First Semester Project

Software Engineering

Alfonso Pedro Ridao 308833

Supervisor:

Mona Andersen

Allan Henriksen

Software Engineering

First Semester

1st of June 2020

Table of content

1 Introduction 1
2 Analysis 2
2.1 Functional Requirements 2
2.2 Non-Functional Requirements 3
2.3 Use case Diagram 3
2.4 Use case Description 4
2.5 Activity Diagram 6
2.6 Domain Model 7
3 Design 8
3.1 Diagrams 8
3.2 GUI 10
3.3 Website. 13
4 Implementation 15
4.1 GUI 15
4.2 Web 20
5 Test 21
6 Results 22
7 Discussion 24
8 Conclusion 25
9 Sources of information 26
Appendices 27

Abstract

Via Club project is a single-user system created to improve how the club manager controls the information about players and matches and shares this information with the web page.

Once the interview with the final user was done, the development of the system started with the analysis. In this process was considered all the requirements and developed the use case diagrams, use case descriptions, activity diagrams, and domain model.
The implementation phase was created In Java, following the design model previously created. The whole GUI was made using JavaFX and Scene Builder.
The whole system was tested, considering the functional requirement.

As a final result, there is an intuitive and simple system with a friendly user interface. The system fulfills the requirements and more. The system allows management of the information and stores the data in binary files and XML and TXT.

1 Introduction

Technology enables us to automate numerous processes, which thereby increases our productivity. This is possible because it allows us to use fewer resources. Technology also makes it easy to store more information while maintaining the integrity of that information. We are better able to store sensitive and confidential information to make it less vulnerable to a data breach. (Chron, 2018)

Even that an awful lot of technology has been designed without older adult capabilities in mind, like smartphones — with small screens and small print and small icons — which make those devices difficult to use for older adults. (The Garage HP, 2019).

VIA Club is a football team that Mr. Bob Oldenuff has managed for the last 31 years. He has been preparing the list of players for each match on paper for his whole career. The list includes all the players on the pitch and the players on the bench. The list can be different depending on what kind of match they are playing. Constantly changing the player list represents a problem when done on paper. (Appendix A, Project description).

Currently, the solutions on the market like "KampKlar" (DBU, 2021.) are not fit the client's desires since most of them are either overly complicated or feature things that the client does not want in his program, such as being able to access the program remotely and needing a login with a password.

The manager expressed that he feels constant pressure from the people around him. He has a stressful job. He is complaining about the fans and player's attitude. He knows that probably switching to a new way of making the list would be for the best. (Appendix A, Project description).

Furthermore, he realizes that the actual process consumes too much time. The list could be lost or destroyed, and getting copies could be a real problem. He also must remember if anyone is suspended or injured or if the player has been benched for too many games in a row. Also, he must consider which player can play in which position according to each player's skill set. (Appendix A, Project description).

The manager also states his technical desires. He does not want to log in with a password to the system, and he wants something easy, with a user interface and a mouse control. He also mentioned that the club's web page is not updated in content and new technologic standards. (Appendix A, Project description).

Based on the needs presented above, analysis, design, implementation, and test will follow.

2 Analysis

To help the Manager of VIA Club, a system will be developed based on his requirement. Taking consideration of the interview, there is a split between functional and non-functional requirements. Each of them has been categorized by importance.

2.1 Functional Requirements

Critical Priority:

1. As a manager, I want to visualize a list of all the players in the system, in order to know my team.
2. As a manager, I want to visualize a list of all the matches, the previous and the upcoming.
3. As a manager, I want to visualize the player list of players for a specific match, in order to make the players know
4. As a manager, I want to create a new match giving the opponent, date, place, and type (Cup, League, Friendly).
5. As a manager, I want to add players into the match, in order to have my list updated.

High priority:

6. As manager, I want to select 11 "available" players to go to the pitch, and 5 "available" players to go to the bench if the match is "League."

7. As a manager, I want to select 11 "available" players to go to the pitch, and 6 "available" players to go the bench if the match is "Cup".
8. As a manager, I want to select 11 "available" or "suspended" players to go to the pitch, and unlimited "available" or "suspended" players to go the bench if the match is "Friendly".
9. As a manager, I want to register a new player with a full name, unique number, a list of positions that have been trained, and the Status (available, suspended, injured, unavailable).
10. As a manager, I want to modify the Status of a player (available, suspended, injured, unavailable).
11. As a manager, I want to modify the fields on a match (date, place, and kind of match). The type of match can be changed only if there are no conflicts with the number of players and Status
12. As a manager, I want to delete a match in order to have the list updated
13. As a manager, I want to modify the players' list in a specific match, in order to have the list updated.
14. As a manager, I want to export the data to a website in order to show the public the updated data.

Low priority:

15. As a manager, I want to modify any player record as long as another player does not already use the number.
16. As a manager, I want to delete a player from the list in order to have the list updated.
17. As a manager, I want to see how long the player has been playing without rest or how long he has been resting without play.

2.2 Non-Functional Requirements

18. Every update in the system includes writing to a file
19. The system needs to be implemented in JAVA
20. The system must have a GUI and be controlled by a mouse and keyboard.

2.3 Use case Diagram

The following use case diagram represents the relation between the user with the system. It is based on the functional requirement.

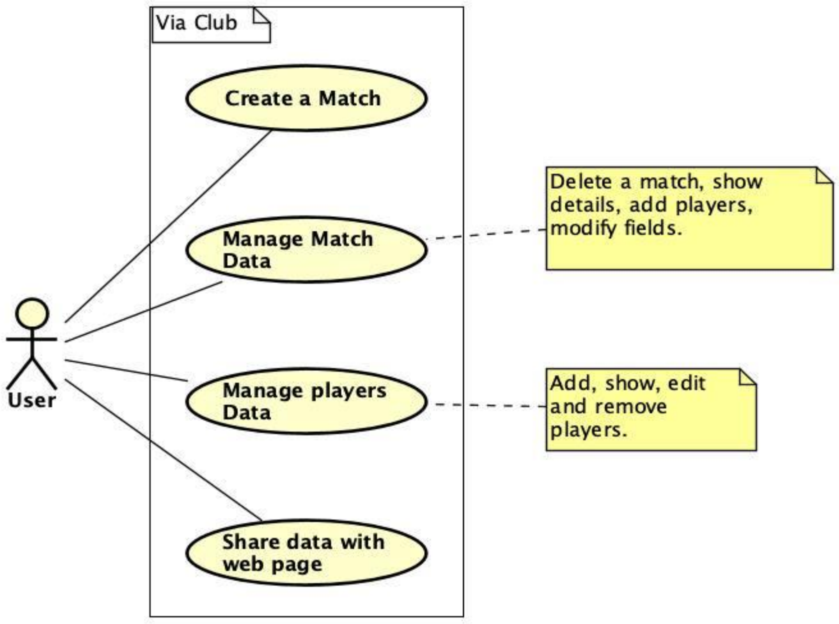

Via Club
Create a Match
Manage Match Data
Manage players Data
Share data with web page
User
Delete a match, show details, add players, modify fields.
Add, show, edit and remove players.

Figure 1 - Use Case Diagram

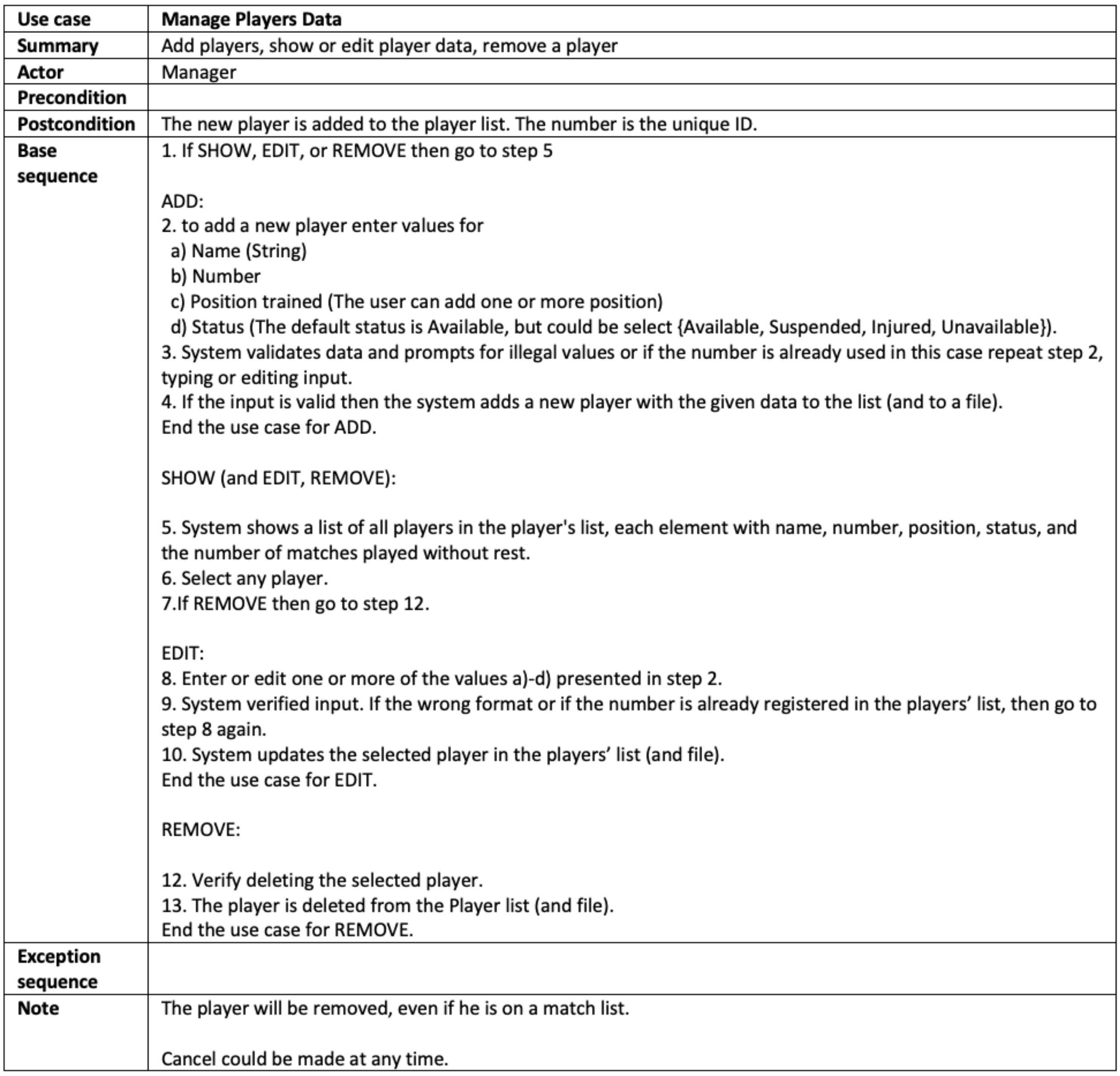
2.4 Use case Description

Four uses cases descriptions were made to extend the use cases diagram for each action (Create a match, manage match data, manage players' data, share data with the web page).

Concerning the match, "Create a match" create the match, selecting before the data—opponent, date, kind of match, and place—.
"Manage Match Data" allows to add/delete players to the match, editing the match, or deleting the match.

Manage Players Data allows interaction with the players' data. Create new players, delete a player, or modify any field of an existing player. This use case description is presented step by step in the following figure. To see all the use case descriptions, please look at Appendix B.

Share Data with page related to the action to create a file to be load by the web user.



Use case	**Manage Players Data**
Summary	Add players, show or edit player data, remove a player
Actor	Manager
Precondition	
Postcondition	The new player is added to the player list. The number is the unique ID.
Base sequence	1. If SHOW, EDIT, or REMOVE then go to step 5 ADD: 2. to add a new player enter values for a) Name (String) b) Number c) Position trained (The user can add one or more position) d) Status (The default status is Available, but could be select {Available, Suspended, Injured, Unavailable}). 3. System validates data and prompts for illegal values or if the number is already used in this case repeat step 2, typing or editing input. 4. If the input is valid then the system adds a new player with the given data to the list (and to a file). End the use case for ADD. SHOW (and EDIT, REMOVE): 5. System shows a list of all players in the player's list, each element with name, number, position, status, and the number of matches played without rest. 6. Select any player. 7.If REMOVE then go to step 12. EDIT: 8. Enter or edit one or more of the values a)-d) presented in step 2. 9. System verified input. If the wrong format or if the number is already registered in the players' list, then go to step 8 again. 10. System updates the selected player in the players' list (and file). End the use case for EDIT. REMOVE: 12. Verify deleting the selected player. 13. The player is deleted from the Player list (and file). End the use case for REMOVE.
Exception sequence	
Note	The player will be removed, even if he is on a match list. Cancel could be made at any time.

Figure 2 - Use case description to manage players data

The link between the requirement and the use case can be view in detail in Appendix C.

2.5 Activity Diagram

The following activity diagram shows the process to manage a Player. The first step is to create a player or show a current player.
To create a player, the user must insert a name, number, position, and Status (Status could be available, suspended, injured, or unavailable).
To take any action on an existing player, the user must select "show." Once the user is viewing the player, could choose options to remove the player or modify any existing field.

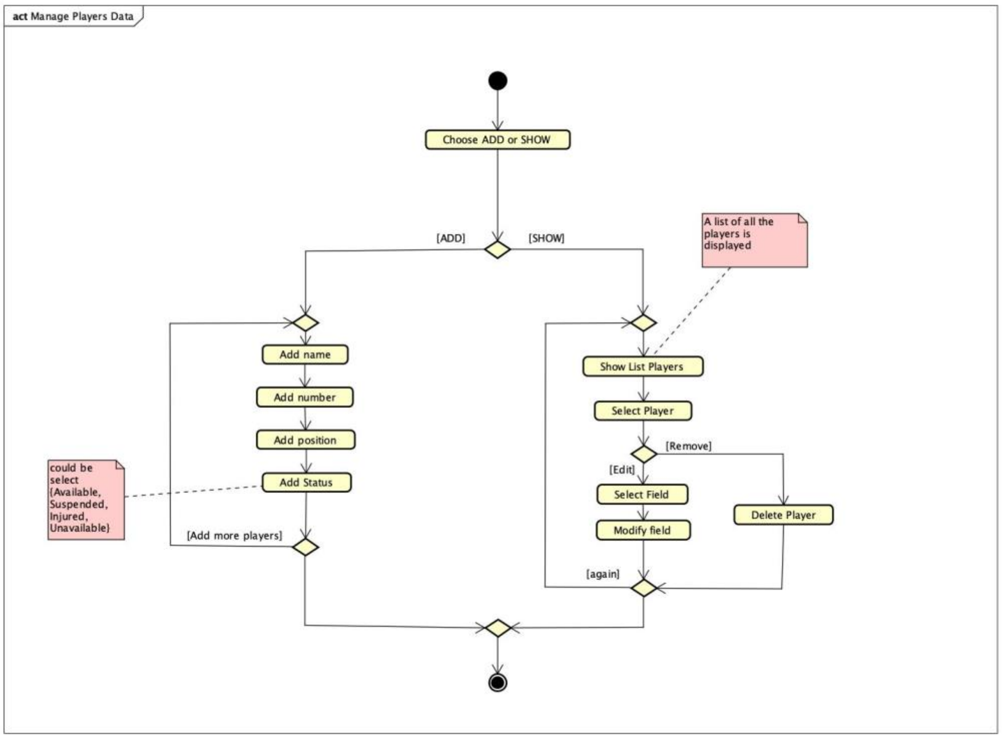


Figure 3 - Activity Diagram for Manage players data

To see all the activity diagrams, please look in Appendix D.

2.6 Domain Model

The following domain model represents the relation between the classes.
The manager class contains players and matches. Every match also contains a date and player. Each player contains a status and also a date.

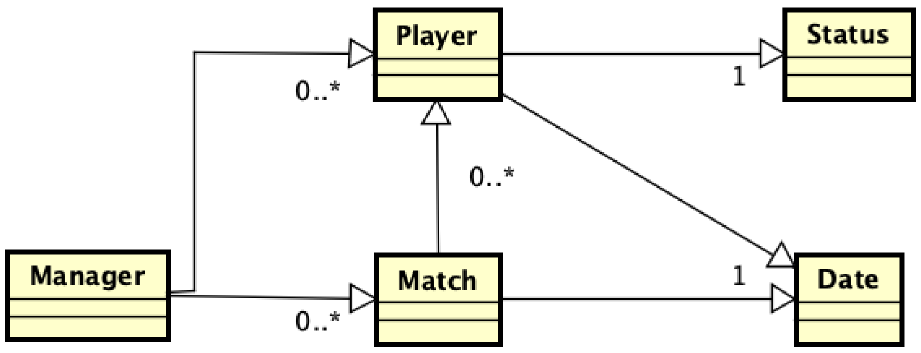
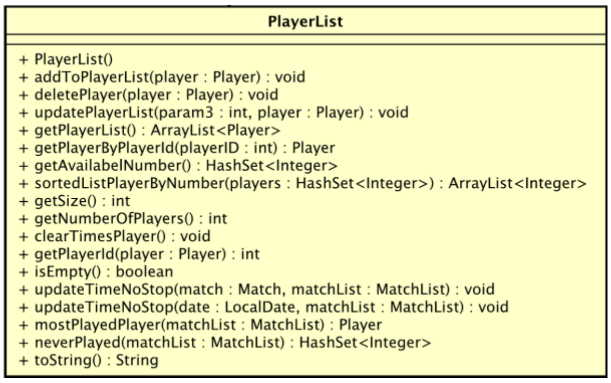


Figure 4 - Domain model

3 Design

3.1 Diagrams

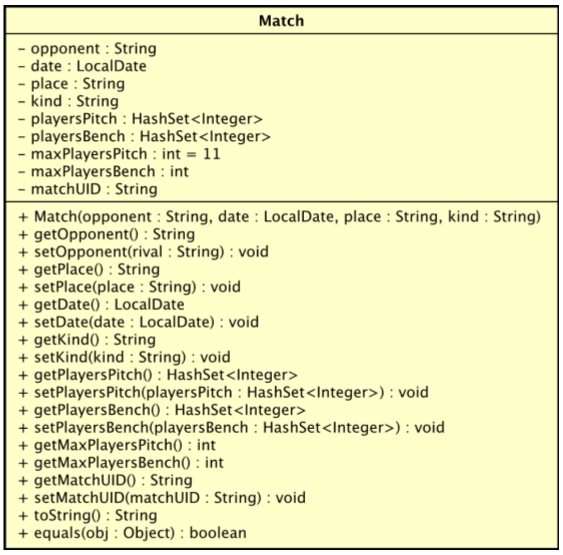
Based on the domain model, the class diagram was created.



```
PlayerList

+ PlayerList()
+ addToPlayerList(player : Player) : void
+ deletePlayer(player : Player) : void
+ updatePlayerList(param3 : int, player : Player) : void
+ getPlayerList() : ArrayList<Player>
+ getPlayerByPlayerId(playerID : int) : Player
+ getAvailabelNumber() : HashSet<Integer>
+ sortedListPlayerByNumber(players : HashSet<Integer>) : ArrayList<Integer>
+ getSize() : int
+ getNumberOfPlayers() : int
+ clearTimesPlayer() : void
+ getPlayerId(player : Player) : int
+ isEmpty() : boolean
+ updateTimeNoStop(match : Match, matchList : MatchList) : void
+ updateTimeNoStop(date : LocalDate, matchList : MatchList) : void
+ mostPlayedPlayer(matchList : MatchList) : Player
+ neverPlayed(matchList : MatchList) : HashSet<Integer>
+ toString() : String
```

Figure 5 – PlayerList class

Inside the Model package, we find classes representing the Match and The Player, with the fields and methods related to the single object. And then classes for the list. PlayerList and MatchList have all the functionality to control the different lists.



```
Match

- opponent : String
- date : LocalDate
- place : String
- kind : String
- playersPitch : HashSet<Integer>
- playersBench : HashSet<Integer>
- maxPlayersPitch : int = 11
- maxPlayersBench : int
- matchUID : String

+ Match(opponent : String, date : LocalDate, place : String, kind : String)
+ getOpponent() : String
+ setOpponent(rival : String) : void
+ getPlace() : String
+ setPlace(place : String) : void
+ getDate() : LocalDate
+ setDate(date : LocalDate) : void
+ getKind() : String
+ setKind(kind : String) : void
+ getPlayersPitch() : HashSet<Integer>
+ setPlayersPitch(playersPitch : HashSet<Integer>) : void
+ getPlayersBench() : HashSet<Integer>
+ setPlayersBench(playersBench : HashSet<Integer>) : void
+ getMaxPlayersPitch() : int
+ getMaxPlayersBench() : int
+ getMatchUID() : String
+ setMatchUID(matchUID : String) : void
+ toString() : String
+ equals(obj : Object) : boolean
```

Figure 6 - Match Class

Also, there is a class called SystemStatus that sets and shows the system status of the player. That Status is only for system use.

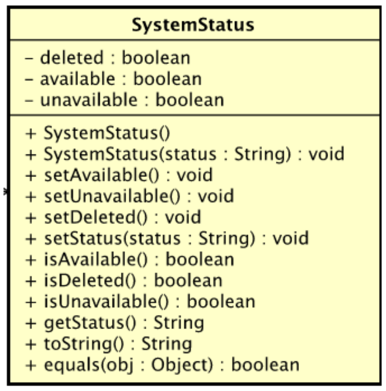


Figure 7 - SystemStatus class

Inside the model package, there is a ControlManager class. This class act like a connector between myFileHandler (controlling the opening and close of the files of the whole system) and AdminPanellController (the main GUI controller). This class has all its methods as statics methods, allowing the execution of the method without creating the object first. (Tony Gaddis, 2015).

Finally, besides MyFileHandler, inside the "Util" package is located a static class called AlertControl. This class launches different kinds of Alert messages all around the system and is called by the GUI controllers.

To see the complete class diagram, please look into Appendix E.

The sequence diagram shows the flow to create a new player.

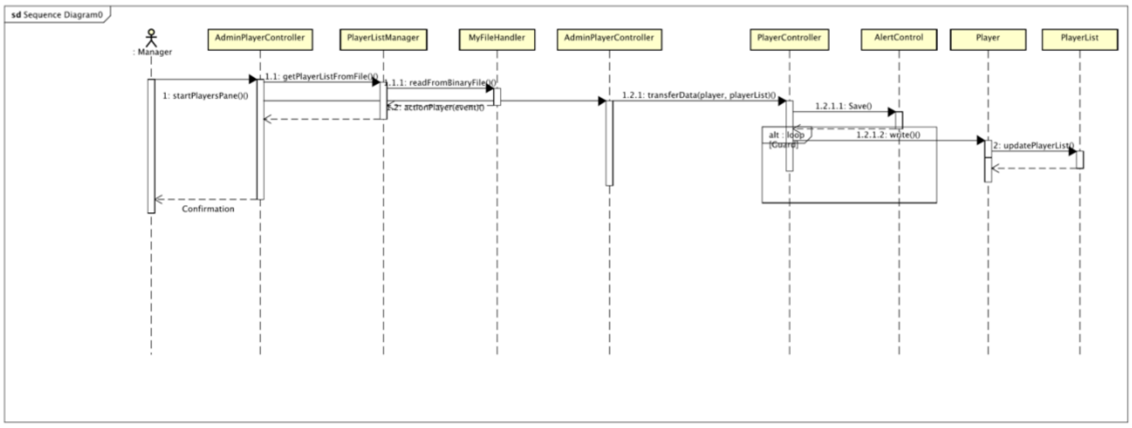


Figure 8 - Sequence Diagram

First, the manager access the "Players" tab. Once he accesses the player panel, the playlist object is returned from calling to PlayerListManager, and then MyFileHandler. From this player tab, the user clicks in "Add Player." The actionPlayer shoots the new window with the new controller PlayerController. The system will validate the fields, and if there are conflicts, AlertControl is shooted.

After validating the fields, the system allows the user to press "Save" to create a new player object and insert it at the end of the ArrayList<Players>, inside de playerList object.

To see the diagram in detail, please look into Appendix F

To see the original ASTAH diagrams, please look into Appendix G.

3.2 GUI

The graphics interface is simple. There is the main window, with an Admin panel, where the user can select between 3 different tabs. The first tab is just a reader tab, and swipe information between 8 different cards to show some vital information to the user. Those cards are displayed or disappear depends if there is some content to show or not.

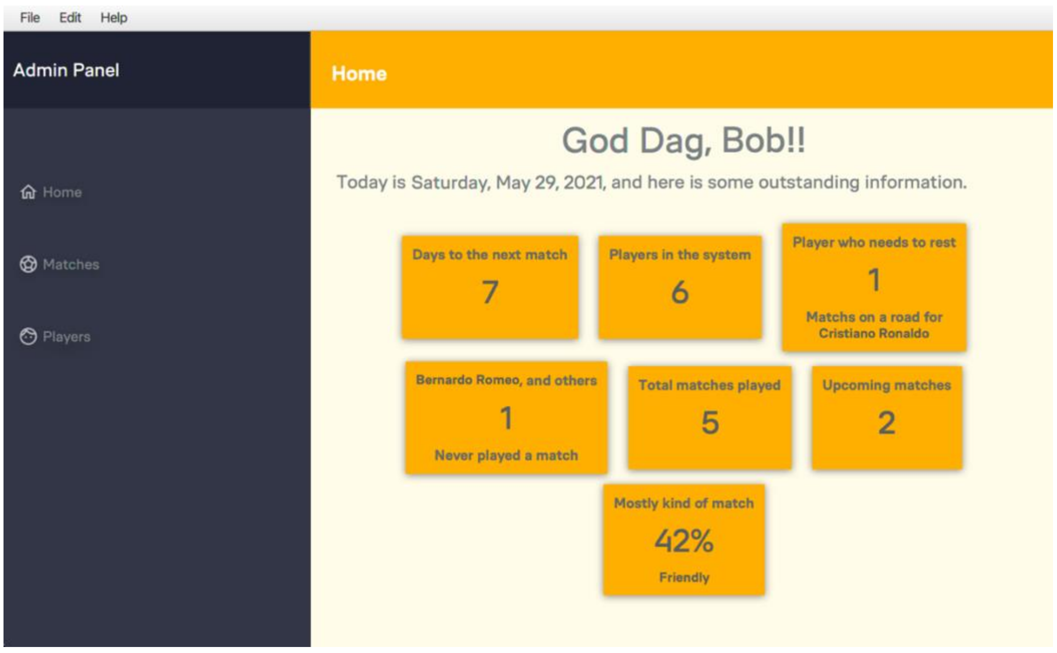


Figure 9 - Main View

If the user selects the "Matches" tab, different options appear to take actions related to the match. "Edit" or "Add" match will open a new window. Meanwhile, the new window is opened, the user could not interact with the main window.

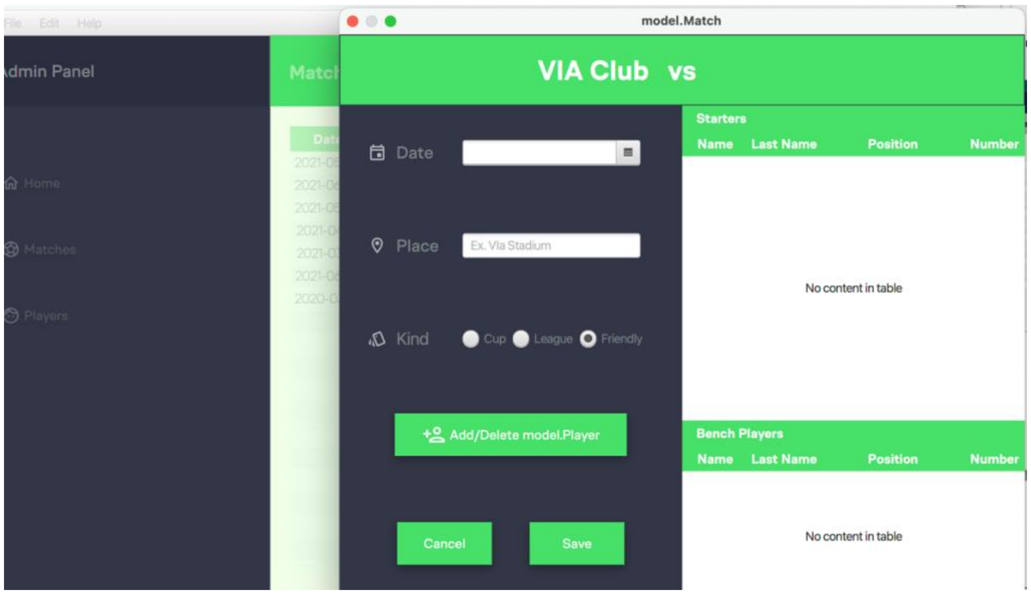
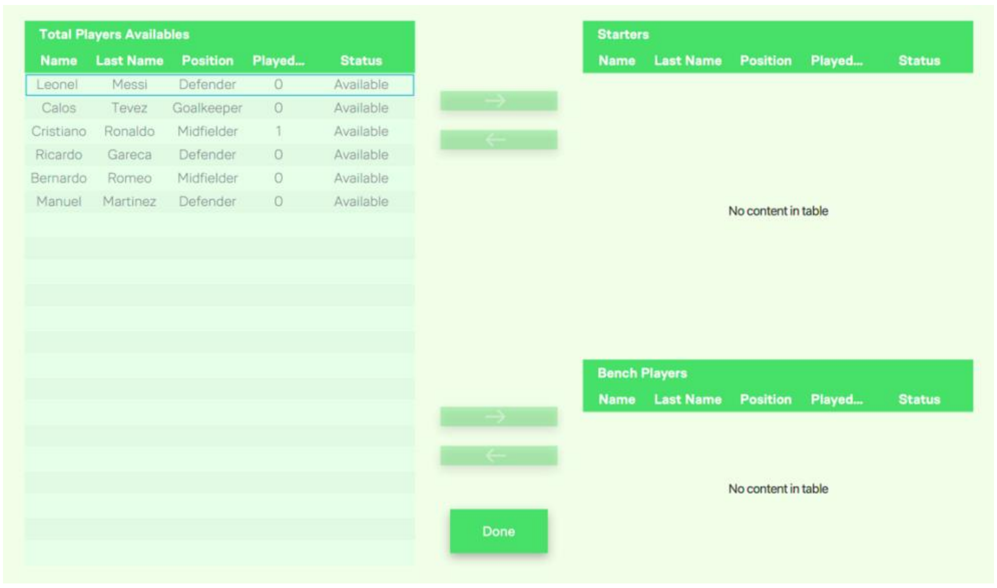


Figure 10 - Add match

If the user clicks on **"**Add/Delete Player,**"** a third window will appear with three tables related to the complete list, the starters list, and the bench list.



Total Players Availables

Name	Last Name	Position	Played...	Status
Leonel	Messi	Defender	0	Available
Calos	Tevez	Goalkeeper	0	Available
Cristiano	Ronaldo	Midfielder	1	Available
Ricardo	Gareca	Defender	0	Available
Bernardo	Romeo	Midfielder	0	Available
Manuel	Martinez	Defender	0	Available

Starters

Name	Last Name	Position	Played...	Status

No content in table

Bench Players

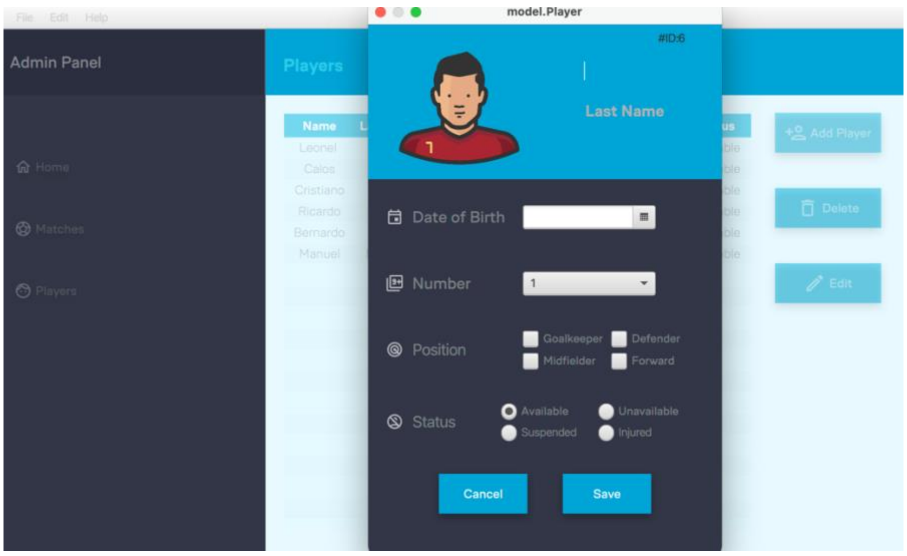
Name	Last Name	Position	Played...	Status

No content in table

Figure 11 - adding players to a match

The arrow buttons "move" a player from tab to tab. The buttons appear disabled if the proper table is not selected.

Something similar occurs when the user clicks the "Players" tab and then selects "Add Player" or "Edit" option.



model.Player

#ID:6

Last Name

Date of Birth

Number: 1

Position: Goalkeeper, Defender, Midfielder, Forward

Status: [✓ Available] Unavailable, Suspended, Injured

Figure 12 - Add Players

Inside the "Menu," among other options, we find the "Save" option. This option is only available if the user accessed any window where it is possible to make changes. Also, if this option is available, once we click on "Quit," the system will ask the user if the changes must be saved.

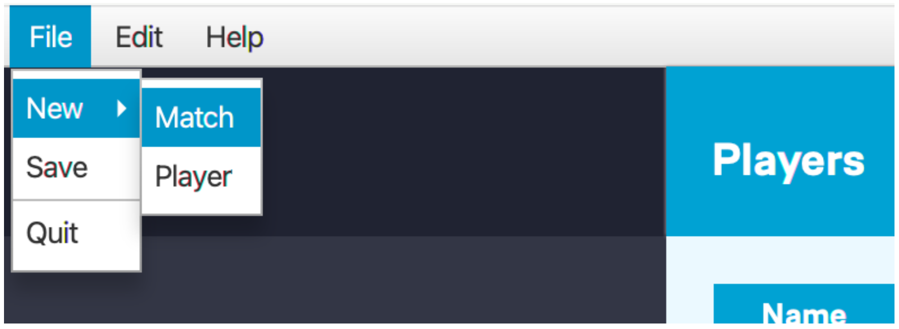
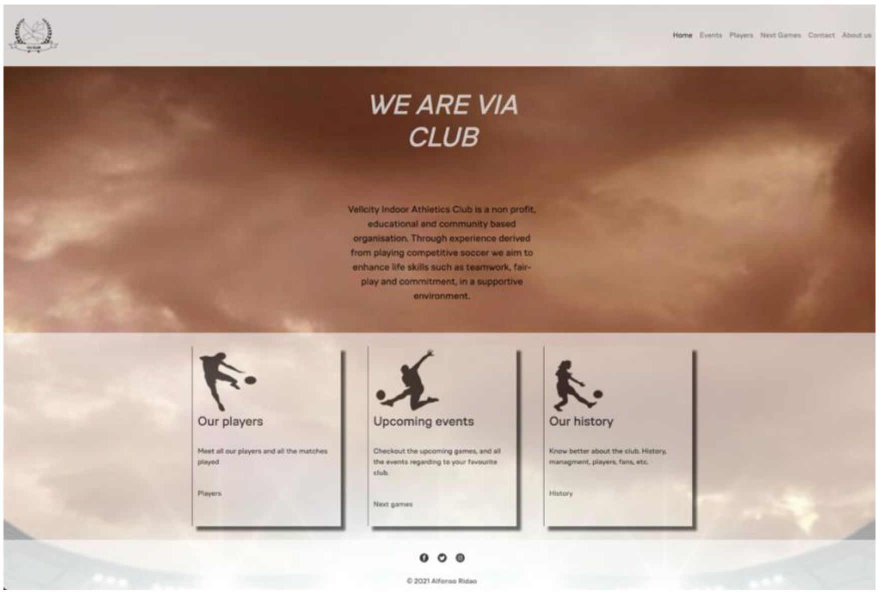


Figure 13 - File Options

To have complete details of the functions, please visit the user guide in Appendix H.

3.3 Website.

The website is simple and with a responsive design. The navbar is made with bootstrap, and it changes the format depending on the type of screen. Flex-box was used to show the content in a responsive way, solving tricky problems, including how to position, center, or dynamically resize elements on a page. (Benjamin LaGrone, 2013).



WE ARE VIA CLUB

Vellcity Indoor Athletics Club is a non profit, educational and community based organisation. Through experience derived from playing competitive soccer we aim to enhance life skills such as teamwork, fair-play and commitment, in a supportive environment.

Our players

Meet all our players and all the matches played

Players

Upcoming events

Checkout the upcoming games, and all the events regarding to your favourite club.

Next games

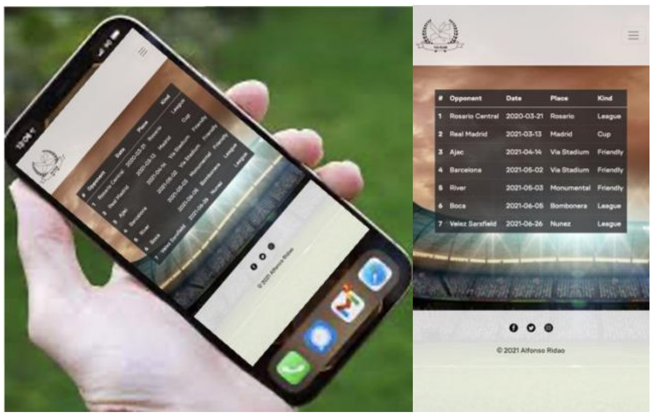
Our history

Know better about the club. History, managment, players, fans, etc.

History

© 2021 Alfonso Ridao

Figure 14 - Index page

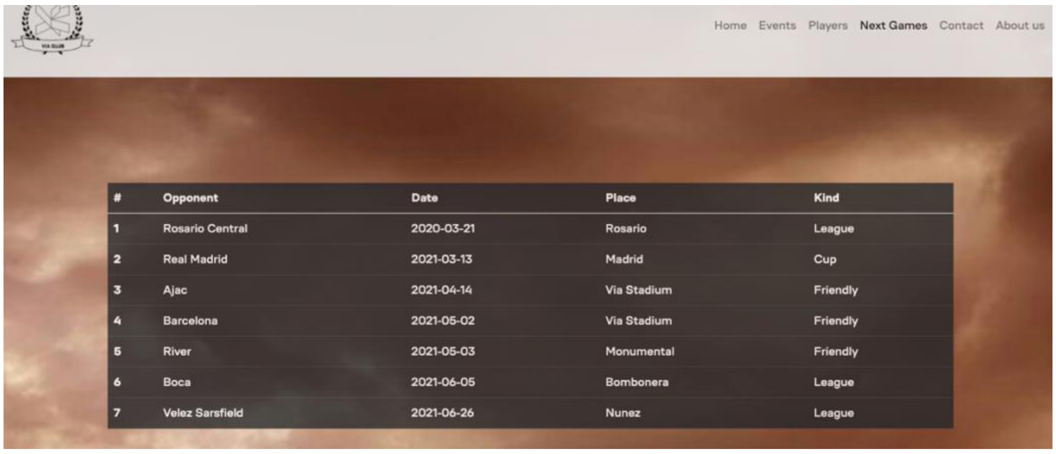


#	Opponent	Date	Place	Kind
1	Rosario Central	2020-03-21	Rosario	League
2	Real Madrid	2021-03-13	Madrid	Cup
3	Ajac	2021-04-14	Via Stadium	Friendly
4	Barcelona	2021-05-02	Via Stadium	Friendly
5	River	2021-05-03	Monumental	Friendly
6	Boca	2021-06-05	Bombonera	League
7	Velez Sarsfield	2021-06-26	Nunez	League

© 2021 Alfonso Ridao

Figure 15 – Match List in a mobile phone.

The "Next Games" tab reads the information of the file "match-list.xml" created previously in the Java System and displays it as a list, sorted by date.



#	Opponent	Date	Place	Kind
1	Rosario Central	2020-03-21	Rosario	League
2	Real Madrid	2021-03-13	Madrid	Cup
3	Ajac	2021-04-14	Via Stadium	Friendly
4	Barcelona	2021-05-02	Via Stadium	Friendly
5	River	2021-05-03	Monumental	Friendly
6	Boca	2021-06-05	Bombonera	League
7	Velez Sarsfield	2021-06-26	Nunez	League

Figure 16 - Match list

4 Implementation

Based on the design outputs, an implementation process followed. The implementation process began by implementing the basic classes from the class diagram.

The following example is how was implemented the case description showed in the analysis phase.

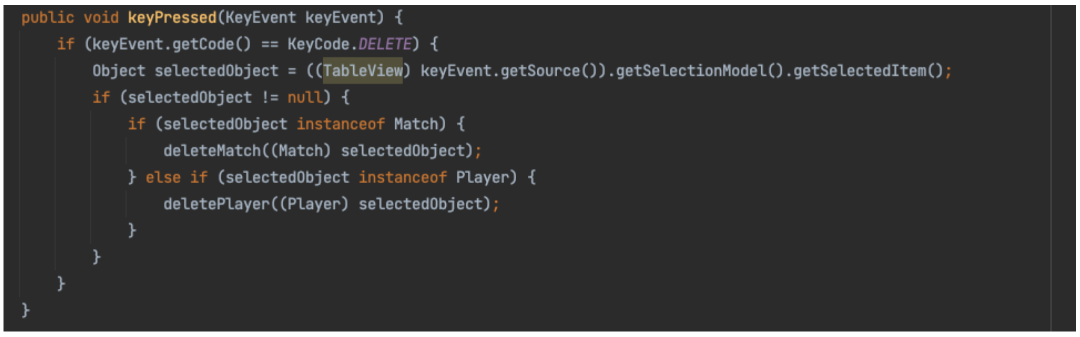
4.1 GUI

Manage Players Data.

The user selects between any of the Players' tab options to interact with the player. "Add Player," "Delete Player," or "Edit Player."

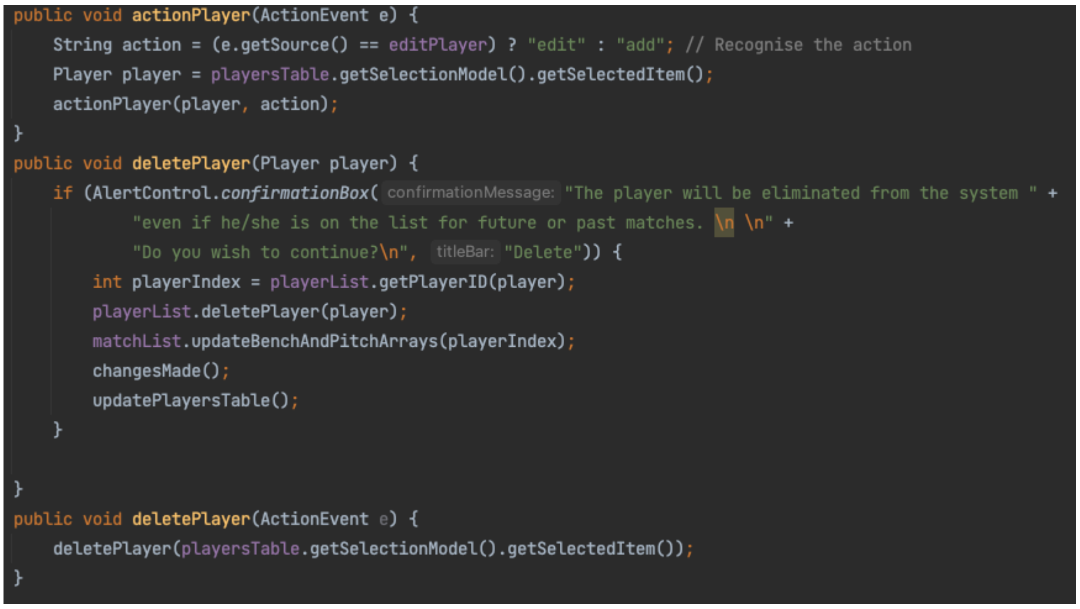
Delete player has its own event and calls to deletePlayer() in the player list to eliminate it. The function ask for a confirmation to AlertControl() before proceed to call .deletePlayer().

There are two ways to call deletePlayer(), and one is pressing the button delete. The other one is selecting the player on the tab and pressing the key DELETE on the keyboard.



```java
public void keyPressed(KeyEvent keyEvent) {
    if (keyEvent.getCode() == KeyCode.DELETE) {
        Object selectedObject = ((TableView) keyEvent.getSource()).getSelectionModel().getSelectedItem();
        if (selectedObject != null) {
            if (selectedObject instanceof Match) {
                deleteMatch((Match) selectedObject);
            } else if (selectedObject instanceof Player) {
                deletePlayer((Player) selectedObject);
            }
        }
    }
}
```

Figure 17 - Pressing DELETE event

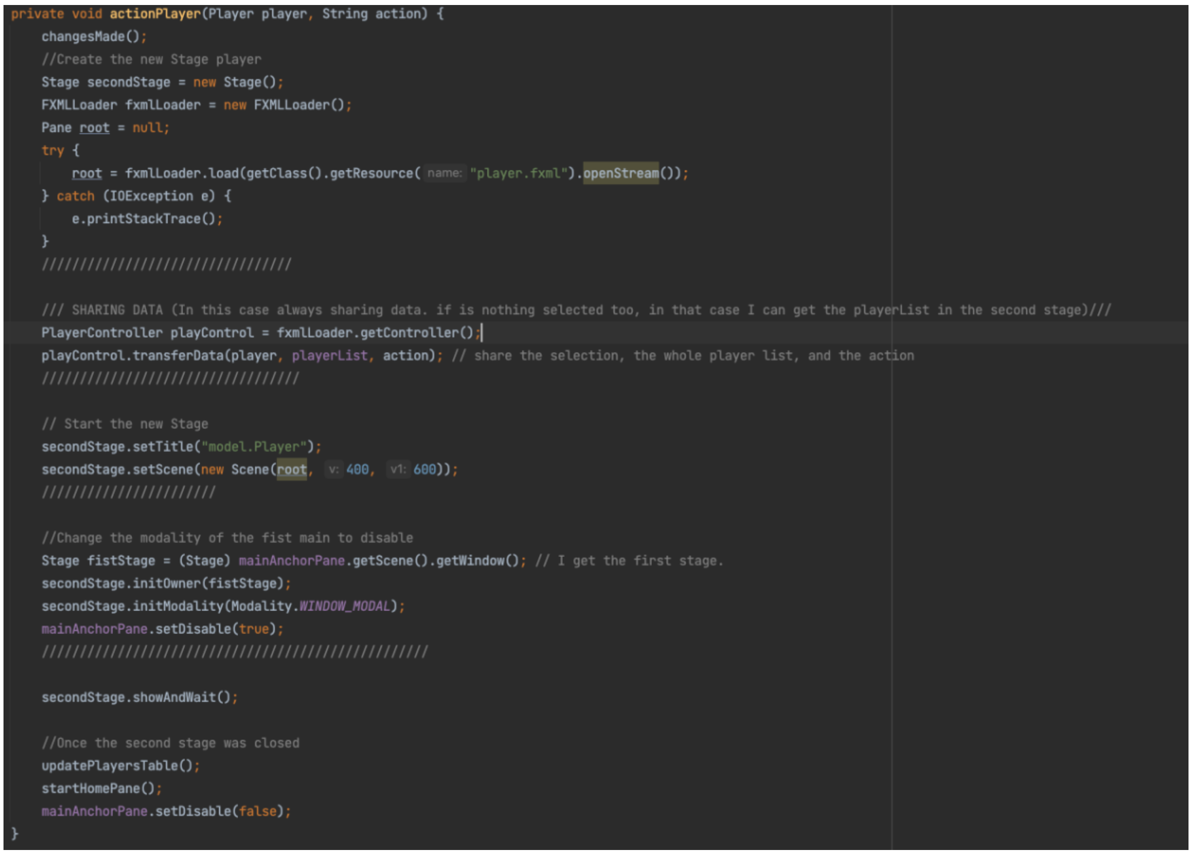


```java
public void actionPlayer(ActionEvent e) {
    String action = (e.getSource() == editPlayer) ? "edit" : "add"; // Recognise the action
    Player player = playersTable.getSelectionModel().getSelectedItem();
    actionPlayer(player, action);
}
public void deletePlayer(Player player) {
    if (AlertControl.confirmationBox( confirmationMessage: "The player will be eliminated from the system " +
            "even if he/she is on the list for future or past matches. \n \n" +
            "Do you wish to continue?\n",  titleBar: "Delete")) {
        int playerIndex = playerList.getPlayerID(player);
        playerList.deletePlayer(player);
        matchList.updateBenchAndPitchArrays(playerIndex);
        changesMade();
        updatePlayersTable();
    }

}
public void deletePlayer(ActionEvent e) {
    deletePlayer(playersTable.getSelectionModel().getSelectedItem());
}
```

Figure 18 - actionPlayer and deletePlayer

actionPlayer is shooted once "Edit Player" or "Add Player" was clicked, and call actionPlayer() with the player and the action. If the step is "Add Player," then the player is null.

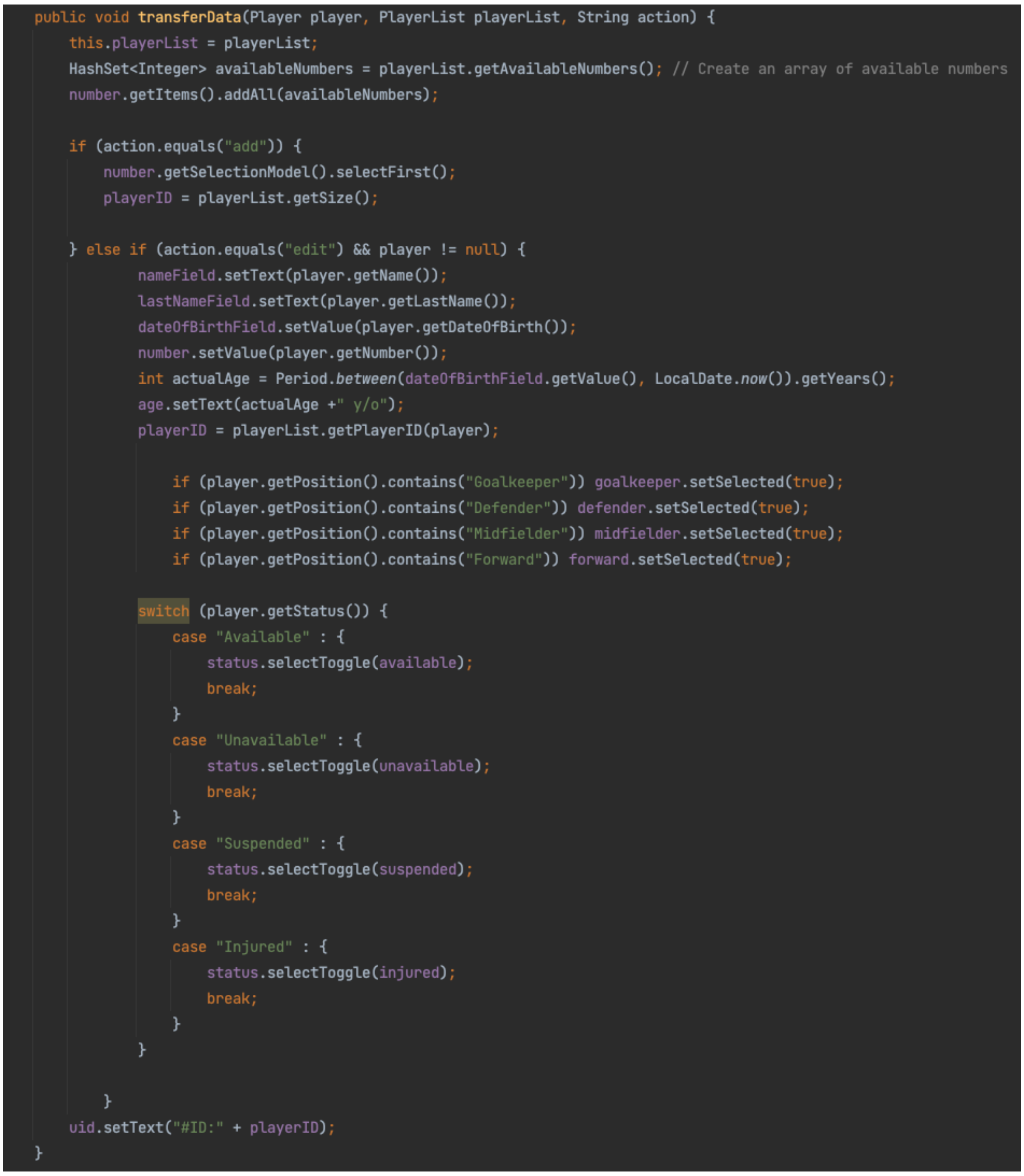
actionPlayer will be responsible for opening a new window, the "Player" window. First, call changesMade() just to be sure that a further modification occurs in the system. Then share the player, the player list, and the action to the new controller. After that, launch the second stage. Also, set the first stage as disable.



```java
private void actionPlayer(Player player, String action) {
    changesMade();
    //Create the new Stage player
    Stage secondStage = new Stage();
    FXMLLoader fxmlLoader = new FXMLLoader();
    Pane root = null;
    try {
        root = fxmlLoader.load(getClass().getResource( name: "player.fxml").openStream());
    } catch (IOException e) {
        e.printStackTrace();
    }
    /////////////////////////////////

    /// SHARING DATA (In this case always sharing data. if is nothing selected too, in that case I can get the playerList in the second stage)///
    PlayerController playControl = fxmlLoader.getController();
    playControl.transferData(player, playerList, action); // share the selection, the whole player list, and the action
    //////////////////////////////////

    // Start the new Stage
    secondStage.setTitle("model.Player");
    secondStage.setScene(new Scene(root,  v: 400,  v1: 600));
    //////////////////////

    //Change the modality of the fist main to disable
    Stage fistStage = (Stage) mainAnchorPane.getScene().getWindow(); // I get the first stage.
    secondStage.initOwner(fistStage);
    secondStage.initModality(Modality.WINDOW_MODAL);
    mainAnchorPane.setDisable(true);
    //////////////////////////////////////////////////

    secondStage.showAndWait();

    //Once the second stage was closed
    updatePlayersTable();
    startHomePane();
    mainAnchorPane.setDisable(false);
}
```

Figure 19 - actionPlayer in mainController

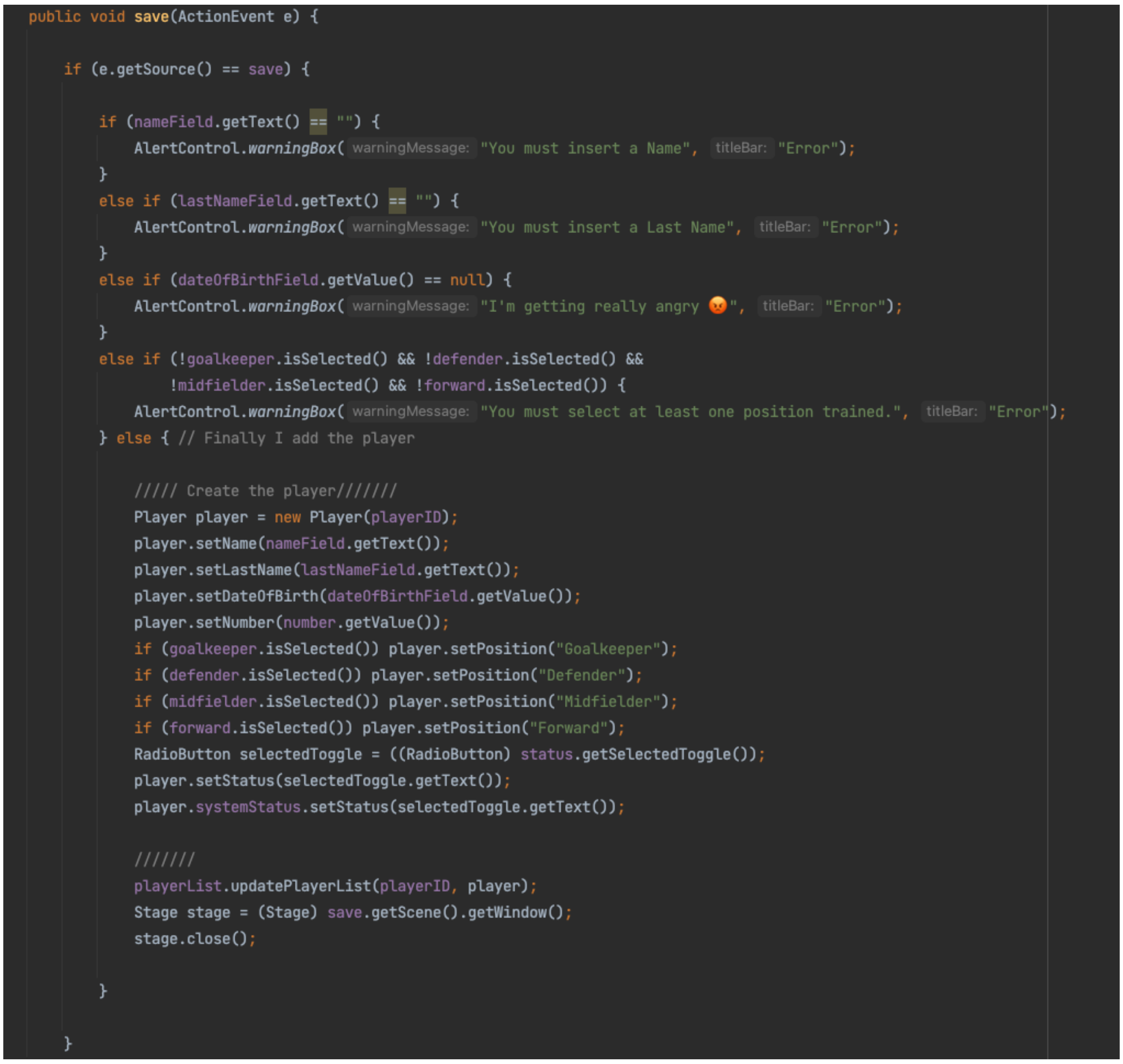
Inside the PlayerController, the controller receives the data through transferData, and in case that is an "Edit Player" option, the function fills the fields with the player information. In case is a new player, the only data to create is a playerUID.



```
public void transferData(Player player, PlayerList playerList, String action) {
    this.playerList = playerList;
    HashSet<Integer> availableNumbers = playerList.getAvailableNumbers(); // Create an array of available numbers
    number.getItems().addAll(availableNumbers);

    if (action.equals("add")) {
        number.getSelectionModel().selectFirst();
        playerID = playerList.getSize();

    } else if (action.equals("edit") && player != null) {
            nameField.setText(player.getName());
            lastNameField.setText(player.getLastName());
            dateOfBirthField.setValue(player.getDateOfBirth());
            number.setValue(player.getNumber());
            int actualAge = Period.between(dateOfBirthField.getValue(), LocalDate.now()).getYears();
            age.setText(actualAge +" y/o");
            playerID = playerList.getPlayerID(player);

                if (player.getPosition().contains("Goalkeeper")) goalkeeper.setSelected(true);
                if (player.getPosition().contains("Defender")) defender.setSelected(true);
                if (player.getPosition().contains("Midfielder")) midfielder.setSelected(true);
                if (player.getPosition().contains("Forward")) forward.setSelected(true);

            switch (player.getStatus()) {
                case "Available" : {
                    status.selectToggle(available);
                    break;
                }
                case "Unavailable" : {
                    status.selectToggle(unavailable);
                    break;
                }
                case "Suspended" : {
                    status.selectToggle(suspended);
                    break;
                }
                case "Injured" : {
                    status.selectToggle(injured);
                    break;
                }
            }

        }
    uid.setText("#ID:" + playerID);
}
```

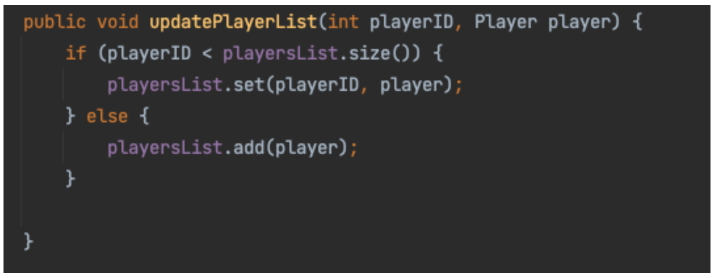
Figure 20 - Data transferred from the main stage

Finally, the action is taken once the user press Save. After this action, regardless of if it is an "Add" action or an "Edit" action, the system will create a new player object with the information in the fields. And then will call to the method updatePlayerList. If the player already exists, it will be a replacement; otherwise, it will create.



```java
public void save(ActionEvent e) {

    if (e.getSource() == save) {

        if (nameField.getText() == "") {
            AlertControl.warningBox( warningMessage: "You must insert a Name",  titleBar: "Error");
        }
        else if (lastNameField.getText() == "") {
            AlertControl.warningBox( warningMessage: "You must insert a Last Name",  titleBar: "Error");
        }
        else if (dateOfBirthField.getValue() == null) {
            AlertControl.warningBox( warningMessage: "I'm getting really angry 😡",  titleBar: "Error");
        }
        else if (!goalkeeper.isSelected() && !defender.isSelected() &&
                !midfielder.isSelected() && !forward.isSelected()) {
            AlertControl.warningBox( warningMessage: "You must select at least one position trained.",  titleBar: "Error");
        } else { // Finally I add the player

            ///// Create the player///////
            Player player = new Player(playerID);
            player.setName(nameField.getText());
            player.setLastName(lastNameField.getText());
            player.setDateOfBirth(dateOfBirthField.getValue());
            player.setNumber(number.getValue());
            if (goalkeeper.isSelected()) player.setPosition("Goalkeeper");
            if (defender.isSelected()) player.setPosition("Defender");
            if (midfielder.isSelected()) player.setPosition("Midfielder");
            if (forward.isSelected()) player.setPosition("Forward");
            RadioButton selectedToggle = ((RadioButton) status.getSelectedToggle());
            player.setStatus(selectedToggle.getText());
            player.systemStatus.setStatus(selectedToggle.getText());

            ///////
            playerList.updatePlayerList(playerID, player);
            Stage stage = (Stage) save.getScene().getWindow();
            stage.close();

        }

    }
```

Figure 21 - Save action, PlayerController

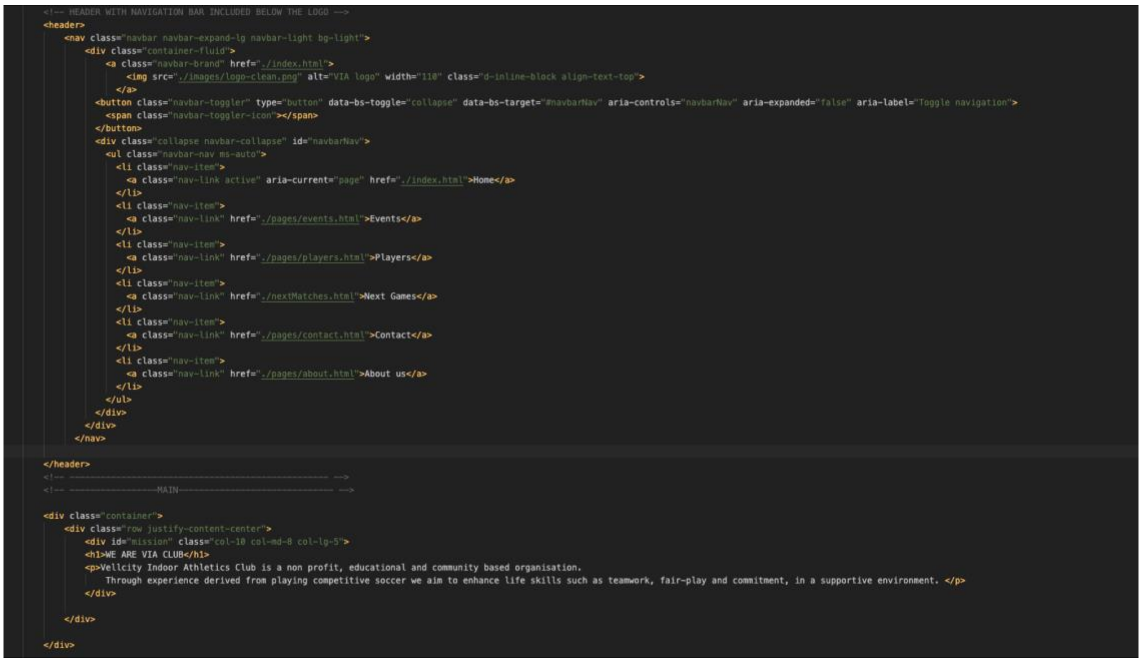


```java
public void updatePlayerList(int playerID, Player player) {
    if (playerID < playersList.size()) {
        playersList.set(playerID, player);
    } else {
        playersList.add(player);
    }

}
```

Figure 22 - updatePlayerList a method in PlayerList

4.2 Web

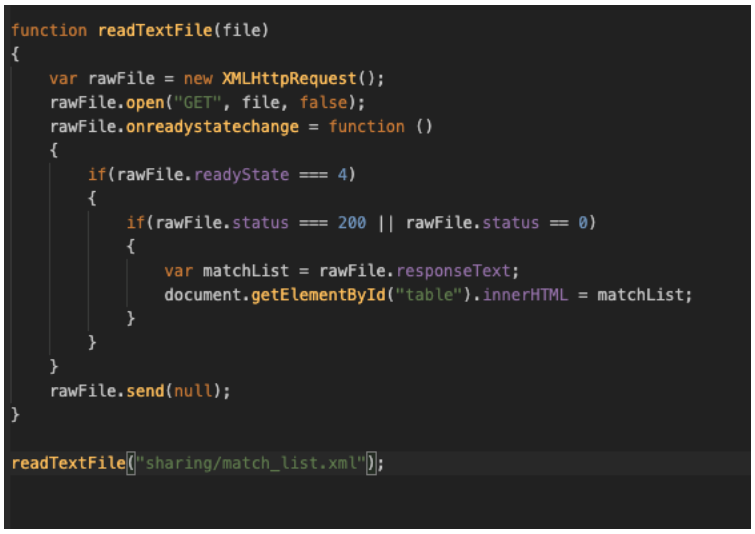
The website has a simple design. The navbar made with bootstrap



```html
<!-- HEADER WITH NAVIGATION BAR INCLUDED BELOW THE LOGO -->
<header>
    <nav class="navbar navbar-expand-lg navbar-light bg-light">
        <div class="container-fluid">
            <a class="navbar-brand" href="./index.html">
                <img src="./images/logo-clean.png" alt="VIA logo" width="110" class="d-inline-block align-text-top">
              </a>
          <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
          </button>
          <div class="collapse navbar-collapse" id="navbarNav">
            <ul class="navbar-nav ms-auto">
              <li class="nav-item">
                <a class="nav-link active" aria-current="page" href="./index.html">Home</a>
              </li>
              <li class="nav-item">
                <a class="nav-link" href="./pages/events.html">Events</a>
              </li>
              <li class="nav-item">
                <a class="nav-link" href="./pages/players.html">Players</a>
              </li>
              <li class="nav-item">
                <a class="nav-link" href="./nextMatches.html">Next Games</a>
              </li>
              <li class="nav-item">
                <a class="nav-link" href="./pages/contact.html">Contact</a>
              </li>
              <li class="nav-item">
                <a class="nav-link" href="./pages/about.html">About us</a>
              </li>
            </ul>
          </div>
        </div>
      </nav>

</header>
<!-- ---------------------------------------------------- -->
<!-- ----------------MAIN-------------------------------- -->

<div class="container">
    <div class="row justify-content-center">
        <div id="mission" class="col-10 col-md-8 col-lg-5">
        <h1>WE ARE VIA CLUB</h1>
        <p>Vellcity Indoor Athletics Club is a non profit, educational and community based organisation.
            Through experience derived from playing competitive soccer we aim to enhance life skills such as teamwork, fair-play and commitment, in a supportive environment. </p>
        </div>

    </div>

</div>
```

Figure 23 - HTML code of Index

The script where the file "match-list" is called and read the whole match list.



```javascript
function readTextFile(file)
{
    var rawFile = new XMLHttpRequest();
    rawFile.open("GET", file, false);
    rawFile.onreadystatechange = function ()
    {
        if(rawFile.readyState === 4)
        {
            if(rawFile.status === 200 || rawFile.status == 0)
            {
                var matchList = rawFile.responseText;
                document.getElementById("table").innerHTML = matchList;
            }
        }
    }
    rawFile.send(null);
}

readTextFile("sharing/match_list.xml");
```

Figure 24 - script to import the match table

5 Test

After the implementation of the GUI and the website, the test part was performed to check the correct performance of all the use case descriptions.

The system was tested using the information related to the use case descriptions. Following is the table where the outcome of the testing is showed.

Use Case	*Test result*	*Comments*
Create a Match	Works	The match could be created with de information required.
Manage Match Data	Works	The system can edit or delete any match information and add or remove players from the list.
Manage Players Data	Works	The system can register a new player with all the information required. It can be edited or removed
Share Data with Website	Works	The system creates a file with a list of all the matches created.

6 Results

The following table shows the previous requirement.

Priority	#	Requirement	Working
Critical Priority	1	As a manager, I want to visualize a list of all the players in the system, in order to know my team.	YES
	2	As a manager, I want to visualize a list of all the matches, the previous and the upcoming.	YES
	3	As a manager, I want to visualize the player list of players for a specific match, in order to make the players know.	YES
	4	As a manager, I want to create a new match giving the opponent, date, place, and type (Cup, League, Friendly).	YES
	5	As a manager, I want to add players into the match, in order to have my list updated.	YES
High Priority	6	As manager, I want to select 11 "available" players to go to the pitch, and 5 "available" players to go to the bench if the match is "League."	YES
	7	As a manager, I want to select 11 "available" players to go to the pitch, and 6 "available" players to go the bench if the match is "Cup".	YES
	8	As a manager, I want to select 11 "available" or "suspended" players to go to the pitch, and unlimited "available" or "suspended" players to go the bench if the match is "Friendly".	YES
	9	As a manager, I want to register a new player with a full name, unique number, a list of positions that have been trained, and the Status (available, suspended, injured, unavailable).	YES
	10	As a manager, I want to modify the Status of a player (available, suspended, injured, unavailable).	YES

	11	As a manager, I want to modify the fields on a match (date, place, and kind of match). The type of match can be changed only if there are no conflicts with the number of players and Status.	YES
	12	As a manager, I want to delete a match in order to have the list updated	YES
	13	As a manager, I want to modify the players' list in a specific match, in order to have the list updated.	YES
	14	As a manager, I want to export the data to a website in order, to show to the public the updated data.	YES
Low Priority	15	As a manager, I want to modify any player record as long as the number is not already used by another player.	YES
	16	As a manager, I want to delete a player from the list, in order to have the list updated.	YES
	17	As a manager, I want to see how long the player has been playing without rest, or how long has been resting without play.	YES

7 Discussion

Even though the system is working well, some questions would be nice to be discussed with the primary user to improve specific details. Here are some important examples.

-For how long is suspended a player? What is mean to be suspended?

The current system uses a "state" to mark the user as "Suspended" or "Available." There is a lack of information to know if the suspension is relatives to matches (amount of matches to play) or relative to time. The final decision was to not include any aspect related to this state to not decrease the functionality and readability of the software.

-If a player is changed, the historic matches are modified,

If a player played a match with a particular number, and time later he changes the number, the number was also changed once the user visualizes the past match. Because of that reason, there is a chance to export the list to text to keep the formats.

-Related to change the kind of match,

If the system has a "Friendly" match, with eight players on the bench, and the user change that match to "Cup," the system will delete the extra players to fit the new requirement. Even though the system alerts this situation, the extraction of the players occurs randomly.

-"X" is not the same as quit.

If the user closes the program using "quit" the system (if there were any changes to save), it will ask before closing if the user wants to save the changes. This option does not appear if the user closes the program using the "x" in the corner of the window. This could be fixed but requires a different implementation that was not considered.

8 Conclusion

The main goal of this project was to create a tool to help the manager of a team be more efficient in the ways that he used to manage the information related to his team. Also, to develop a new style webpage to share some data easily.

The analysis starts understanding all the requirements, and the following chapters had this consideration. The implementation and design of the project went further to the expectation creating a friendly user interface and intelligently showing information.

Based on the testing part, the conclusion is that every requirement has been fulfilled. The result is an intuitive, friendly tool with a simple interface and capabilities beyond the requirement.

9 Sources of information

Chron, 2018. Information Technology & Its Uses in Business Management. Available at: <https://smallbusiness.chron.com/information-technology-its-uses- business-management-51648.html> [Accessed March 7th, 2021].

The Garage HP,2019. Aging in the digital age: How technology is changing the way we grow old. Available at <https://garage.hp.com/us/en/modern- life/technology-aging-seniors-security.html> [Accesses March 7th, 2021].

DBU, 2021. [online] DBU Klubservice. Available at: <http://klubservice.dbu.dk/kampklar/vejledning/til-traenere-og-holdledere> [Accessed 9 Mar. 2021].

Tony Gaddis, 2015, *Starting out with Java: Early Objects*, Fifth edition, Harlow, England, Published by Pearson Education 2015.

Jon Duckett, 2014, *Javascript and jQuery: interactive front-end web development*, Indianapolis, Indiana, Published by John Wiley and Sons 2014.

Jon Duckett, 2011, *HTML and CSS: design and build websites*, Indianapolis, Indiana, Published by John Wiley and Sons 2011.

Benjamin LaGrone, 2013, *HTML5 and CSS3: Responsive Web Design Cookbook*, Birmingham, Published by Packt Publishing 2013.

Appendices

- Appendix A1: Personal Behavioural Style (E-Stimate).
- Appendix A: Project Description.
- Appendix B: Use Case Description.
- Appendix C: Requirements and Related Use Cases.
- Appendix D: Activity Diagrams.
- Appendix E: Class Diagrams.
- Appendix F: Sequence Diagram.
- Appendix G: Astah Files.
- Appendix H: User Guide.
- Appendix I: Software Source Code.
- Appendix J: JavaDocs.
- Appendix K: Website Source Code.

Process report

Alfonso Pedro Ridao 308833

Allan Henriksen

Mona Andersen

Software Engineering

First Semester

1st of June 2021

Table of content

1 Introduction 1
2 Group Description 1
3 Project Initiation 4
4 Project Description 4
5 Project Execution 5
6 Personal Reflections 5
7 Supervision 6
8 Conclusions 6
Appendices 1

1 Introduction

This report could be one of the most challenging parts of the project. The whole project was done just for myself. And even though I have many fights with me due to different aspects of my life, I could keep myself motivated and focus during every phase of the project.

I felt prepared for this project. I assisted to every class, in every course during the semester, and focused on subjects where I felt weak.
I haven't had many supervision meetings, and I made almost all the decisions for myself. That is not something to be proud of, but I tried to avoid the "online" meeting as much as I can.

May the reporting guideline is not prepared to make an efficient one-person group report, but I will do my best.

2 Group Description

What makes us citizens of a particular country? Our place of birth? The passport we handle? The place we were raised? The family who raised us?
Well, that depends exclusively on who is evaluating that and which considerations it will be taken.

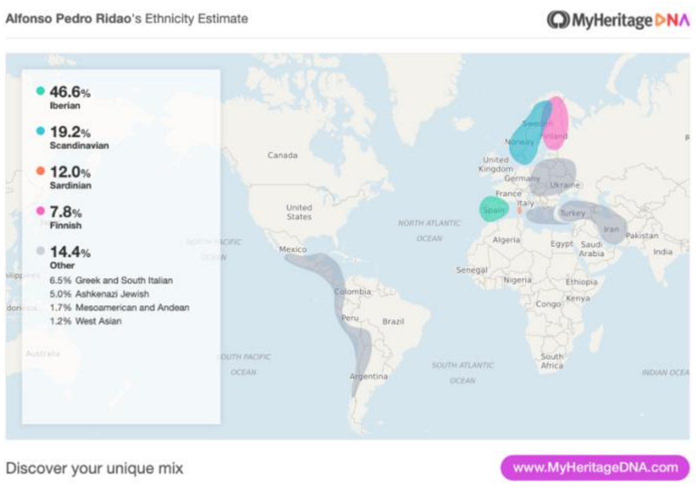


Figure 1 - Alfonso RIDAO - Ethnicity Estimate

In my case, I was born in Argentina, but with European roots. With 46% of Iberian blood and almost 20% of Scandinavian genes (My Heritage, 2021), among others *(Fig 1)*. I have two Nationalities (Italian and Argentinian) and speak three languages, and I make a big effort to have the minimum of civil knowledge to be considered "Earth Citizens." (Timi Ećimović, 2016)

As planet Earth is not a Hofstede method option, the graphic shows the three countries more representatives in my life *(Fig 2).*

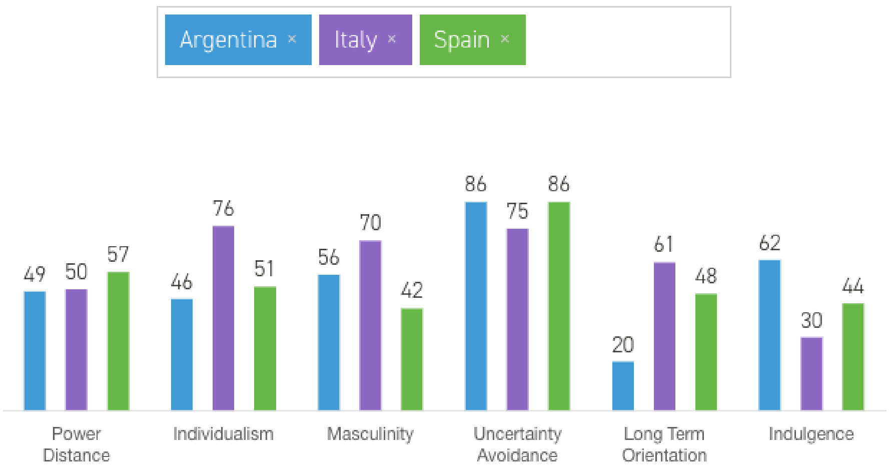


Figure 2 - Hofstede Insights, 2021

As we can see in figure 2, most of the three countries have similar values in power distance. This cultural aspect is perceived in group work as members do not have equality, and the power of the group is distributed in a hierarchical way where some lead and others follow (Hofstede Insights, 2020).
Also, we can see that Italians show a little bit more individualism and masculinity. This refers to whether a person focuses on individual goals or aims of the group (Hofstede Insights, 2020). Cleary, this is my Italian part developed.

Looking at the e-stimate profile, we can understand in deep the behavior of this "One-Person group."

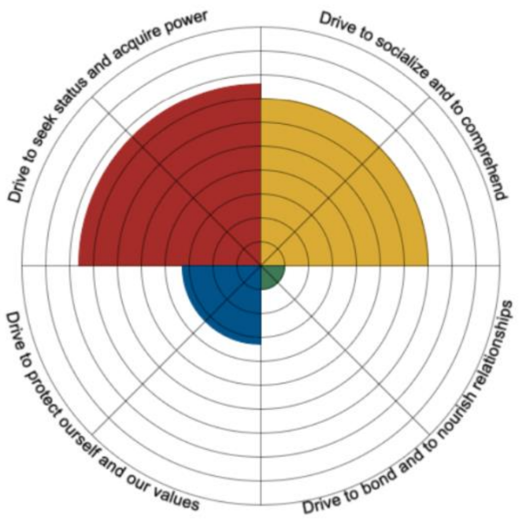


Figure 3 - e-stimate profile - Alfonso Ridao

The red behavioral tendency is a reflex that likes to control their surroundings – (want to achieve goals and obtain results) (e-disc, 2021). This could be a reason why there are no others in the group.
Also, we can see a great part of yellow behavioral tendency, which means that it likes to be at the center of attention. People with yellow behavioral tendencies are innovative, outgoing, and convincing. (e-disc, 2021). This could be reflected in the energy and hours of implementing the project, even though we know that the software part is just 25% of the final project.

Unfortunately, the lack of Green Tendency - Pursue cooperation - focus on people, are sociable, and prefer harmony. (e-disc,2021) - is a reflection and consideration to be aware of in the future and a big step to start to work on it.

Lastly, a low level of behavioral tendencies, seek known surroundings - they are careful and focus on precision and detail. - It could have a significant impact on the final resolution of the project to reach all the duties and deadlines.

3 Project Initiation

A great group was formed. And I was proudly part of group number 3. I met the people, and they were friendly people to work with, probably.
We present the case together, and we made the project description and a video related to the project description.
Unfortunately, I could not meet them personally. We had the whole semester online, and I struggled to work with some people that I don't know, in something that we never did, without meeting each other. I took the risk to do this by myself, and every day since then, I wonder if I made the right decision. Even though I start to think that passing SEP1 is possible, I am still wondering if it could be a better way to keep working in the group.

4 Project Description

Realizing the Project Description had a significant impact on the group relationship. It was our first assignment together, and I think it was not made efficiently. It is understandable; it was just the first phase, of the first project, of the first semester, of a group that did not know each other.
Finally, I think we achieve a great result. We could evaluate all the problems presented, and it was used and consulted during the whole part of the project development.

5 Project Execution

After the Easter break, I was ready to start with the execution phase on my own. Beginning with the analysis, I must admit that after thinking about the project for a long time, I felt this part relatively easy. I could finally be selfish, making decisions for my own.

The design part was challenging, but I found it fun. Even though I had no previous experience designing any system, I did not find significant issues to develop it. Some of the Diagram classes were eliminated or modified in the whole process, but the primary idea was kept as the initial idea.
The implementation took about three days, without including the GUI or graphics, totally it was around six days of full-time coding. The main point I found with the graphics interface. I never used one before, and it took me time to learn how to use it properly.

I must recognize that I had to come back to the design and modify some aspects in this step. I experienced some of the shortcomings of the waterfall method here.
After that, an extended test was running, and many bugs were found, returning to the implementation or even design and modifying some aspects.

6 Personal Reflections

Reaching this step is getting more challenging to follow this template. The whole process report includes my reflections, but if I have to do a reflective writing, this could be related to the fact that I decided to do this alone.

I was struggling to do the project effectively when I was in my group. I was not being the best version of myself, and the conflicts were increasing.
According to Friedrich Glasl's Conflict Escalation Theory, every group can either escalate or de-escalate a conflict (Friedrich Glasl, 2021). I think I did not give time to de-escalate.

Stress levels were increasing, and I did not could deal with my partners. I am not happy with the resolution. I know the situation with the pandemic is not often, so I am looking forward to working in a team during the next semester.

7 Supervision

When I am writing this report, I have not had many supervision meetings with any of the supervisors. I did not want, not because I believe I can do this without supervision, but because I felt challenging to handle the online meetings and discuss certain aspects online. I am a visual person, and I need to have deep conversations in a meeting, so I tried to avoid unnecessary meetings. This behavior made me found solutions for myself. I am sure that you will find many errors through this project, which could be easily fixed with a few meetings, but I decided to keep the online meeting as minimum as possible.

8 Conclusions

This is a project to be made by different persons. Not just to reach the goal of approving the assignment. But to increase the knowledge in every aspect and be prepared to follow the guidelines in the following semesters.

Some of my recommendations could be:

- Find people in your group who are sharing your project's standard.
- Find someone who complements you and someone from whom you can learn.
- There are many ways to reach the same result. Be open.
- Understand the cultural difference.
- Keep yourself positive, and keep the wrong energy out of the group.

And again. I think I could not be the best person to give some advice or recommendation. But I learned many valuable things from this project, and I am looking forward to applying to the next one.

Thank you for reading until the end.

Appendices

- Timi Ećimović, 2016. The philosophy of true harmony in global citizenship. [online] Available at <https://www.un.org/en/chronicle/article/philosophy-true-harmony-global-citizenship>

- Hofstede, G., 2021. Country Comparison. [online] Hofstede Insights. Available at: <https://www.hofstede-insights.com/product/compare-countries/> [Accessed 25 May. 2020].

- e-disc 2021. E-stimate Persona profile. [pdf] Available at: <Appendix A1>

- My Heritage 2021. Personal profile [pdf] Available at <https://www.myheritage.com/>

- Friedrich Glasl, 2021. 9 Stages of Conflict Escalation according to Friedrich Glasl. Available at: < https://projectmanagement.guide/9-stages-of-conflict-escalation-according-to-friedrich-glasl/ >