

POLITECNICO DI MILANO



POLITECNICO
MILANO 1863

DESIGN AND IMPLEMENTATION OF MOBILE
APPLICATIONS

iSport

Design Document

di
Gianluigi Oliva

February 6, 2019

Contents

1	Introduction	3
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	
2	Architecture	7
2.1	Database	7
2.2	Client	8
3	Use case functional requirements analysis	9
4	Sequence Diagram	31
5	User Interfaces	44
6	External Services and Libraries	46
7	Software System Attribute	52
7.1	Reliability	52
7.2	Availability	52
7.3	Security	52
7.4	Maintainability	52
7.5	Usability	52
8	Test Cases	53
9	Cost Estimation	54

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. Inoltre per favorire la creazione di una comunità di discussione l’applicazione fornirà un sistema di live chat attraverso cui scambiarsi pareri.

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.
- **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**

1 Introduction

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 on the bet amount.

Nella sezione "Chat" ci si collegherà ad una room globale in cui poter parlare con altri utenti che stanno utilizzando l'applicazione per scambiarsi commenti e pareri.

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:

- View news previews
- Read the complete article
- View football match results real time
- Display goal-scores
- Display booking (caution)
- Display team playing the match
- Display statistics
- Display game share
- Compose your ticket
- Display the potential winnings of the ticket

- Condividere delle notizie su Facebook
- Salvare delle notizie tra i preferiti
- chattare con altri utenti
- Esegire l'accesso

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.

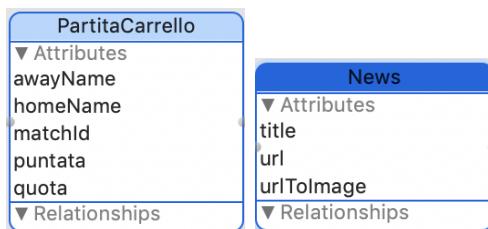
Assumptions and Dependencies

- **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.

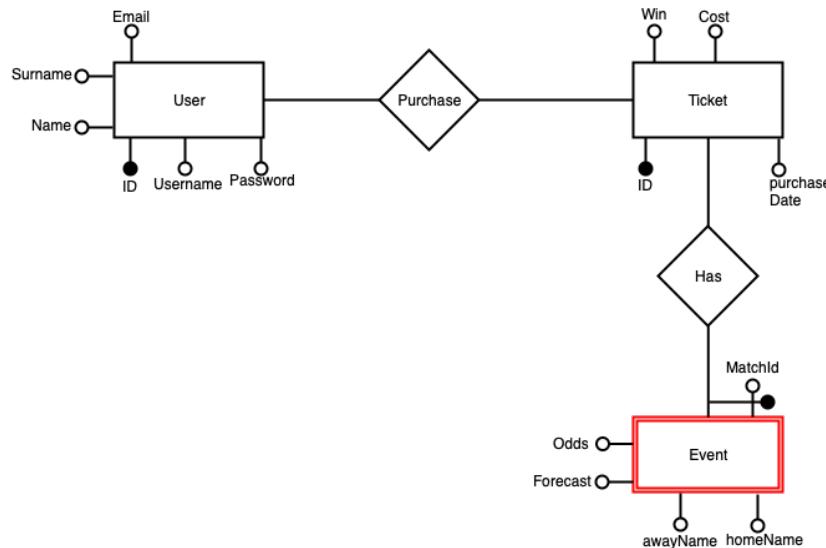
2 Architecture

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.



Per permettere l'acquisto di schedine è stato usato un Database online utilizzando la piattaforma di Firebase. Il modello E/R utilizzato per quest'ultimo è il seguente:



Per garantire la privacy di ogni utente, l'accesso al Database è regolato mediante le seguenti 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.

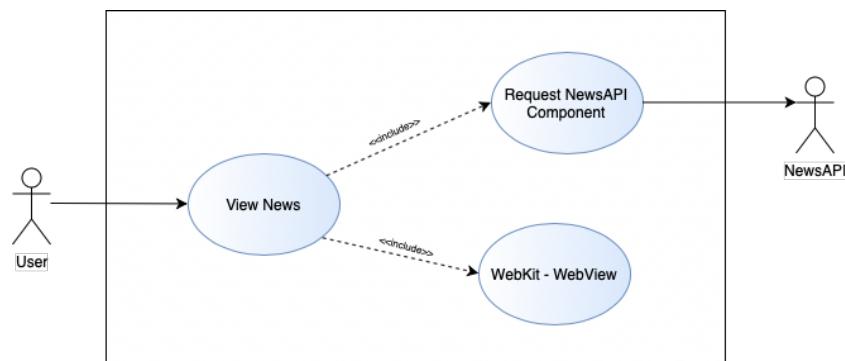


Figure 3.1: Use case relativo alla visualizzazione e manipolazione delle notizie

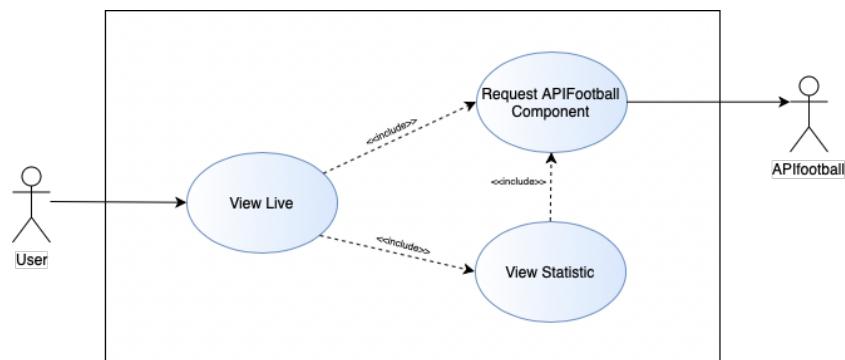


Figure 3.2: Use case relativo alla visualizzazione e manipolazione dei risultati

3 Use case functional requirements analysis

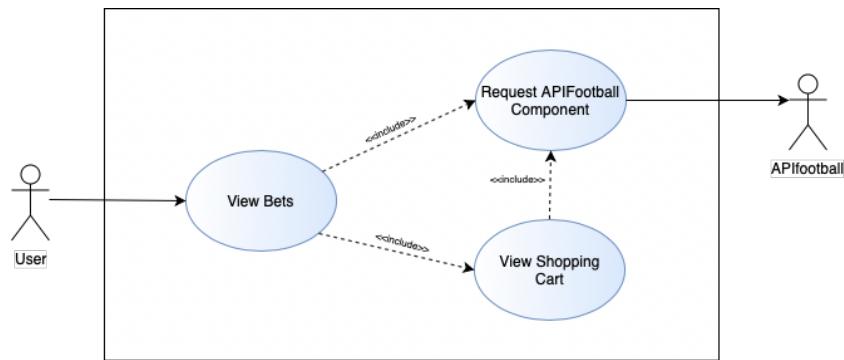


Figure 3.3: Use case relativo alla visualizzazione e manipolazione delle quote

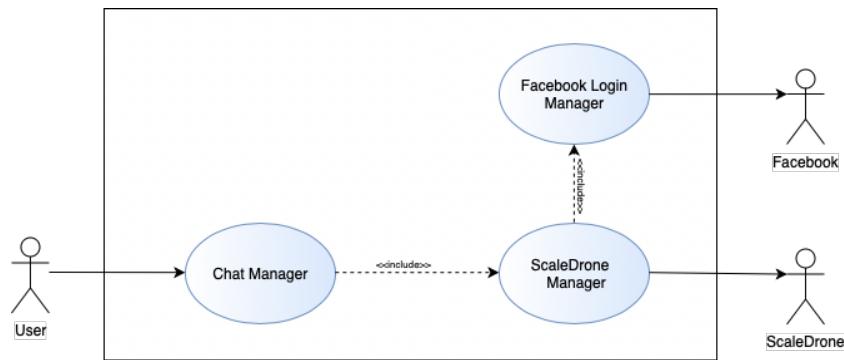


Figure 3.4: Use case del servizio chat

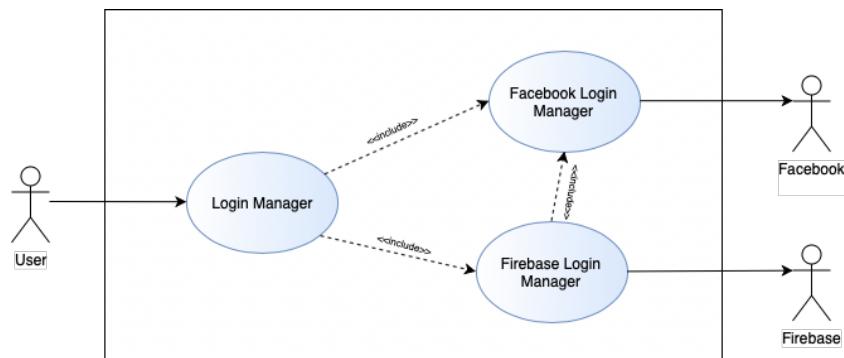


Figure 3.5: Use case per l'esecuzione del Login

3 Use case functional requirements analysis

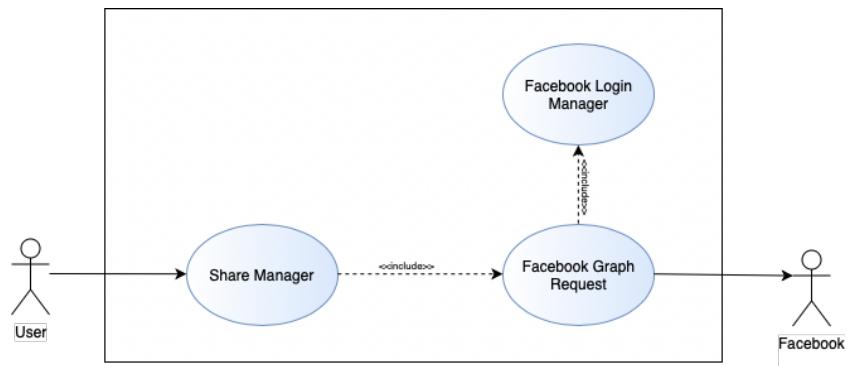


Figure 3.6: Use case relativo condivisione di notizie

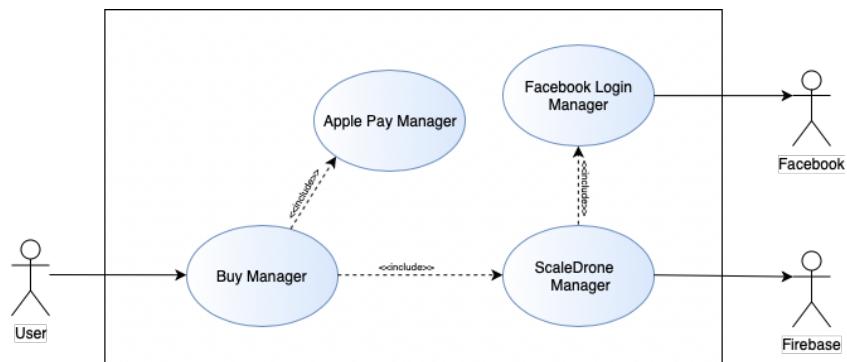


Figure 3.7: Use case relativo all'acquisto di schedine

Login

Name	Login
Actor	Guest
Entry Condition	Guest with login credentials
Goal	1, 2
Event Flow	<ul style="list-style-type: none"> • L'utente apre l'applicazione • L'utente preme sulla tab "Login" del "Side Menu" • L'utente viene rimandato sul sito di Facebook per l'inserimento delle credenziali • L'applicazione effettua il login tramite Facebook • L'applicazione effettua il login a Firebase usando l'account di Facebook
Exit condition	Actor becomes User
Exceptions	L'utente non è connesso alla rete. L'utente non ha un account Facebook.

Logout

Name	Logout
Actor	User
Entry Condition	Actor is logged in
Goal	1, 2
Event Flow	<ul style="list-style-type: none"> • L'utente preme sulla tab "Logout" del "Side Menu" • L'applicazione effettua il logout da Facebook • L'applicazione effettua il logout da Firebase
Exit condition	Actor is logged out and becomes Guest
Exceptions	L'utente non è connesso alla rete.

Visualizzare Notizie

Name	Visualizzare Notizie
Actor	Guest
Entry Condition	L'utente ha scaricato l'applicazione
Goal	1, 2
Event Flow	<ul style="list-style-type: none"> • L'utente apre l'applicazione • L'utente preme sulla tab "News" del "Side Menu" • L'applicazione fornisce un elenco delle notizie principali • L'utente preme sull'articolo di cui vuole visualizzare la notizia completa • L'utente preme il tasto "Done" per terminare la lettura
Exit condition	L'utente ha letto la notizia di interesse.
Exceptions	L'utente non è connesso alla rete e quindi non può inviare le richieste per ottenere gli articoli e quindi non può mostrarle. L'articolo è stato rimosso dal sito di origine, ma non nel database del servizio terzo

Condividere Notizie

Name	Condividere Notizie
Actor	User
Entry Condition	L'attore si è autenticato correttamente
Goal	1, 2
Event Flow	<ul style="list-style-type: none"> • L'utente apre l'applicazione • L'utente preme sulla tab "News" del "Side Menu" • L'applicazione fornisce un elenco delle notizie principali • L'utente preme il simbolo di "Share" della notizia che vuole condividere • L'utente compone l'eventuale messaggio da allegare nel post • L'utente preme il tasto "Pubblica" per terminare la condivisione
Exit condition	L'utente ha condiviso su Facebook la notizia di interesse.
Exceptions	L'utente non è connesso alla rete e quindi non può inviare le richieste per ottenere gli articoli e quindi non può mostrarli. L'articolo è stato rimosso dal sito di origine, ma non nel database del servizio terzo. La connessione è debole e l'applicazione non riesce a mandare i dati per la condivisione.

Salvare Notizie

Name	Salvare Notizie
Actor	User
Entry Condition	L'attore ha scaricato l'applicazione
Goal	1, 2
Event Flow	<ul style="list-style-type: none"> • L'utente apre l'applicazione • L'utente preme sulla tab "News" del "Side Menu" • L'applicazione fornisce un elenco delle notizie principali • L'utente preme il simbolo di "Elenco" della notizia che vuole memorizzare • L'applicazione memorizza le informazioni necessarie nel database locale
Exit condition	L'utente ha memorizzato una notizia per una visione successiva.
Exceptions	L'utente non è connesso alla rete e quindi non può inviare le richieste per ottenere gli articoli e quindi non può mostrarle. L'articolo è stato rimosso dal sito di origine, ma non nel database del servizio terzo.

Visualizzare Notizie Salvate

Name	Visualizzare Notizie Salvate
Actor	User
Entry Condition	L'attore ha salvato delle notizie in precedenza
Goal	1, 2
Event Flow	<ul style="list-style-type: none"> • L'utente apre l'applicazione • L'utente preme sulla tab "News" del "Side Menu" • L'utente preme il tasto "Elenco" nella Navigation Bar per consultare le notizie salvate • L'applicazione fornisce un elenco delle notizie salvate in precedenza
Exit condition	L'utente ha memorizzato una notizia per una visione successiva.
Exceptions	L'utente non è connesso alla rete e quindi non può inviare le richieste per ottenere gli articoli e quindi non può mostrarle. L'articolo è stato rimosso dal sito di origine, ma non nel database del servizio terzo.

Visualizzare Partite

Name	Visualizzare Partite
Actor	Guest
Entry Condition	L'attore ha scaricato l'applicazione
Goal	3
Event Flow	<ul style="list-style-type: none"> • L'utente apre l'applicazione • L'utente preme sulla tab "Live" del "Side Menu" • L'applicazione fornisce un elenco delle partite del giorno
Exit condition	L'utente ha visualizzato i risultati delle partite.
Exceptions	L'utente non è connesso alla rete e quindi non può inviare le richieste per ottenere i risultati e quindi non può mostrargli. La connessione potrebbe essere debole e quindi non riuscire a ricevere i dati richiesti.

Visualizzare Formazione Partita

Name	Visualizzare Formazione Partita
Actor	Guest
Entry Condition	L'utente ha scaricato l'applicazione
Goal	3, 6
Event Flow	<ul style="list-style-type: none"> • L'utente apre l'applicazione • L'utente preme sulla tab "Live" del "Side Menu" • L'applicazione fornisce un elenco delle partite del giorno • L'utente preme sulla partita di cui vuole ottenere l'informazione richiesta • L'utente preme sul bottone raffigurante il campo di gioco
Exit condition	L'utente ha visualizzato le formazioni delle due squadre della partita richiesta.
Exceptions	L'utente non è connesso alla rete e quindi non può inviare le richieste per ottenere i risultati e quindi non può mostrargli. La connessione potrebbe essere debole e quindi non riuscire a ricevere i dati richiesti.

Visualizzare Marcatori Partita

Name	Visualizzare Marcatori Partita
Actor	Guest
Entry Condition	L'attore ha scaricato l'applicazione
Goal	3, 4
Event Flow	<ul style="list-style-type: none"> • L'utente apre l'applicazione • L'utente preme sulla tab "Live" del "Side Menu" • L'applicazione fornisce un elenco delle partite del giorno • L'utente preme sulla partita di cui vuole ottenere l'informazione richiesta • L'utente preme sul bottone raffigurante un pallore
Exit condition	L'utente ha visualizzato i marcatori della partita richiesta.
Exceptions	L'utente non è connesso alla rete e quindi non può inviare le richieste per ottenere i risultati e quindi non può mostrargli. La connessione potrebbe essere debole e quindi non riuscire a ricevere i dati richiesti.

Visualizzare Statistiche Partita

Name	Visualizzare Statistiche Partita
Actor	Guest
Entry Condition	L'utente ha scaricato l'applicazione
Goal	3, 7
Event Flow	<ul style="list-style-type: none"> • L'utente apre l'applicazione • L'utente preme sulla tab "Live" del "Side Menu" • L'applicazione fornisce un elenco delle partite del giorno • L'utente preme sulla partita di cui vuole ottenere l'informazione richiesta • L'utente preme sul bottone raffigurante un grafico
Exit condition	L'utente ha visualizzato le statistiche della partita richiesta.
Exceptions	L'utente non è connesso alla rete e quindi non può inviare le richieste per ottenere i risultati e quindi non può mostrargli. La connessione potrebbe essere debole e quindi non riuscire a ricevere i dati richiesti.

Visualizzare Ammonizioni Partita

Name	Visualizzare Ammonizioni Partita
Actor	Guest
Entry Condition	L'utente ha scaricato l'applicazione
Goal	3, 5
Event Flow	<ul style="list-style-type: none"> • L'utente apre l'applicazione • L'utente preme sulla tab "Live" del "Side Menu" • L'applicazione fornisce un elenco delle partite del giorno • L'utente preme sulla partita di cui vuole ottenere l'informazione richiesta • L'utente preme sul bottone raffigurante un cartellino
Exit condition	L'utente ha visualizzato gli ammoniti ed espulsi della partita richiesta.
Exceptions	L'utente non è connesso alla rete e quindi non può inviare le richieste per ottenere i risultati e quindi non può mostrargli. La connessione potrebbe essere debole e quindi non riuscire a ricevere i dati richiesti.

Visualizzare Quote

Name	Visualizzare Quote
Actor	Guest
Entry Condition	L'attore ha scaricato l'applicazione
Goal	8
Event Flow	<ul style="list-style-type: none"> • L'utente apre l'applicazione • L'utente preme sulla tab "Bet" del "Side Menu" • L'applicazione fornisce un elenco delle quote dei principali pronostici delle partite odierne
Exit condition	L'utente ha visualizzato le quote delle partite odierne.
Exceptions	L'utente non è connesso alla rete e quindi non può inviare le richieste per ottenere le quote. La connessione potrebbe essere debole e quindi non riuscire a ricevere i dati richiesti.

Visualizza Schedina Composta

Name	Visualizza Schedina Composta
Actor	Guest
Entry Condition	L'attore ha scaricato l'applicazione
Goal	9
Event Flow	<ul style="list-style-type: none"> • L'utente apre l'applicazione • L'utente preme sulla tab "Bet" del "Side Menu" • L'utente preme sul bottone raffigurante un carrello nella NavBar
Exit condition	L'utente visualizza i pronostici presenti nella schedina.
Exceptions	L'utente non è connesso alla rete e quindi non può inviare le richieste per ottenere le quote. La connessione potrebbe essere debole e quindi non riuscire a ricevere i dati richiesti.

Aggiungere Partita a Schedina

Name	Aggiungere Partita a Schedina
Actor	Guest
Entry Condition	L'utente ha scaricato l'applicazione
Goal	9
Event Flow	<ul style="list-style-type: none"> • L'utente apre l'applicazione • L'utente preme sulla tab "Bet" del "Side Menu" • L'applicazione fornisce un elenco delle quote dei principali pronostici delle partite odierne • L'utente preme sulla quota relativa al pronostico e partita da aggiungere
Exit condition	L'utente ha aggiunto un pronostico di una partita alla schedina.
Exceptions	L'utente non è connesso alla rete e quindi non può inviare le richieste per ottenere le quote. La connessione potrebbe essere debole e quindi non riuscire a ricevere i dati richiesti.

Eliminare Partita da Schedina

Name	Eliminare Partita a Schedina
Actor	Guest
Entry Condition	L'utente ha scaricato l'applicazione
Goal	9
Event Flow	<ul style="list-style-type: none"> • L'utente apre l'applicazione • L'utente preme sulla tab "Bet" del "Side Menu" • L'utente preme sul bottone raffigurante un carrello nella NavBar • L'utente effettua uno swipe verso sinistra sulla partita da eliminare
Exit condition	L'utente ha eliminato un pronostico di una partita alla schedina.
Exceptions	Nessuna Eccezione

Calcolare potenziale Vincita Schedina

Name	Calcolare potenziale Vincita Schedina
Actor	Guest
Entry Condition	L'attore ha scaricato l'applicazione
Goal	10
Event Flow	<ul style="list-style-type: none"> • L'utente apre l'applicazione • L'utente preme sulla tab "Bet" del "Side Menu" • L'utente preme sul bottone raffigurante un carrello nella NavBar • L'utente inserisce l'importo da giocare nell'apposito input • L'utente preme il tasto "Done"
Exit condition	L'utente visualizza la potenziale vincita della schedina attuale.
Exceptions	Nessuna Eccezione

Acquisto Schedina

Name	Acquisto Schedina
Actor	User
Entry Condition	L'attore si è autenticato correttamente
Goal	10
Event Flow	<ul style="list-style-type: none"> • L'utente apre l'applicazione • L'utente preme sulla tab "Bet" del "Side Menu" • L'utente preme sul bottone raffigurante un carrello nella NavBar • L'utente inserisce l'importo che vuole giocare • L'utente preme sul bottone "Buy"
Exit condition	L'utente acquista la schedina attuale.
Exceptions	La connessione è assente e non può portare a termine l'acquisto. L'utente non è autenticato. L'utente non ha inserito nessuna partita nella schedina

Visualizza Schedine Acquistate

Name	Visualizza Schedine Acquistate
Actor	User
Entry Condition	L'attore si è autenticato correttamente
Goal	9
Event Flow	<ul style="list-style-type: none"> • L'utente apre l'applicazione • L'utente preme sulla tab "Bet" del "Side Menu" • L'utente preme sul bottone "cronologia" nella NavBar • L'applicazione scarica le informazioni su tutte le schedine acquistate • L'utente preme sulla cella corrispondente alla schedina che vuole visualizzare • L'applicazione carica il dettaglio di tutti gli eventi presenti nella schedina
Exit condition	L'utente visualizza una schedina acquistata in precedenza.
Exceptions	L'utente non è connesso alla rete e quindi non può inviare le richieste per ottenere la cronologia delle giocate. La connessione potrebbe essere debole e quindi non riuscire a ricevere i dati richiesti. L'utente non è autenticato. L'utente non ha giocato nessuna schedina.

Chat

Name	Login
Actor	User
Entry Condition	L'attore si è autenticato correttamente
Goal	1, 2
Event Flow	<ul style="list-style-type: none"> • L'utente apre l'applicazione • L'utente preme sulla tab "Chat" del "Side Menu" • L'utente viene rimandato sul sito di Facebook per l'inserimento delle credenziali • L'applicazione effettua il login tramite la room di Scaledrone usando le informazioni di Facebook • L'utente scrive i propri messaggi nella casella di testo • L'utente preme invio per spedire il proprio messaggio • L'applicazione carica tutti i messaggi presenti nella room
Exit condition	L'utente interagisce con altre persone
Exceptions	L'utente non è connesso alla rete. L'utente non ha un account Facebook.

4 Sequence Diagram

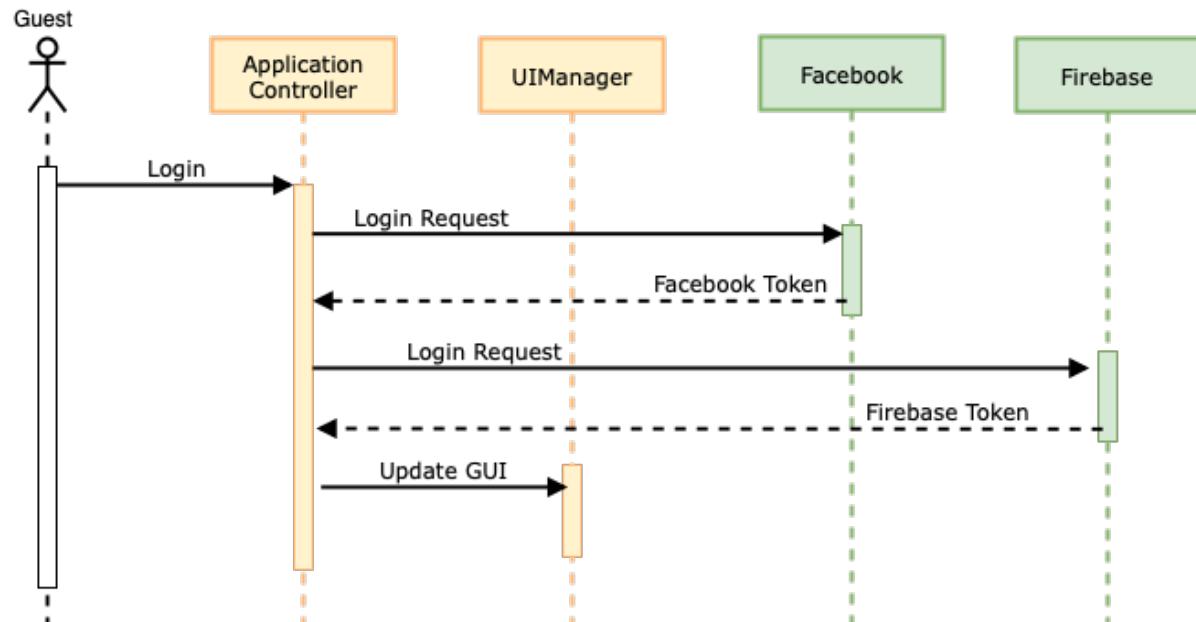
In order to explain our work and facilitate further implementations we have decided to show the logical flows aimed to create some features. The sequence diagrams will describe the interactions between the different parts of the system and the user.

Login

La procedura di "Login" inizia quando l'utente preme il relativo bottone del Side Menu. Una volta premuto l'utente viene reindirizzato sul sito di Facebook per completare la registrazione. Subito dopo il login a Facebook l'applicazione effettuerà il login al Database di Firebase.

Una volta completata questa procedura il sistema aggiornerà la GUI abilitando le sezioni disponibili solo per gli iscritti.

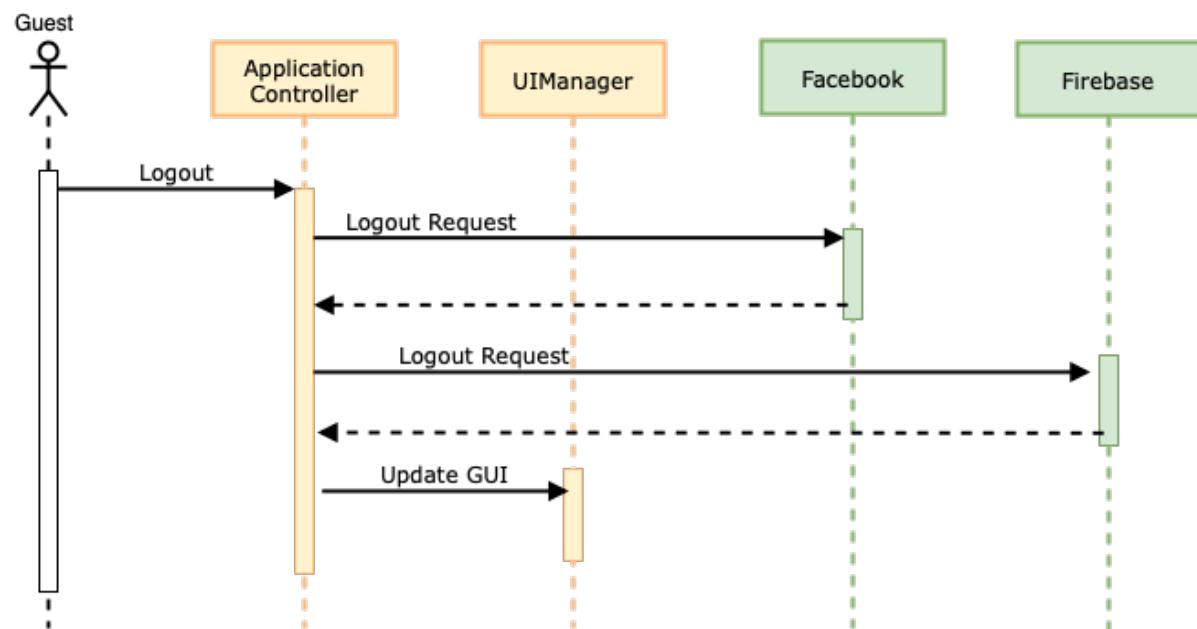
In caso di errore il sistema restituirà un messaggio di errore.



Logout

La procedura di "Logout" inizia quando l'utente preme il relativo bottone del Side Menu. Una volta premuto verrà visualizzato un messaggio di conferma. In caso di approvazione verrà effettuato il logout da Facebook e Firebase.

Una volta completata questa procedura il sistema aggiornerà la GUI abilitando le sezioni non disponibili per i guest.



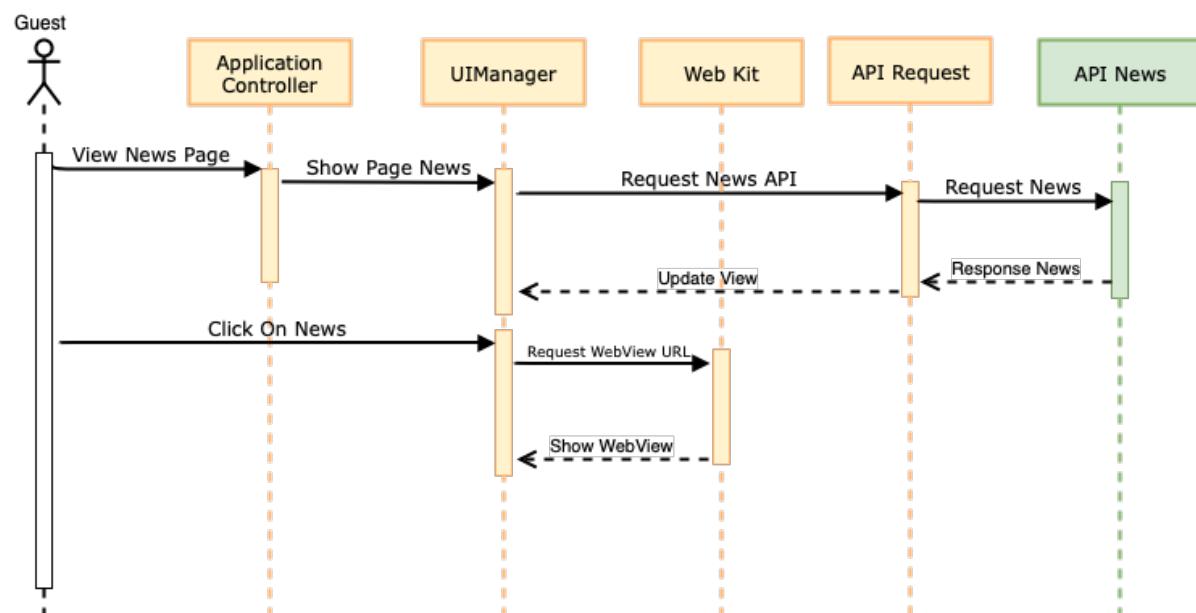
View News

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.

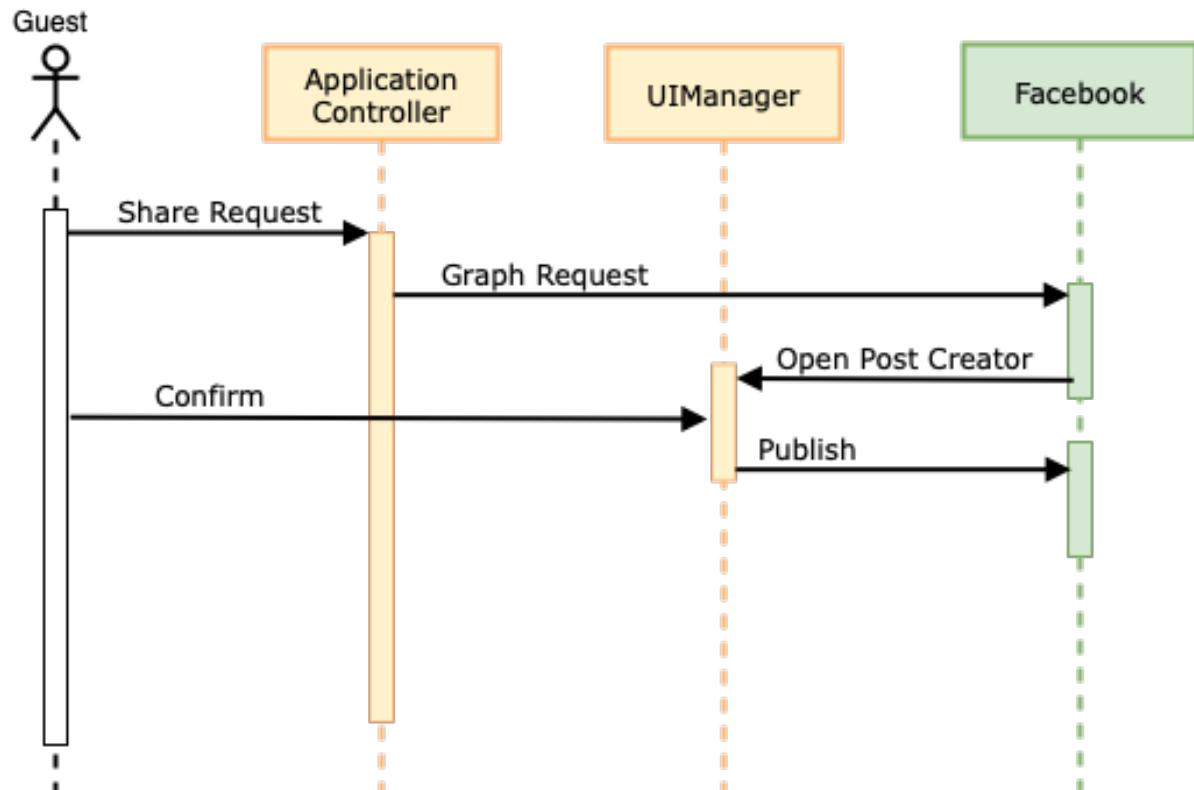


Share News

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

Dopo la normale procedura di "View News" l'utente preme sul bottone di share. A quel punto l'applicazione farà una richiesta alle "Graph" API di Facebook per la creazione del post con il testo inserito dall'utente.

Tale operazione ha come condizione che l'utente sia già autenticato e in caso contrario restituirà una avviso di errore.



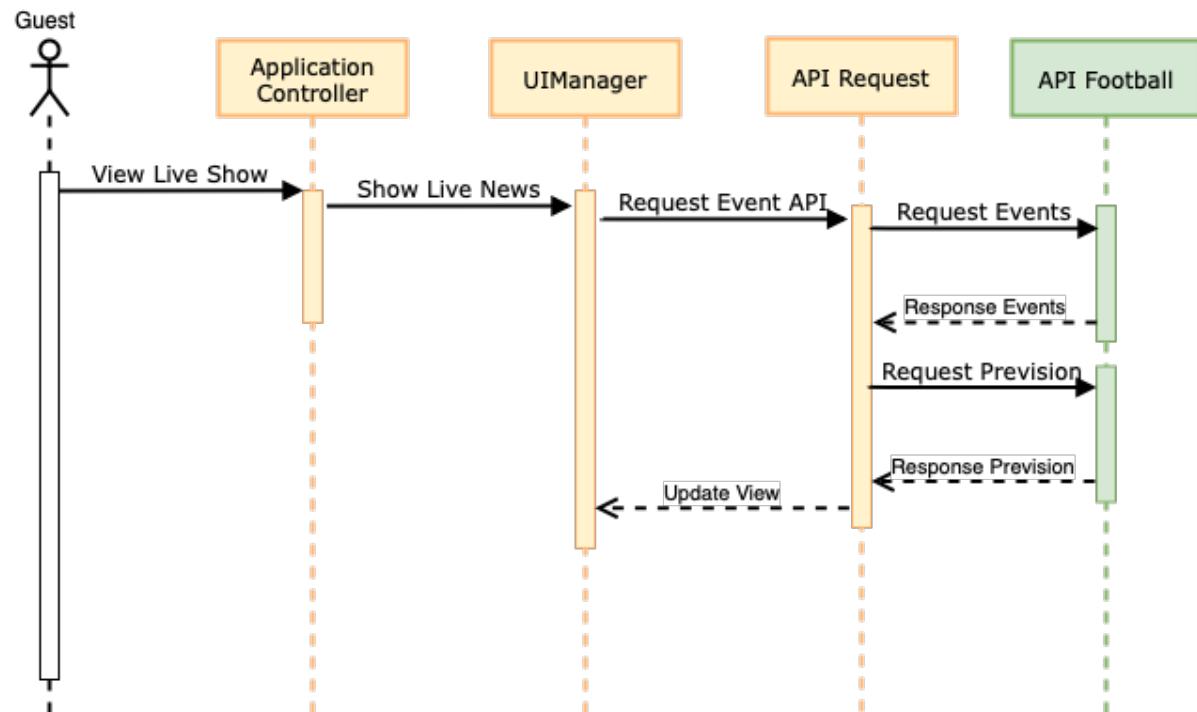
View Match

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.

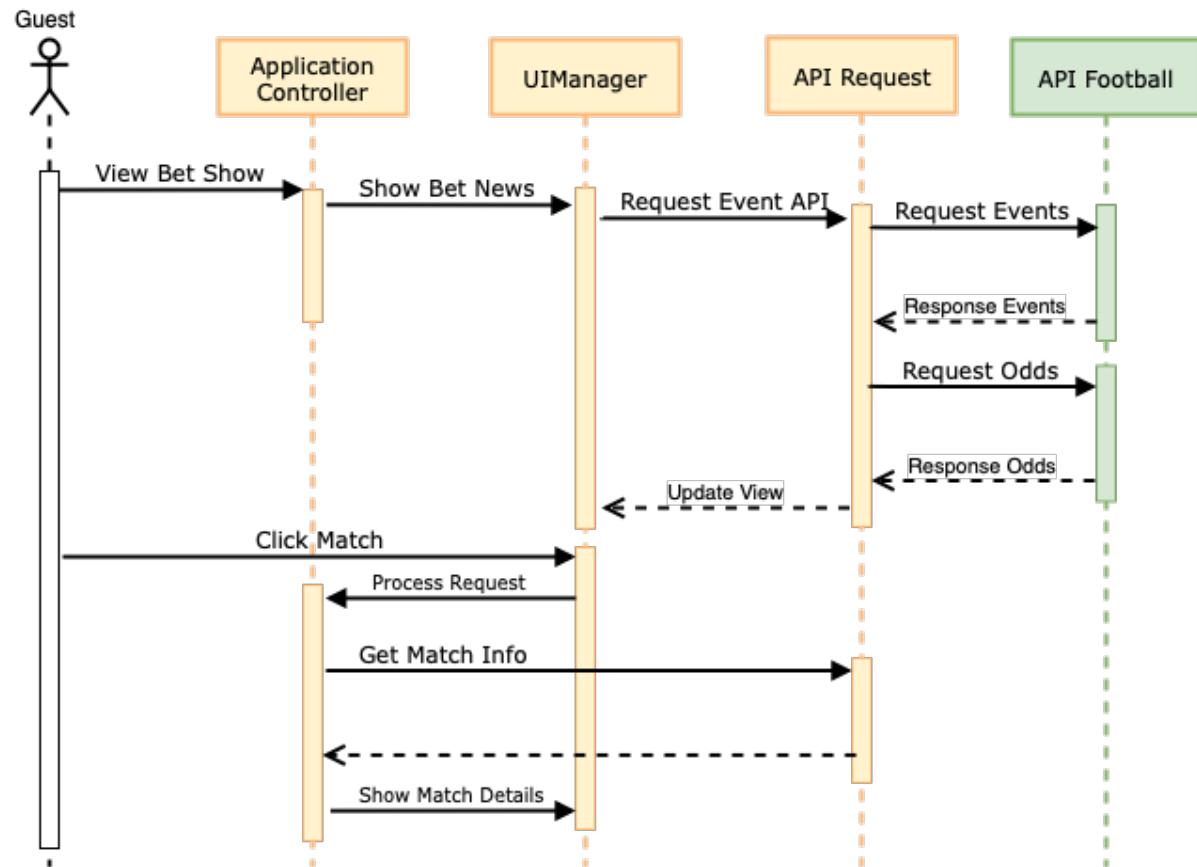


View Match Details

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.

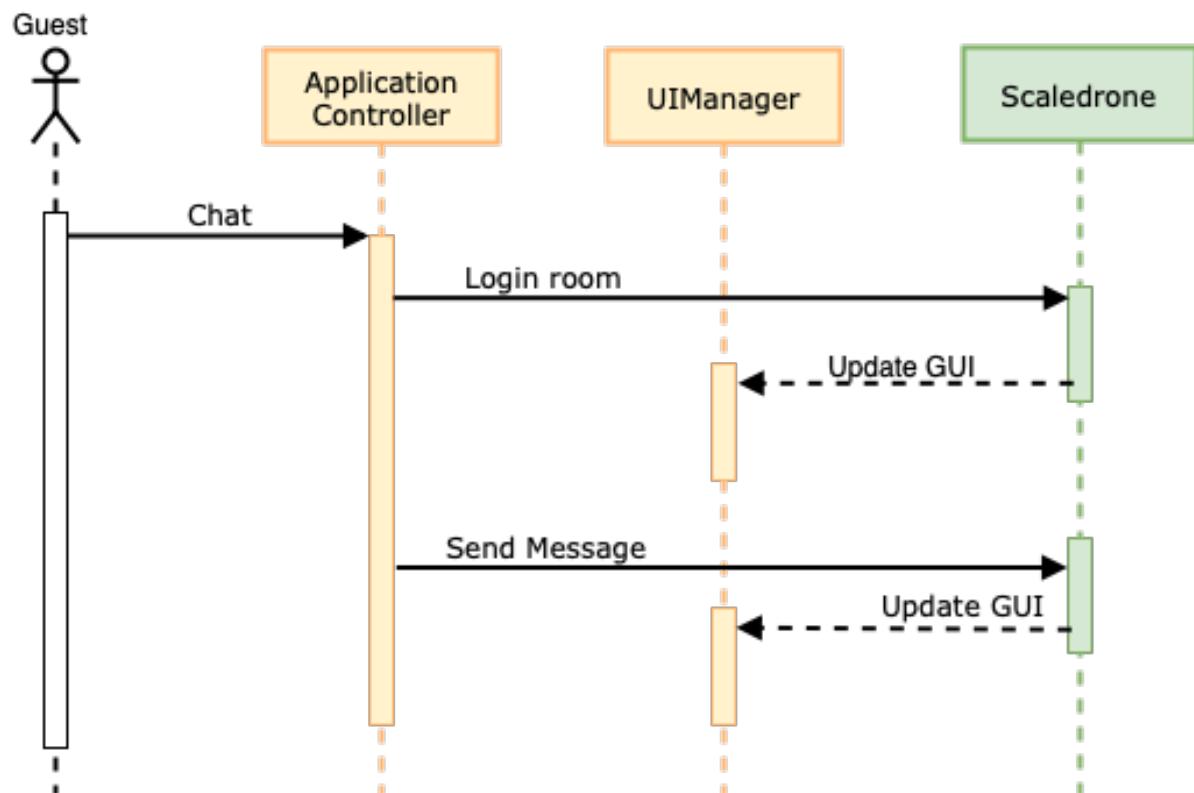


Chat

La procedura di "Chat" inizia quando l'utente preme il relativo bottone del Side Menu. Una volta caricata la schermata, l'applicazione rimarrà in ascolto della room mostrando in tempo reale tutti i messaggi inviati dagli altri utenti. L'utente può poi scrivere e inviare il proprio messaggio.

L'applicazione inoltre scaricherà le immagini del profilo relative ai messaggi inviati dagli altri utenti in maniera asincrona.

Tale procedura presuppone che l'utente sia già autenticato altrimenti verrà visualizzato un messaggio di errore.



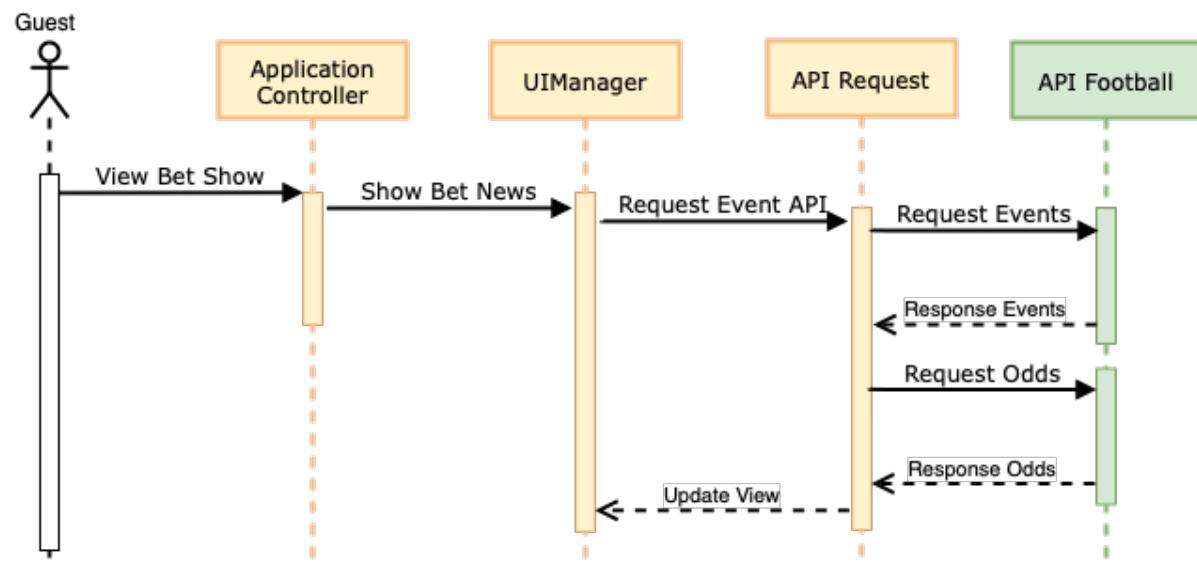
View Odds

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.



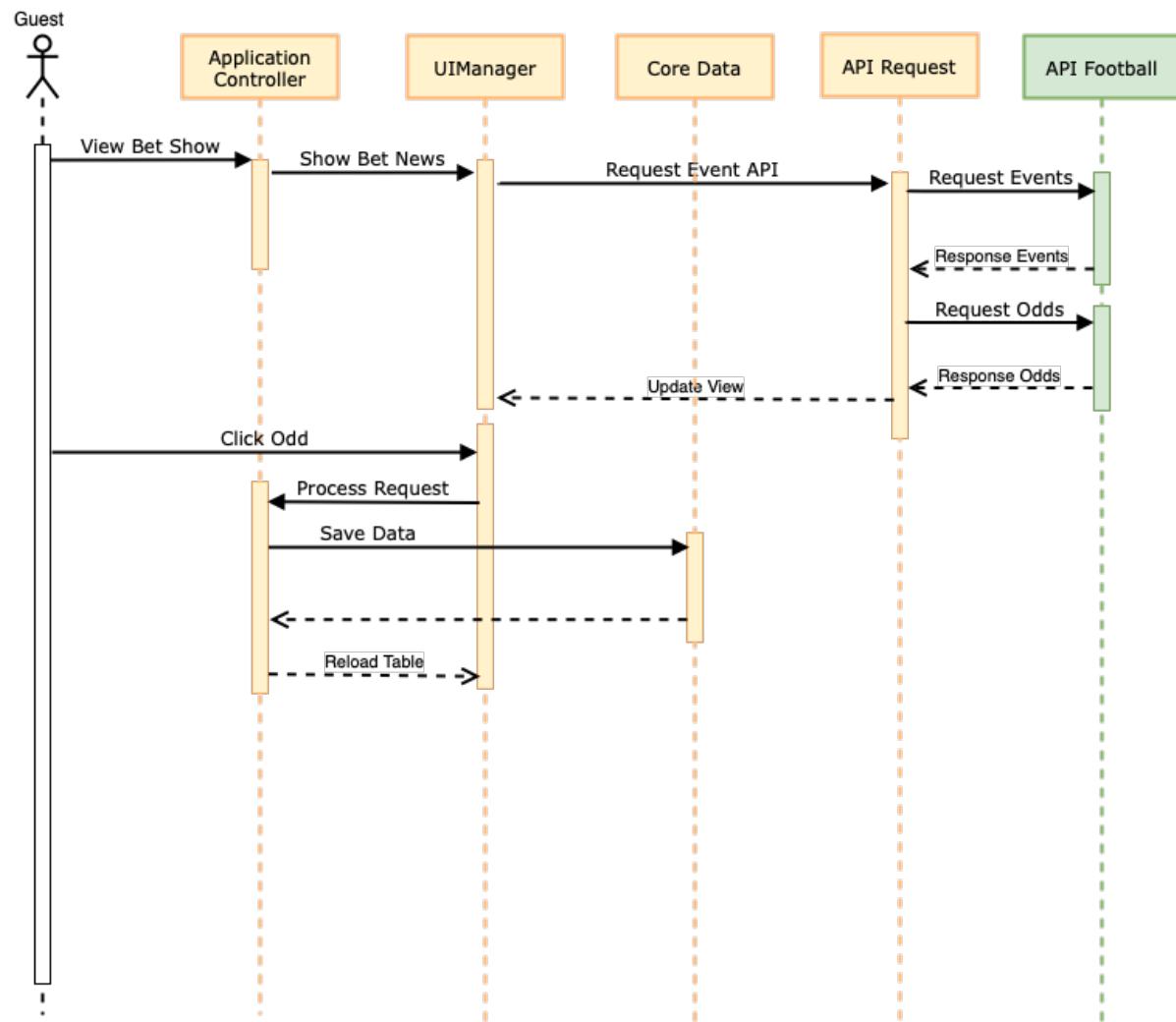
Add Odd

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.

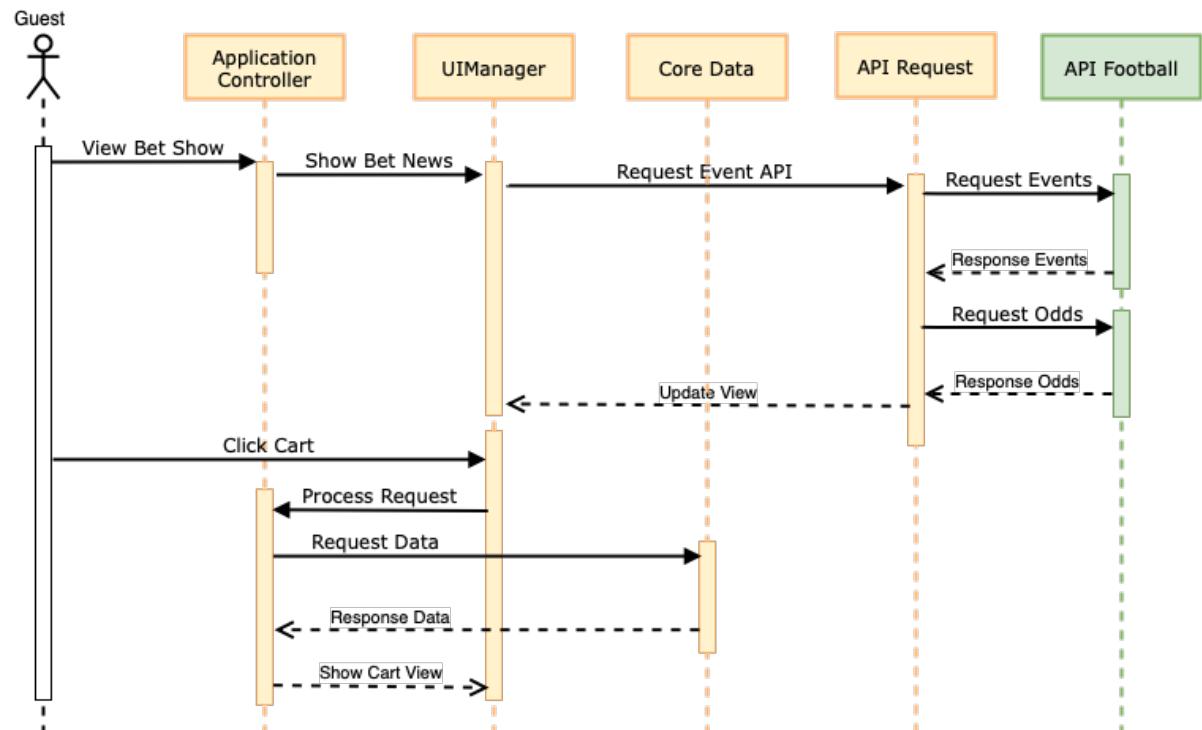


View Cart

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.

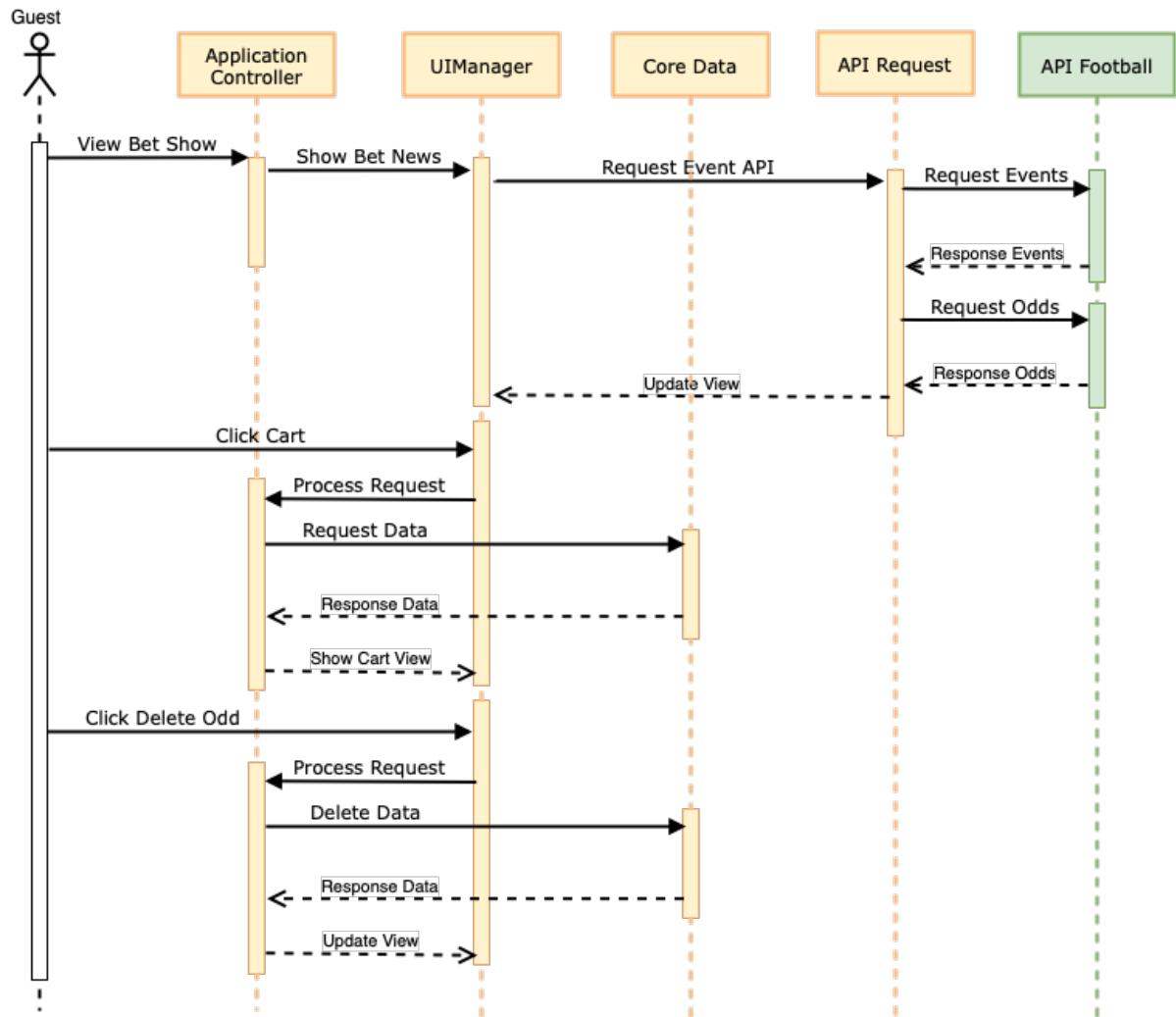


Remove Odd

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.

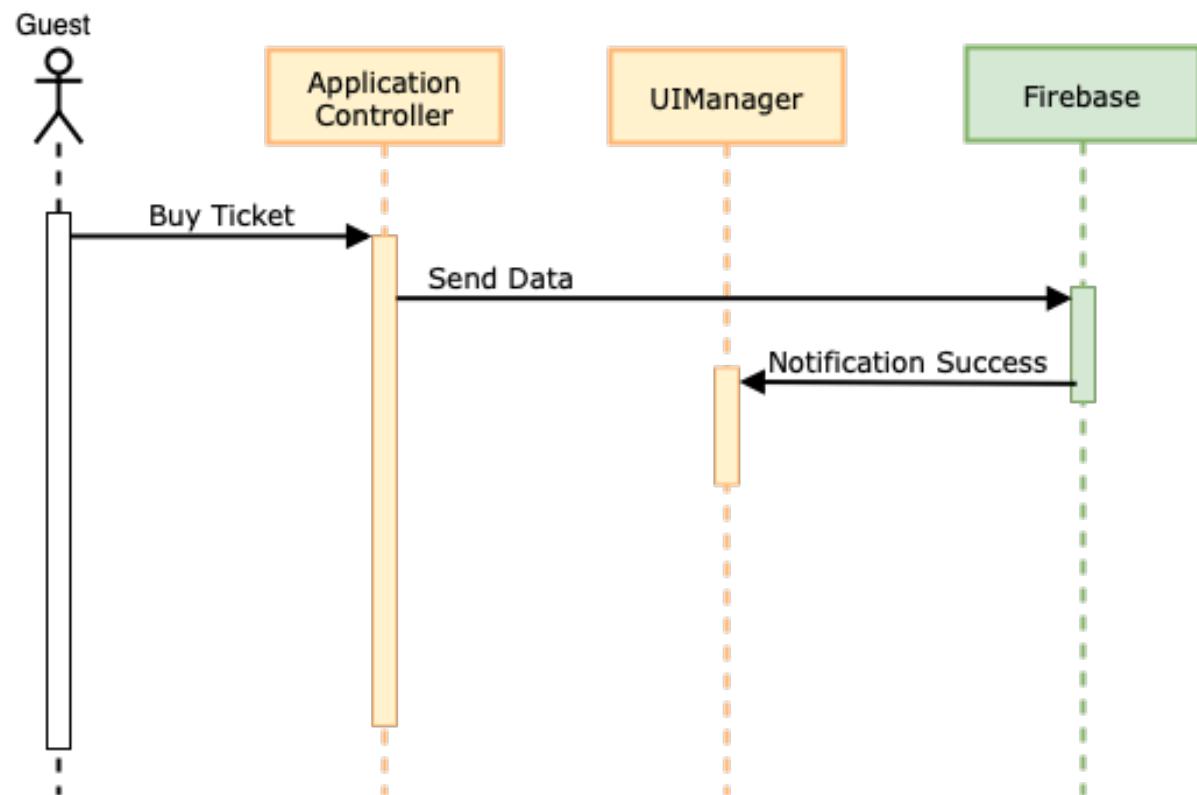


Buy Ticket

La procedura di "Buy Ticket" inizia dopo che l'utente ha completato la composizione della propria schedina.

Una volta premuto il bottone di buy l'applicazione controllerà la correttezza della schedina ed effettuerà l'acquisto

Tale procedura presuppone che l'utente sia già autenticato altrimenti verrà visualizzato un messaggio di errore.

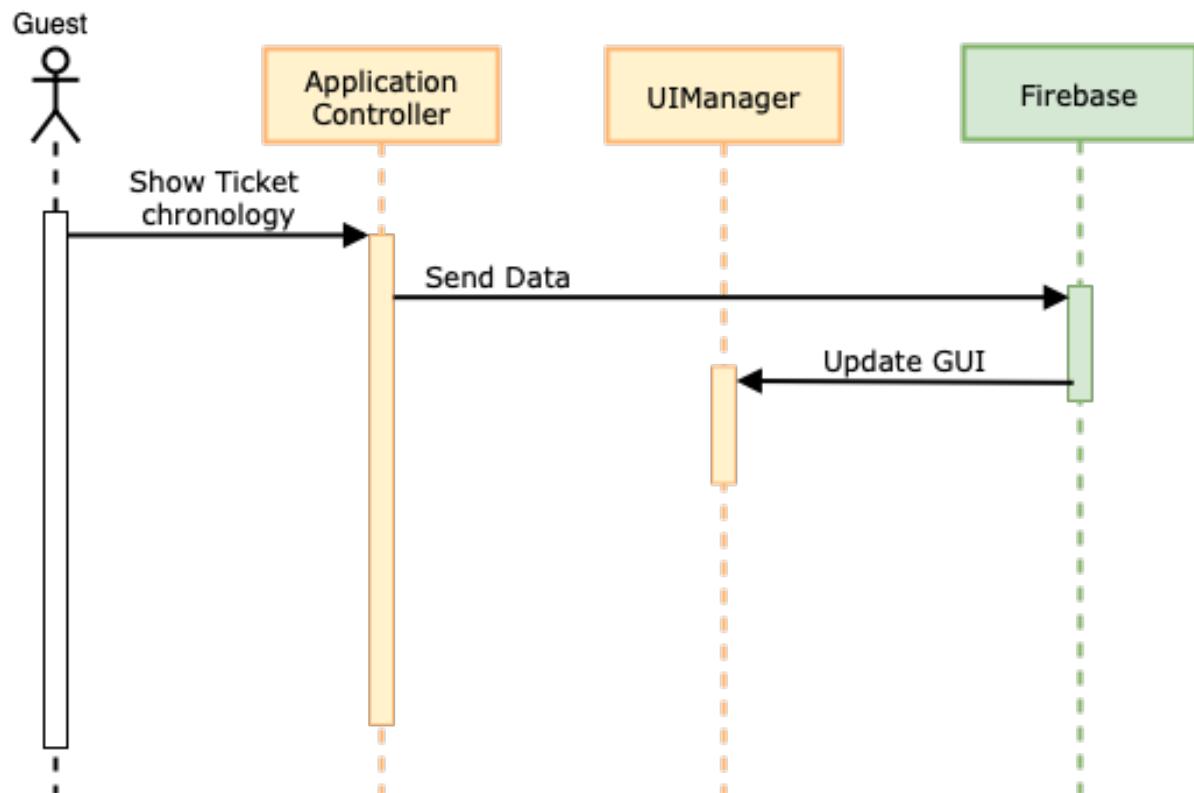


Ticket Chronology

La procedura di "Ticket Chronology" inizia quando l'utente si trova nella sezione "Bet" dopo aver premuto il relativo bottone del Side Menu.

Dopo che l'utente preme sul tasto "Cronologia" del NavBar l'applicazione accederà ai dati presenti nel database di Firebase recuperando tutte le schedine acquistate in precedenza dall'utente e mostrandole a schermo

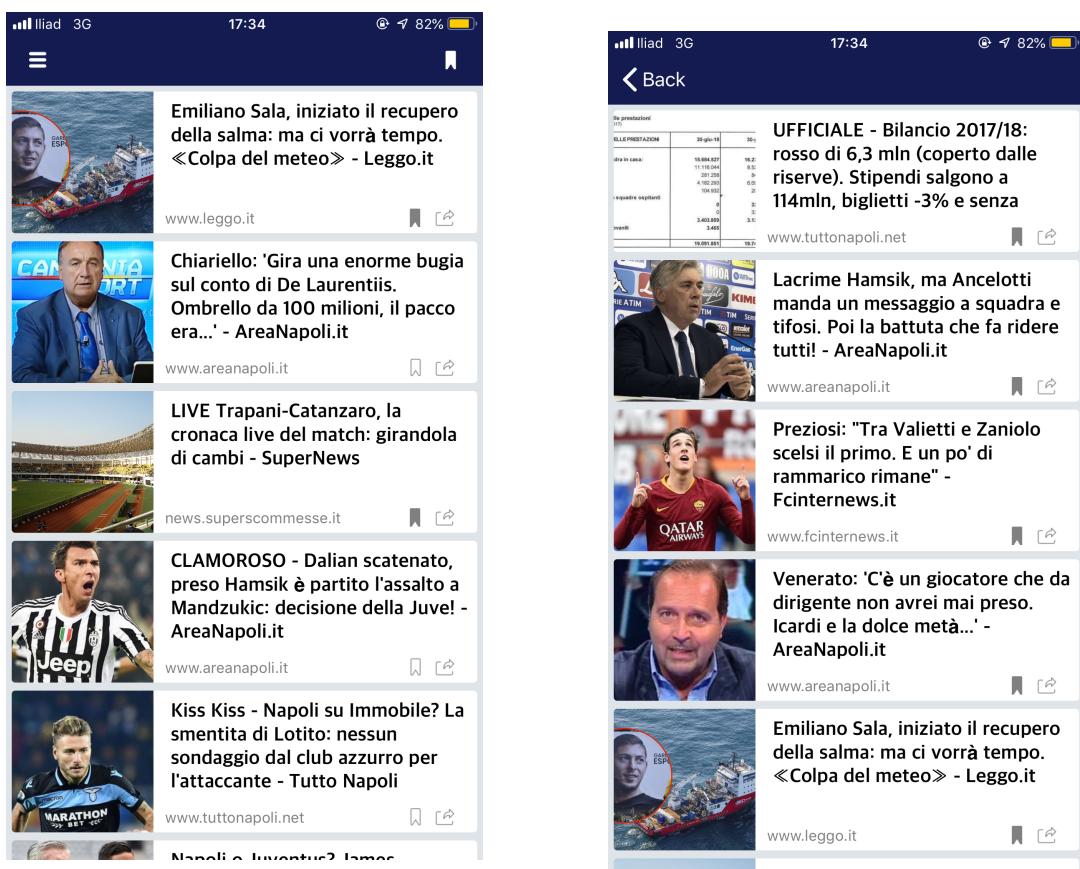
Tale procedura presuppone che l'utente sia già autenticato altrimenti verrà visualizzato un messaggio di errore.



5 User Interfaces

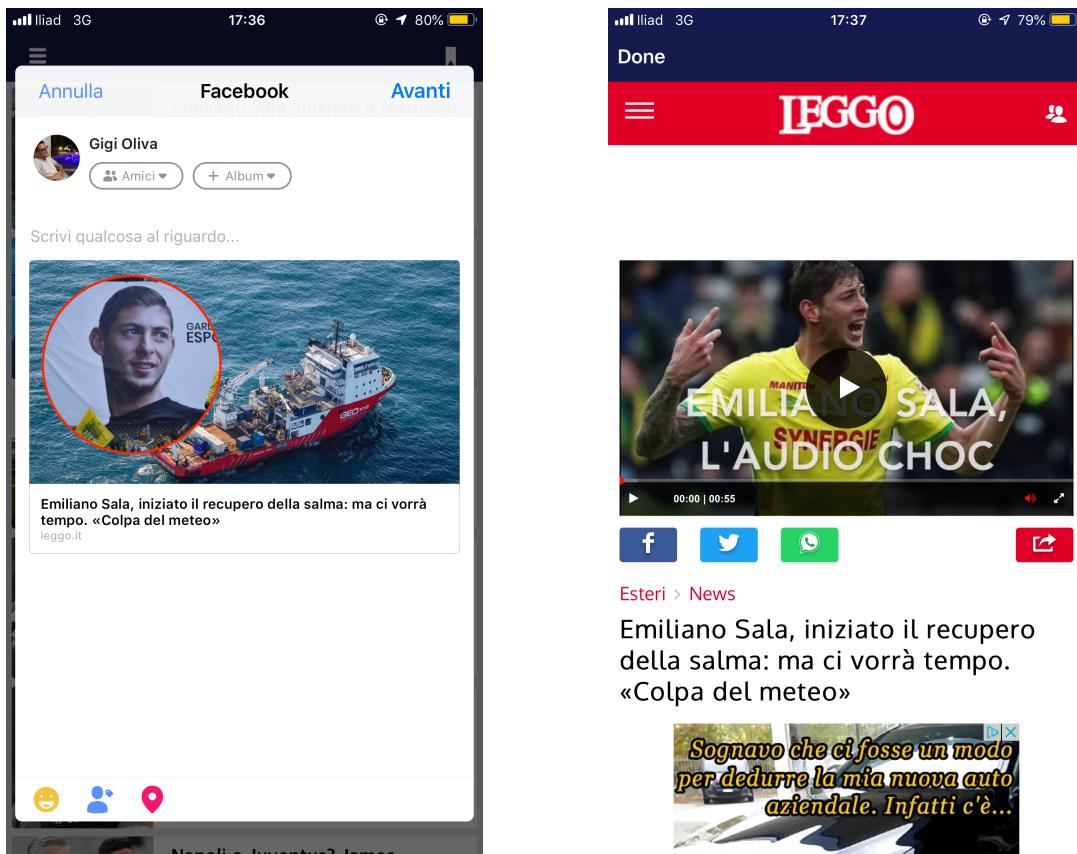
In this chapter will be discussed and displayed the user interface of each page highlighting the functionalities of the iSport application.

News



- (a) In questa schermata vengono mostrate tutte le notizie con la possibilità di salvarle o pubblicarle
- (b) In questa schermata vengono mostrate tutte le notizie salvate

5 User Interfaces



- (a) In questa schermata viene mostrata la con- (b) Use case relativo alla visualizzazione e ma- divisione di news nיפולazione delle notizie

6 External Services and Libraries

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

Per il parsing dei dati JSON è stato usato il componente JSONDecoder nativo dellUIKit.

Facebook SDK

Per la gestione del login sono state utilizzate le Facebook SDK. Quando un utente vuole autenticarsi l'applicazione esegue la chiamata al servizio di Facebook mediante:

```
@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!")
        }
    }
}
```

Tale metodo rimanda al sito o all'applicazione di Facebook e una volta autenticato si viene reindirizzati ad "iSport" che andrà a memorizzare il token fornитогли.

Per la pubblicazione delle notizie invece viene utilizzato il servizio di "Graph API":

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

Per utilizzare il servizio di Facebook sono stati inclusi i seguenti pod:

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

Firebase SDK

Per l'accesso al database vengono utilizzate le SDK di Firebase. In particolare viene utilizzato il login tramite le credenziali di Facebook in modo da avere un login unico.

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

Per l'accesso al database è stato usato il DatabaseManager incluso nelle SDK:

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

Per utilizzare il servizio di Firebase sono stati inclusi i seguenti pod:

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

Scaledrone

Per il servizio di chat in tempo reale sono state utilizzate le API di Scaledrone. Per la connessione alla room viene utilizzata la seguente estensione:

```
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 ?? "")
    }
}
```

Mentre per la ricezione dei messaggi:

```
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)
    }
}
```

Per utilizzare il servizio di Scaledrone sono stati inclusi i seguenti pod:

¹ pod 'Scaledrone', '^> 0.3.0'

NewsAPI

Quando un utente vuole visualizzare le principali notizie del giorno l'applicazione esegue le seguenti operazioni:

- Esegue una richiesta GET al l'endpoint fornito dal servizio
- Se la risposta è valida effettua il parsing del contenuto JSON
- Crea le celle della tabella contenente le informazioni fornite dal servizio e parsate
- Scarica in maniera asincrona le immagini di anteprima

L'endpoint utilizzato è "https://newsapi.org/v2/top-headlines?country=it&category=sports &apiKey=API_KEY". La risposta fornita dal servizio è:

6 External Services and Libraries

```
{
  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"
      },
    }
  ]
}
```

Per il parsing quindi è stata usata la seguente struttura dati:

```
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

Quando un utente vuole visualizzare i risultati delle partite del giorno l'applicazione esegue le seguenti operazioni:

- Esegue una richiesta GET al l'endpoint fornito dal servizio
- Se la risposta è valida effettua il parsing del contenuto JSON
- Crea le celle della tabella contenente le informazioni fornite dal servizio e parsate
- Aggiorna le informazioni ogni 30 secondi per mantenere i dati in tempo reale

L'endpoint utilizzato per ottenere i risultati è "https://apifootball.com/api/?action=get_events". La risposta fornita dal servizio è:

6 External Services and Libraries

```
[  
  {  
    "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": [  
    "cards": [  
    "lineup": {  
      "statistics": [  
    ]  
  ]  
}
```

Per il parsing quindi è stata usata la seguente struttura dati:

```
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: [Goalista]  
    let cards: [Cardista]  
    let statistics: [Statistic]  
    let lineup: Formazione  
}  
  
struct Goalista: Decodable {  
    let time: String?  
    let homeScorer: String?  
    let score: String?  
    let awayScorer: String?  
}  
  
struct Cardista: 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?  
}
```

Per ottenere le previsioni della partita l'endpoint utilizzato è ["https://apifootball.com/api/?action=get_predictions"](https://apifootball.com/api/?action=get_predictions). La risposta fornita dal servizio è:

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"  
  }  
]
```

Per il parsing quindi è stata usata la seguente struttura dati:

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

Per ottenere le quote delle partite l'endpoint utilizzato è ["https://apifootball.com/api/?action=get_odds"](https://apifootball.com/api/?action=get_odds). La risposta fornita dal servizio è:

```
[  
  {  
    "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"  
  }  
]
```

Per il parsing quindi è stata usata la seguente struttura dati:

```
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