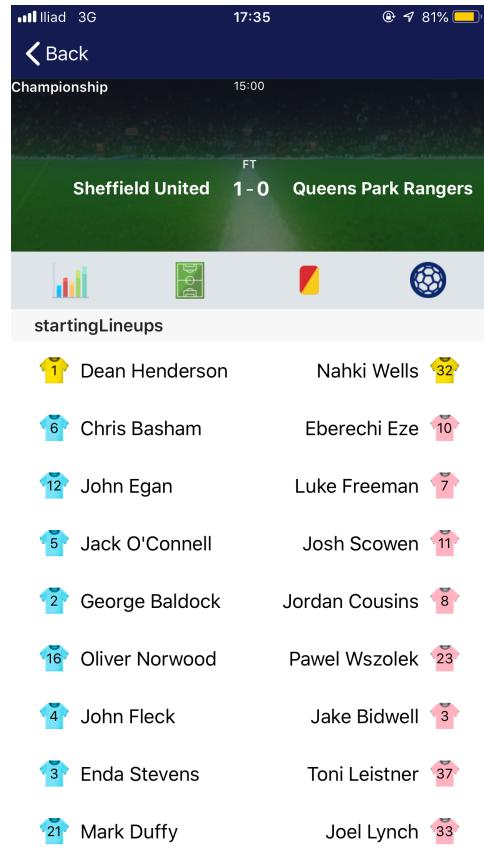


(b) WebView to see and read the complete article from the origin website

## Live



(a) This screen shows real time games

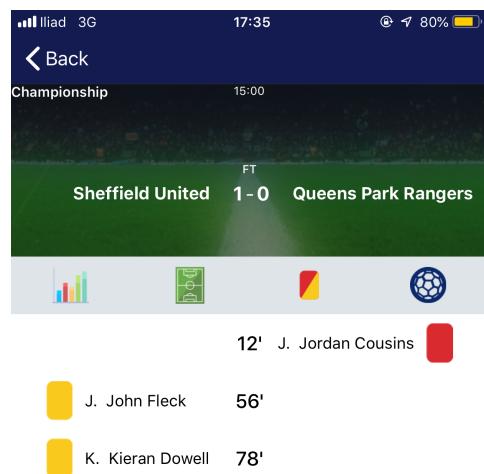


(b) In this screen are shown both teams lineups

## 5 User Interfaces



(a) This screen shows the goals



(b) This screen shows the match cards

## 5 User Interfaces

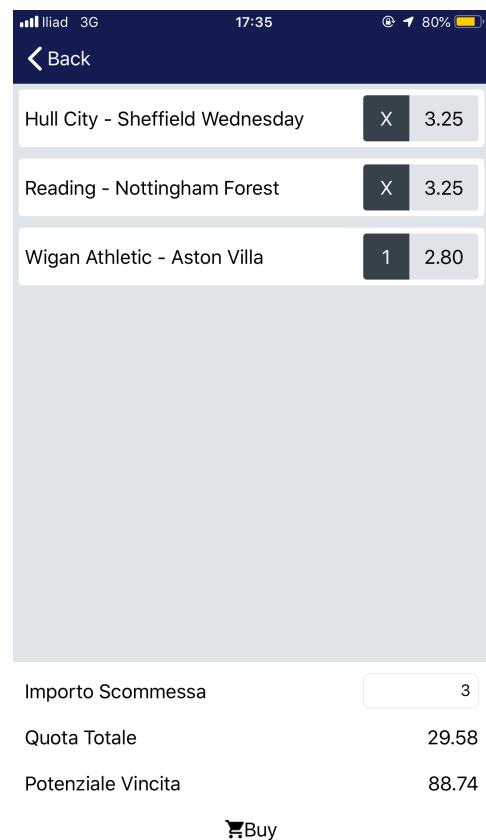


(a) This screen shows statistics

## Odds

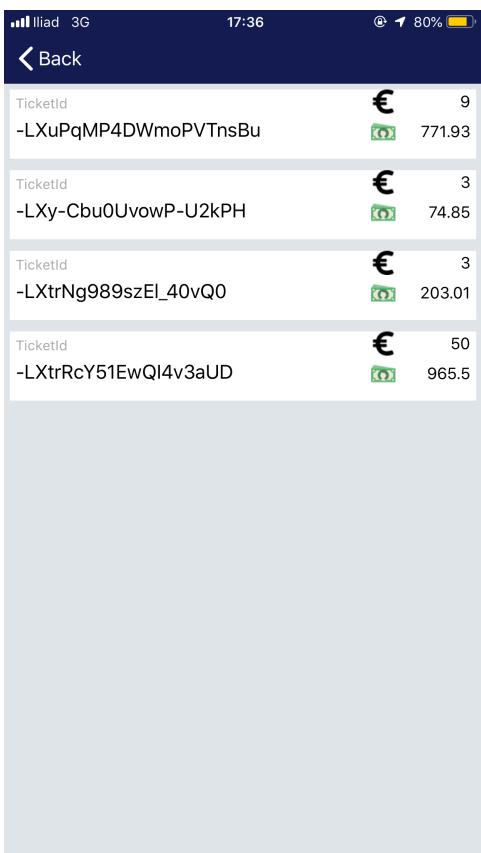


- (a) This screen shows the composition of a ticket



- (b) This screen shows the list of the games inserted in a ticket

## 5 User Interfaces



The screenshot shows a mobile application interface with a dark blue header bar. On the left of the header is a white back arrow icon. In the center, the time '17:36' is displayed. On the right, there are icons for signal strength ('Iliad 3G'), battery level ('80%'), and a location pin.

The main content area displays a list of four card plays, each represented by a light gray rectangular card. Each card contains the following information:

TicketId	€	9
-LXuPqMP4DWmoPVTnsBu	€	9
	€	9
-LXy-Cbu0UvowP-U2kPH	€	3
	€	3
-LXtrNg989szEl_40vQ0	€	3
	€	50
-LXtrRcY51EwQl4v3aUD	€	50

Each card also features a small green icon with a white dollar sign (\$) and a numerical value representing the amount.

- (a) This screen shows the history of all the cards played

## Chat



(a) This screen shows the chat

# 6 External Services and Libraries

iSport application uses several third parties API in order to provide the user all the services.

The JSONDecoder component from UIKit was used to parse the JSON data.

## Facebook SDK

The Facebook SDK was used to manage the login. When a user wants to authenticate the application makes the call to the Facebook service through:

```
@objc func loginButtonClicked() {
    let loginManager = LoginManager()
    loginManager.logIn([ .publicProfile ], viewController: self) { loginResult in
        switch loginResult {
        case .Failed(let error):
            print(error)
        case .Cancelled:
            print("User cancelled login.")
        case .Success(let grantedPermissions, let declinedPermissions, let accessToken):
            print("Logged in!")
        }
    }
}
```

This method is used to send the user on Facebook for the authentication; once authenticated the user redirected to "iSport" which will store the token provided.

Instead, the "Graph API" is used for news publishing:

```
let content = LinkShareContent(url: NSURL("https://developers.facebook.com"))
try ShareDialog.show(from: myViewController, content: content)
```

The following pods were included to use Facebook:

---

```
1 pod 'FacebookCore'
2 pod 'FacebookLogin'
3 pod 'FacebookShare'
```

## Firebase SDK

Firebase SDKs are used to access the database. In particular, login is possible only inserting Facebook credentials in order to have a single login.

## 6 External Services and Libraries

```
Auth.auth().signInAndRetrieveData(with: credential) { (authResult, error) in
    if let error = error {
        print("Errore Log In Firebase \(error)")
        return
    }
    print("Accesso eseguito Firebase")
}
```

For database access, the DatabaseManager included in the SDKs was used:

```
if let user = Auth.auth().currentUser {
    let database = Database.database().reference()
    database.child("Schedina").child(user.uid).observeSingleEvent(of: .value) { (snapshot) in
    }
}
```

The following pods were included to use Firebase:

---

```
1 pod 'Firebase/Core'
2 pod 'Firebase/Auth'
3 pod 'Firebase/Database'
```

---

## Scaledrone

Scaledrone API was used for the real-time chat service. The following extension is used to connect to the room:

```
extension ChatService: ScaledroneDelegate {
    func scaledroneDidConnect(scaledrone: Scaledrone, error: NSError?) {
        print("Connected to Scaledrone")
        room = scaledrone.subscribe(roomName: "observable-room")
        room?.delegate = self
    }

    func scaledroneDidReceiveError(scaledrone: Scaledrone, error: NSError?) {
        print("Scaledrone error", error ?? "")
    }

    func scaledroneDidDisconnect(scaledrone: Scaledrone, error: NSError?) {
        print("Scaledrone disconnected", error ?? "")
    }
}
```

While for receiving messages:

```

extension ChatService: ScaledroneRoomDelegate {
    func scaledroneRoomDidConnect(room: ScaledroneRoom, error: NSError?) {
        print("Connected to room!")
    }

    func scaledroneRoomDidReceiveMessage(
        room: ScaledroneRoom,
        message: Any,
        member: ScaledroneMember?) {

        guard
            let text = message as? String,
            let memberData = member?.clientData,
            let member = Member(fromJSON: memberData)
        else {
            print("Could not parse data.")
            return
        }

        let message = Message(
            member: member,
            text: text,
            messageId: UUID().uuidString)
        messageCallback(message)
    }
}

```

The following pods were included to use the Scaledrone service:

---

<sup>1</sup> pod 'Scaledrone', '~> 0.3.0'

---

## NewsAPI

If a user wants to see the most important daily news, the application will perform the following operations:

- sends a GET request to the endpoint supplied by the service
- If the answer is valid, it parses the JSON content
- Creates the table cells containing the information provided and parsed by the service
- Download preview images asynchronously

The endpoint used is "https://newsapi.org/v2/top-headlines?country=it&category=sports &apiKey=API\_KEY". The answer provided by the service is:

```
{
  status: "ok",
  totalResults: 70,
  - articles: [
    - {
      - source: {
        id: null,
        name: "Milannews.it"
      },
      author: "Salvatore Trovato",
      title: "Milan, La Repubblica: \"Vincoli e crescita, Gazidis indica la via di mezzo\" - Milan News",
      description: "Un \"sentiero ragionevole\" lungo il quale possono camminare insieme sostenibilità finanziaria e libertà di investimento. È questa - riporta il quotidiano La Repubblica",
      url: https://www.milannews.it/news/milan-la-repubblica-vincoli-e-crescita-gazidis-indica-la-via-di-mezzo-321868,
      urlToImage: https://net-storage.tccstatic.com/storage/milannews.it/img_notizie/thumb3/34/34bd9b6fe7d0e1748194-639c61846db6ba1fb8fe64bca2ce28f5.jpeg,
      publishedAt: "2019-01-25T09:37:26Z",
      content: "Un \"sentiero ragionevole\" lungo il quale possono camminare insieme sostenibilità finanziaria e libertà di investimento. È questa - riporta il quotidiano La Repubblica - la via di uscita dal braccio di ferro tra Milan e Uefa suggerita da Ivan Gazidis. \"Lo scop... [+300 chars]"
    },
    - {
      - source: {
        id: null,
        name: "Tuttojuve.com"
      },
    }
  ]
}
```

The following data structure was used for the parsing:

```
struct Article: Decodable {
  let author: String?
  let title: String?
  let description: String?
  let url: String?
  let urlToImage: String?
  let publishedAt: String?
}

struct Articoli: Decodable {
  let status: String
  let articles: [Article]
}
```

## APIFootball

When a user wants to see a daily match score the application performs the following operations:

- sends a GET request to the endpoint supplied by the service
- If the answer is valid, it parses the JSON content
- Creates the table cells containing the information provided and parsed by the service
- Update the information every 30 seconds

## 6 External Services and Libraries

The endpoint used to obtain the results is "https://apifootball.com/api/?action=get\_events". The answer provided by the service is:

```
[  
  {  
    "match_id": "277488",  
    "country_id": "170",  
    "country_name": "Italy",  
    "league_id": "79",  
    "League_name": "Serie A",  
    "match_date": "2018-04-22",  
    "match_status": "FT",  
    "match_time": "15:00",  
    "match_hometeam_name": "Juventus",  
    "match_hometeam_score": "0",  
    "match_awayteam_name": "SSC Napoli",  
    "match_awayteam_score": "1",  
    "match_hometeam_halftime_score": "0",  
    "match_awayteam_halftime_score": "0",  
    "match_hometeam_extra_score": "",  
    "match_awayteam_extra_score": "",  
    "match_hometeam_penalty_score": "",  
    "match_awayteam_penalty_score": "",  
    "match_hometeam_system": "4-3-2-1",  
    "match_awayteam_system": "4-3-3",  
    "match_live": "1",  
    "goalscorer": [  
      {"  
        "card": "Yellow",  
        "time": 10  
      },  
      {"  
        "card": "Red",  
        "time": 60  
      }  
    ],  
    "cards": [  
      {"  
        "card": "Yellow",  
        "time": 10  
      },  
      {"  
        "card": "Red",  
        "time": 60  
      }  
    ],  
    "lineup": {  
      "home": {  
        "name": "Juventus",  
        "position": "GK",  
        "number": 1, "player": "Alisson",  
        "substitution": null,  
        "lineupTime": null  
      },  
      "away": {  
        "name": "SSC Napoli",  
        "position": "GK",  
        "number": 1, "player": "Ottavio",  
        "substitution": null,  
        "lineupTime": null  
      },  
      "lineups": [  
        {"  
          "home": {  
            "name": "Juventus",  
            "position": "GK",  
            "number": 1, "player": "Alisson",  
            "substitution": null,  
            "lineupTime": null  
          },  
          "away": {  
            "name": "SSC Napoli",  
            "position": "GK",  
            "number": 1, "player": "Ottavio",  
            "substitution": null,  
            "lineupTime": null  
          }  
        },  
        {"  
          "home": {  
            "name": "Juventus",  
            "position": "RB",  
            "number": 2, "player": "Danilo",  
            "substitution": null,  
            "lineupTime": null  
          },  
          "away": {  
            "name": "SSC Napoli",  
            "position": "RB",  
            "number": 2, "player": "Mertens",  
            "substitution": null,  
            "lineupTime": null  
          }  
        },  
        {"  
          "home": {  
            "name": "Juventus",  
            "position": "RWB",  
            "number": 3, "player": "Dybala",  
            "substitution": null,  
            "lineupTime": null  
          },  
          "away": {  
            "name": "SSC Napoli",  
            "position": "RWB",  
            "number": 3, "player": "Insigne",  
            "substitution": null,  
            "lineupTime": null  
          }  
        },  
        {"  
          "home": {  
            "name": "Juventus",  
            "position": "LB",  
            "number": 4, "player": "Alvaro Morata",  
            "substitution": null,  
            "lineupTime": null  
          },  
          "away": {  
            "name": "SSC Napoli",  
            "position": "LB",  
            "number": 4, "player": "Giovanni Simeone",  
            "substitution": null,  
            "lineupTime": null  
          }  
        },  
        {"  
          "home": {  
            "name": "Juventus",  
            "position": "LWB",  
            "number": 5, "player": "Blas Iglesias",  
            "substitution": null,  
            "lineupTime": null  
          },  
          "away": {  
            "name": "SSC Napoli",  
            "position": "LWB",  
            "number": 5, "player": "Giovanni Simeone",  
            "substitution": null,  
            "lineupTime": null  
          }  
        },  
        {"  
          "home": {  
            "name": "Juventus",  
            "position": "CB",  
            "number": 6, "player": "Miralem Pjanic",  
            "substitution": null,  
            "lineupTime": null  
          },  
          "away": {  
            "name": "SSC Napoli",  
            "position": "CB",  
            "number": 6, "player": "Kalidou Koulibaly",  
            "substitution": null,  
            "lineupTime": null  
          }  
        },  
        {"  
          "home": {  
            "name": "Juventus",  
            "position": "DCB",  
            "number": 7, "player": "Paulo Dybala",  
            "substitution": null,  
            "lineupTime": null  
          },  
          "away": {  
            "name": "SSC Napoli",  
            "position": "DCB",  
            "number": 7, "player": "Jorginho",  
            "substitution": null,  
            "lineupTime": null  
          }  
        },  
        {"  
          "home": {  
            "name": "Juventus",  
            "position": "CAM",  
            "number": 8, "player": "Adrien Rabiot",  
            "substitution": null,  
            "lineupTime": null  
          },  
          "away": {  
            "name": "SSC Napoli",  
            "position": "CAM",  
            "number": 8, "player": "Fabio Quagliarella",  
            "substitution": null,  
            "lineupTime": null  
          }  
        },  
        {"  
          "home": {  
            "name": "Juventus",  
            "position": "RW",  
            "number": 9, "player": "Cristiano Ronaldo",  
            "substitution": null,  
            "lineupTime": null  
          },  
          "away": {  
            "name": "SSC Napoli",  
            "position": "RW",  
            "number": 9, "player": "Dries Mertens",  
            "substitution": null,  
            "lineupTime": null  
          }  
        },  
        {"  
          "home": {  
            "name": "Juventus",  
            "position": "LW",  
            "number": 10, "player": "Hirving Lozano",  
            "substitution": null,  
            "lineupTime": null  
          },  
          "away": {  
            "name": "SSC Napoli",  
            "position": "LW",  
            "number": 10, "player": "Giovanni Simeone",  
            "substitution": null,  
            "lineupTime": null  
          }  
        }  
      ]  
    }  
  }]
```

The following data structure was used for the parsing:

```
struct Partita: Decodable {  
    let leagueName: String?  
    let matchId: String?  
    let matchStatus: String?  
    let matchTime: String?  
    let matchDate: String?  
    let matchHometeamName: String?  
    let matchHometeamScore: String?  
    let matchAwayteamName: String?  
    let matchAwayteamScore: String?  
    let matchHometeamHalftimeScore: String?  
    let matchAwayteamHalftimeScore: String?  
    let goalscorer: [GoalList]  
    let cards: [CardList]  
    let statistics: [Statistic]  
    let lineup: Formazione  
}  
  
struct GoalList: Decodable {  
    let time: String?  
    let homeScorer: String?  
    let score: String?  
    let awayScorer: String?  
}  
  
struct CardList: Decodable {  
    let time: String?  
    let homeFault: String?  
    let card: String?  
    let awayFault: String?  
}  
  
struct Statistic: Decodable {  
    let type: String?  
    let home: String?  
    let away: String?  
}  
  
struct Formazione: Decodable {  
    let home: Campo?  
    let away: Campo?  
}  
  
struct Campo: Decodable {  
    let startingLineups: [Lineup]?  
    let substitutions: [Lineup]?  
}  
  
struct Lineup: Decodable {  
    let lineupPlayer: String?  
    let lineupNumber: String?  
    let lineupPosition: String?  
    let lineupTime: String?  
}
```

To get the game's predictions, the endpoint used is "https://apifootball.com/api/?action=get\_predictions". The answer provided by the service is:

## 6 External Services and Libraries

```
[  
  {  
    "match_id": "369909",  
    "country_id": "253",  
    "country_name": "Australia",  
    "league_id": "684",  
    "league_name": "A-League",  
    "match_date": "2019-01-08",  
    "match_status": "FT",  
    "match_time": "08:50",  
    "match_hometeam_name": "Western Sydney Wanderers FC",  
    "match_hometeam_score": "2",  
    "match_awayteam_name": "Wellington Phoenix FC",  
    "match_awayteam_score": "3",  
    "match_hometeam_halftime_score": "1",  
    "match_awayteam_halftime_score": "1",  
    "match_hometeam_extra_score": "",  
    "match_awayteam_extra_score": "",  
    "match_hometeam_penalty_score": "",  
    "match_awayteam_penalty_score": "",  
    "match_hometeam_system": "4-3-3",  
    "match_awayteam_system": "3-4-3",  
    "match_live": "0",  
    "prob_HW": "20.00",  
    "prob_D": "25.00",  
    "prob_AW": "55.00"  
  }  
]
```

The following data structure was used for the parsing:

```
struct Prediction: Decodable{  
  let matchId: String?  
  let probHW: String?  
  let probD: String?  
  let probAW: String?  
}
```

To get the odds of the games the endpoint used is "[https://apifootball.com/api/?action=get\\_odds](https://apifootball.com/api/?action=get_odds)". The answer provided by the service is:

```
[  
  {  
    "match_id": "148356",  
    "odd_bookmakers": "10Bet",  
    "odd_date": "2017-02-07 07:41:36",  
    "odd_1": "1.77",  
    "odd_X": "3.74",  
    "odd_2": "5.30"  
  }  
]
```

The following data structure was used for the parsing:

```
struct Odds: Decodable{  
  let matchId: String?  
  let odd1: String?  
  let oddX: String?  
  let odd2: String?  
}
```

# **7 Software System Attribute**

## **7.1 Reliability**

As the majority of function requires an internet connection, the software is reliable as long as there are no connectivity problems.

## **7.2 Availability**

The availability parameter also relies on the internet connection signal and on the responses provided by "https://newsapi.org" and "https://apifootball.com". No problems of availability were found both during the developement phase and beta testing with users.

## **7.3 Security**

The security of the iSport application was the main concern during the developement. As all the information provided are retrived from the web, there are different checks to perform in order to keep the user information safe. Furthermore, for the communication with external services, was chosen HTTPS protocol to guarantee greater security

## **7.4 Maintainability**

The entire application is very maintainable as the code is entirely documented, in particular in several critical function. Therefore any developer who wants to improve it or make changes is able to do it without relevant difficulty.

## **7.5 Usability**

The usability was another of our concern for the development. In order to improve it, a beta version of the application was given to 5 users and tested for 1 weeks. The result of this test highlighted usability issues. In particular, in the first version the buttons for changing information on the match detail page were not clearly visible. Another report concerned the lack of visual feedback after adding a quota to their ticket. Both of these problems were then solved in a later version of the application.

# **8 Test Cases**

# **9 Cost Estimation**