La Salle Campus Barcelona

International Computer Engineering (ICE)

LEAGUE MANAGER

Object Oriented Programming and Design

GROUP 8:

Laia Casas Irure

Valèria Ezquerra Rodriguez

Eugènia Pacheco Ferrando

Sebastián Félix Gorga

Daniel Hess

28th May 2023

Index

1 PROJECT SPECIFICATION SUMMARY

We, Group #8, have decided to develop the 'League Manager' application.

Within this, we are asked for some minimum requirements, which are the following:

The project is based on the idea of an application to manage the information of certain sport leagues. This information can be managed either by the league administrator or players. Depending on the user accessing the platform, they will be presented with different options.

Any user accessing the program will have the option to log in with their credentials. To do so, they will be asked to enter their username and password. League administrator must log in using the username 'admin', while players must use their DNI/email. In addition, they have the possibility to recover the password.

When logging in as an administrator, you can choose between three main themes: teams, leagues or statistics. If the selected option is **teams**, a list of already existing teams will be displayed together with two buttons that allow you to create or delete a team from the database. If the create option is selected, the admin will have to upload a json file containing all the information he/she wants to load into the system. Internally, the system will be in charge of storing the information as well as creating the players that do not exist and showing the administrator a list with their IDs and passwords.

If the selected option is **leagues**, the operation will be identical to the teams. The only difference is that, instead of a json file, you will simply be asked to enter the name of the league (which cannot already exist), a start date and a start time (both of which must be valid). From here, you will be presented with a box to select the teams that will participate in the league you are creating.

To delete a team or a league, the procedure to follow will be as simple as selecting the team/league to delete from a list.

If you choose the **statistics** option, a graph will be displayed showing the number of rounds as well as the number of points accumulated by each team.

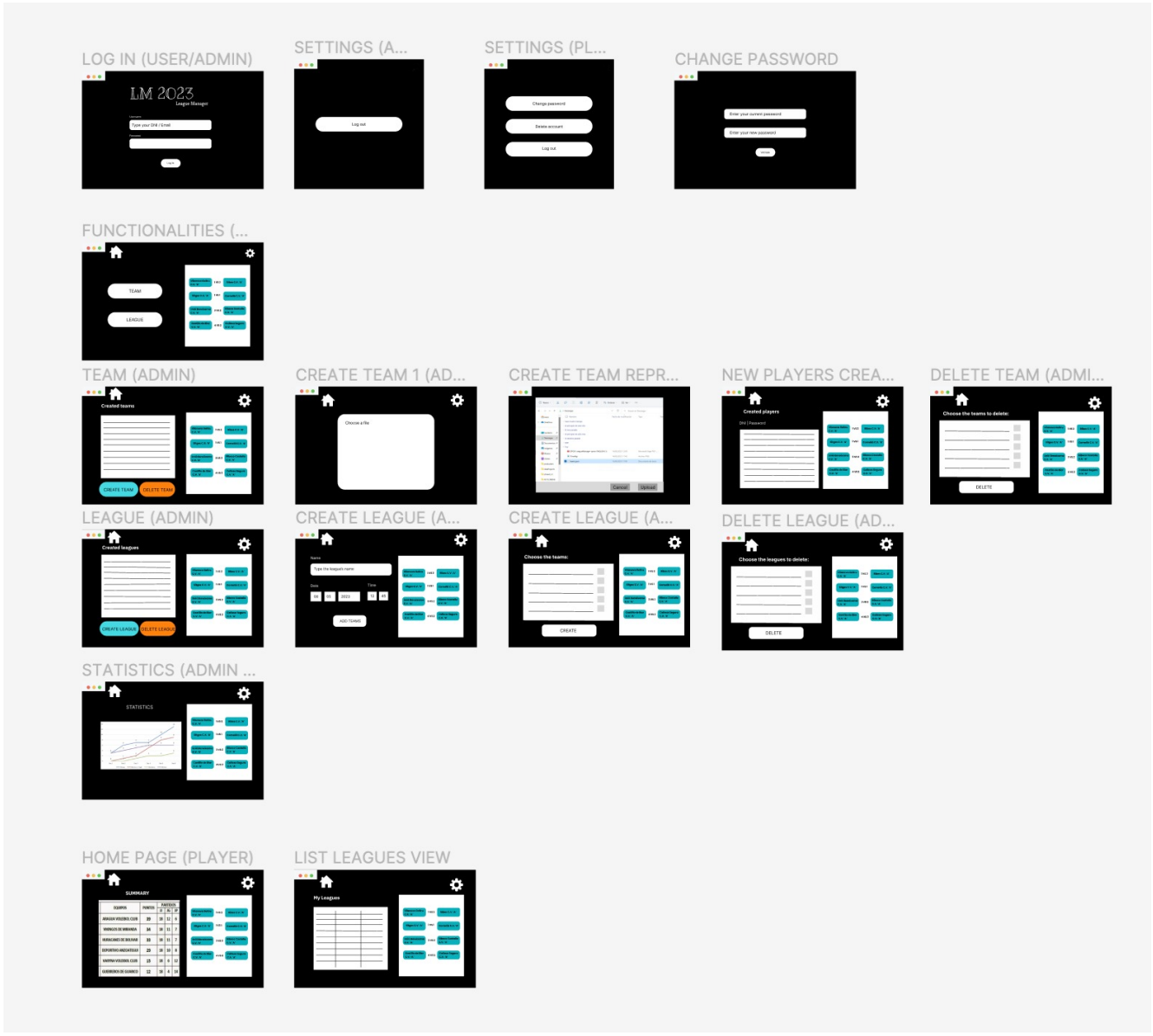
To access the program as a player, the admin must have already registered the player in the system and sent to him the access credentials in the appropriate format. In this way, when the player logs in, he will be able to see all the information regarding the leagues in which he participates. Additionally, players have the ability to change their password or delete their account.

It should be noted that any user can see in real time, as a live streaming, the tally of the matches being played. While the admin can see all the matches, the player will only be shown those in which he is participating. Both users can also log out whenever they want to exit the app.

2 GRAPHICAL USER INTERFACE DESIGN

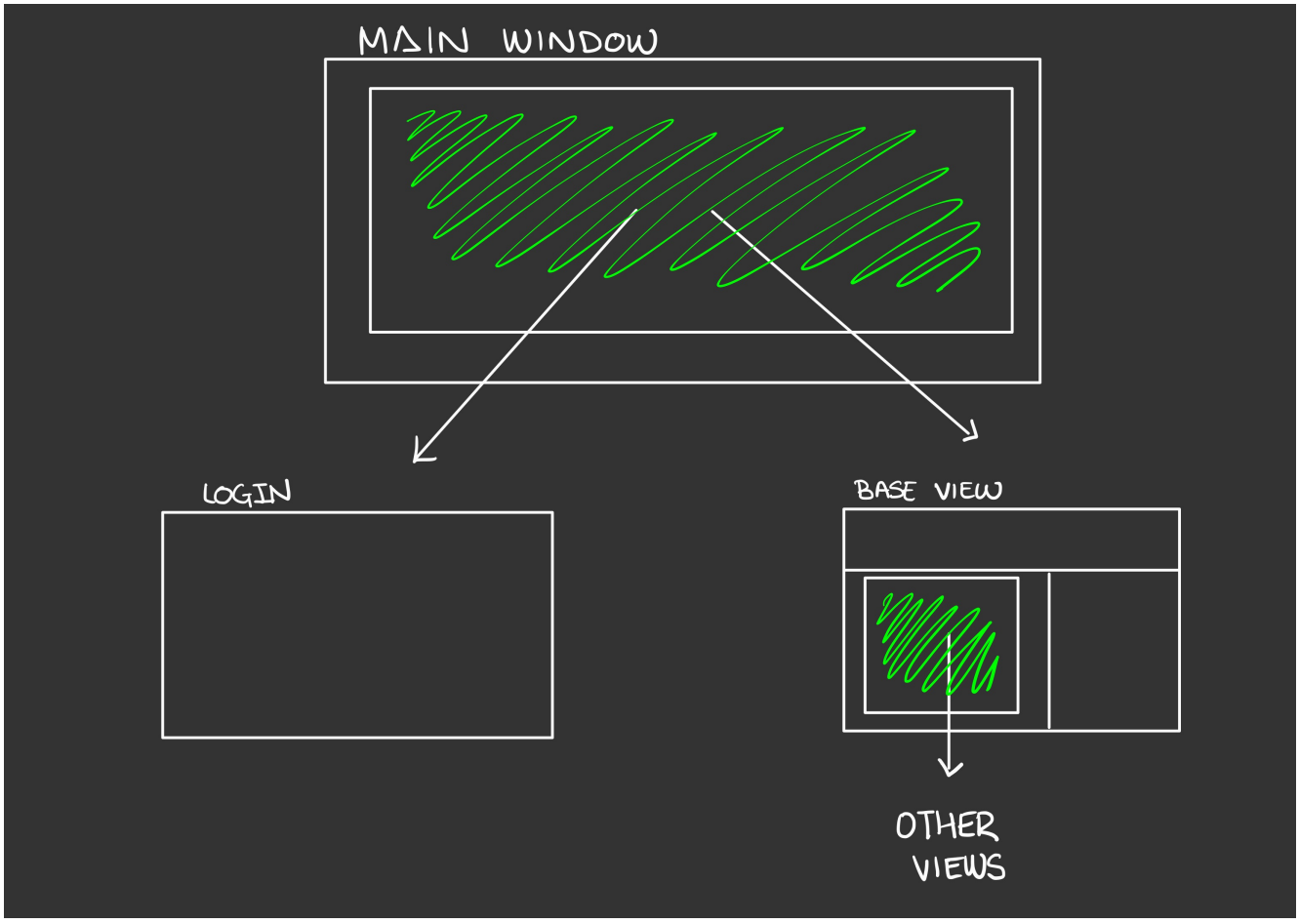
We are going to divide this explanation into different sections since we think that it is easier to explain view by view the different frames that we did for the app and what are the layouts and components needed.

The general mockup is the following:

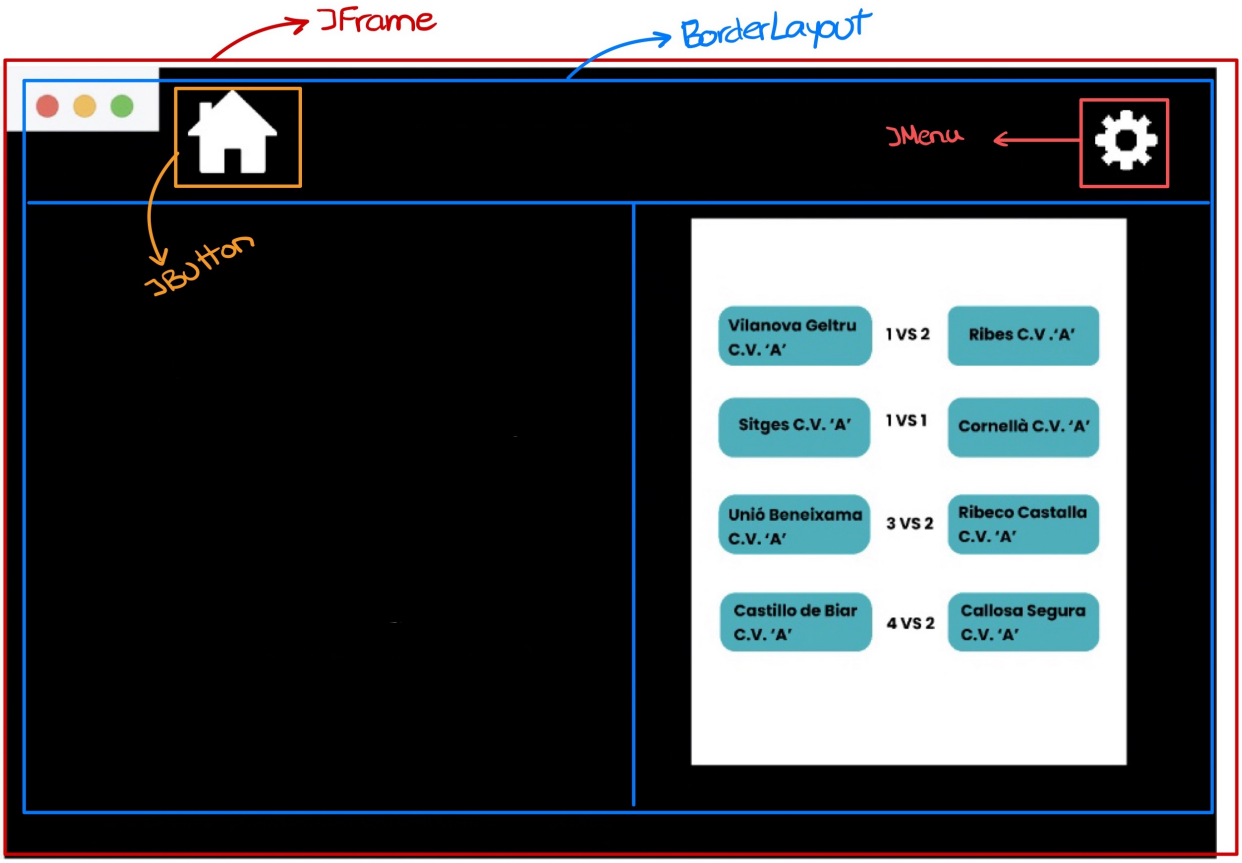


2.1 CardLayouts

First of all we should talk about how the views are structured. Once we made the figma we started to think about how to go from the login screen to the rest of the screens without having more than one **JFrame**. For it we decided to make use of the **CardLayout**. We applied 2 of these, the first of them in the MainWindow and in this there could be the logIn or the BaseView. The second CardLayout is located in the baseView since we realized that the rest of screens all had the same base and the only thing that changed was the part of the center with which it was here where we applied the second cardlayout. Here below we leave a schematic of what we have done:



2.2 Base View



The baseView was created to not have to repeat the same code in all the views since it was not the most effective way to do it because it would be to make copy paste and edit the center. This class makes an extend of a **JPanel** to be able to place it in the first cardLayout.

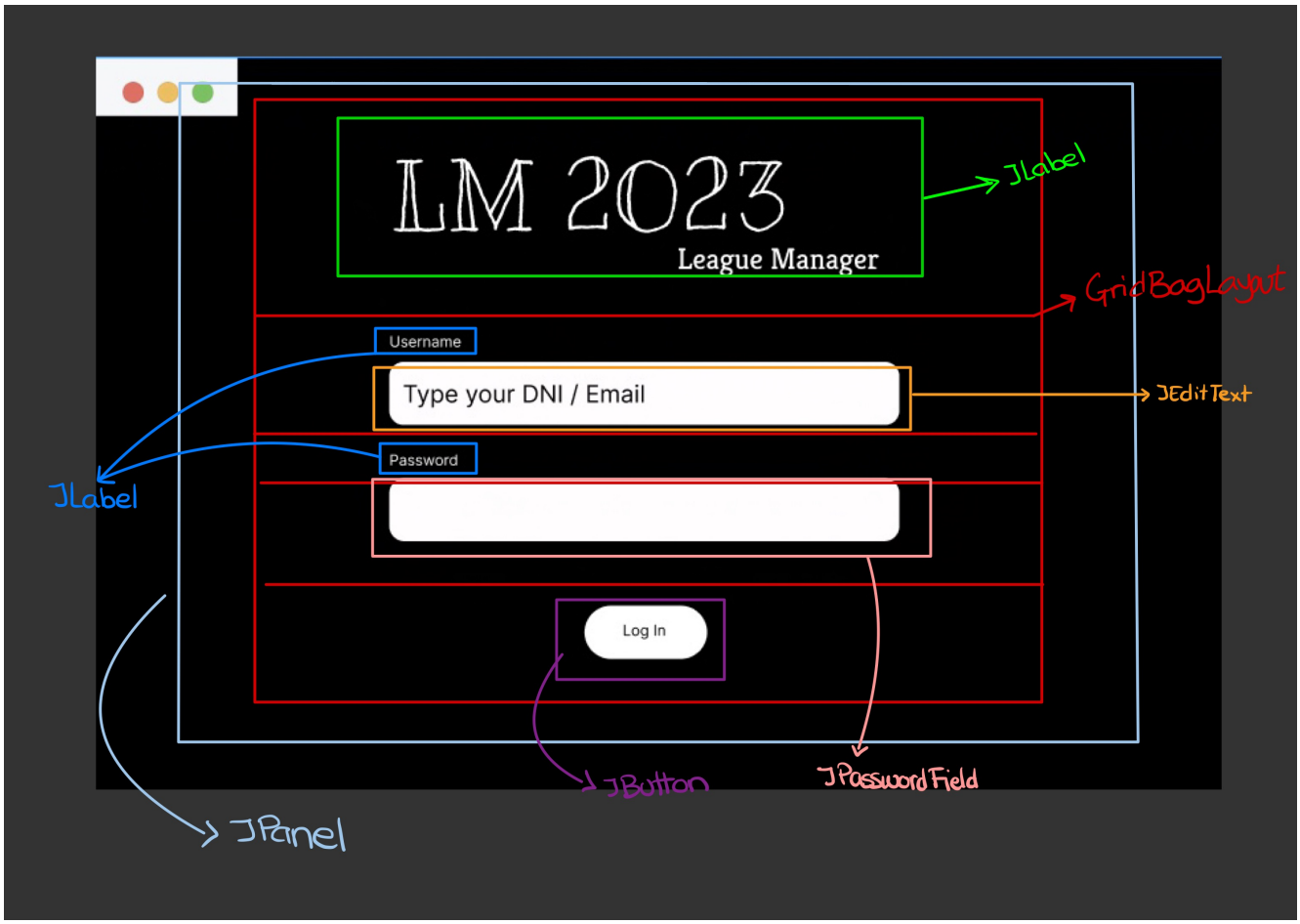
The first thing we do is to create a **BorderLayout** to divide the panel in 3 different sections:

The first one is the North part where the user is going to be able to do 2 things. First he will be able to click on the icon of the house that will be a **JButton** by means of which he will be able to return to the main page of the application. Secondly and lastly you will be able to access the settings that depending on the type of user logged in will be one or the other. The settings icon has a **JMenu** from which the different options will be displayed.

The second section is the Center that in this case as you can see there is nothing because it is the **JPanel** of the second **CardLayout**. It will be in here where all the other views are going to appear depending on what the user chooses.

The third and last section is the East part of the BorderLayout where the LiveStreaming will be displayed all the time, that is to say, the matches that are being played. This liveStreaming is going to be explained how it is done after with more detail.

2.3 Log In View



The Log In view is the first screen shown to the user. It prompts the user for its credentials, that is username and password. In our program, there's two possible cases, entering as an administrator or as a player. For the first case, the user will have to prompt "admin" and the password saved in the config.json file. However, for the second case, the player will be able to log in with their DNI or Email.

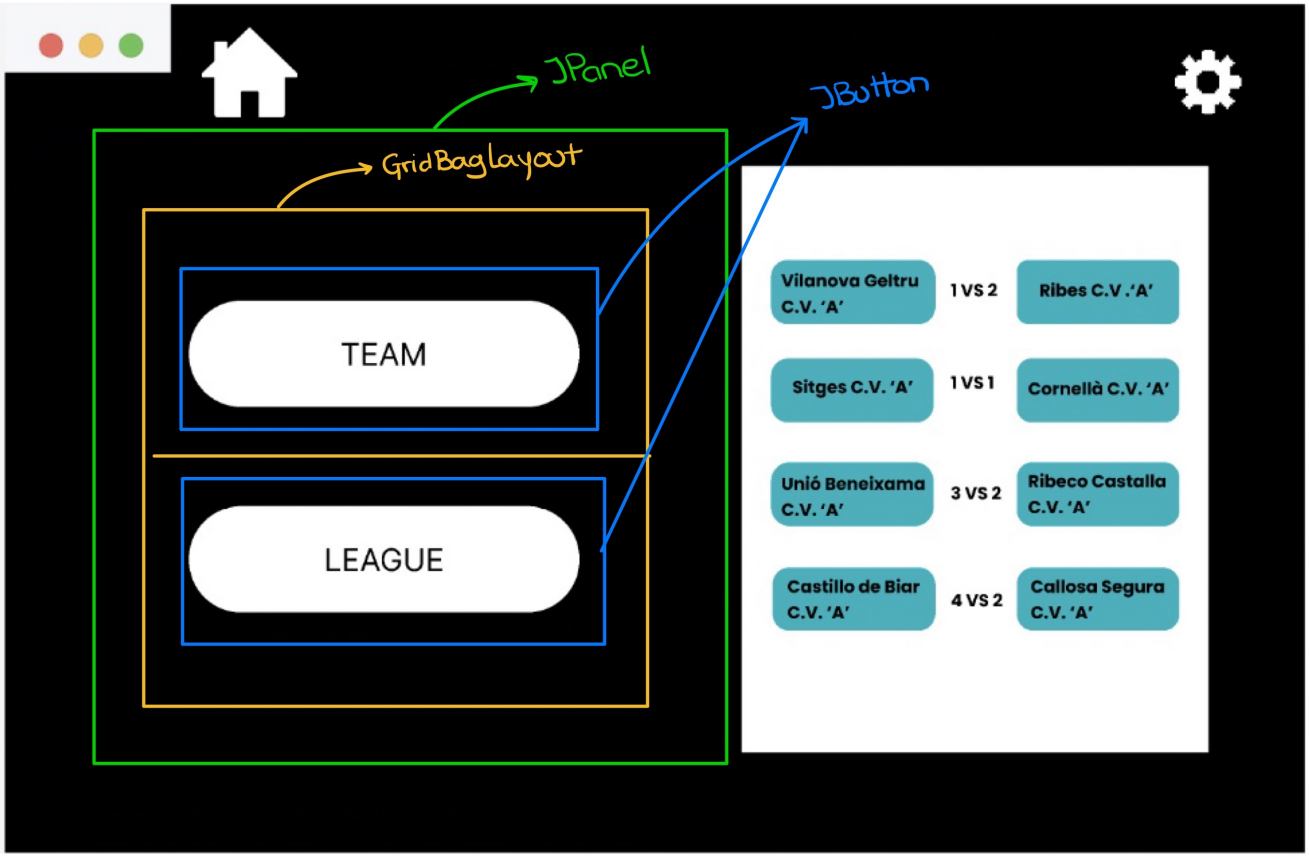
To be able to do that, we have designed a **JPanel** that contains a **Gridbag layout** with five main panels that have a **BoxLayout**, defined size and Black as background color.

The first one creates the header to show the main title of the application. To do so, we have two **JLabels**, "LM 2023" placed on the left and a subtitle "League Manager" just below on the right side.

Second panel shows the username settings with a **JLabel** and a **JTextField** to enter the corresponding information. Between those elements, we have placed two vertical spacers. Similarly, the next panel has a Password **JLabel** and a **JPasswordField** to ensure the security of our user.

Moving into the fourth panel, we find a **JLabel** that is first empty and is filled by calling setMessageLabelText() whenever the credentials entered are incorrect. Also, in this case we set a red border to both TextField and PasswordField. Finally, the fifth panel has a **JButton** centered to be able to validate credentials and continue with the program whenever the user presses it.

2.4 Admin Main View



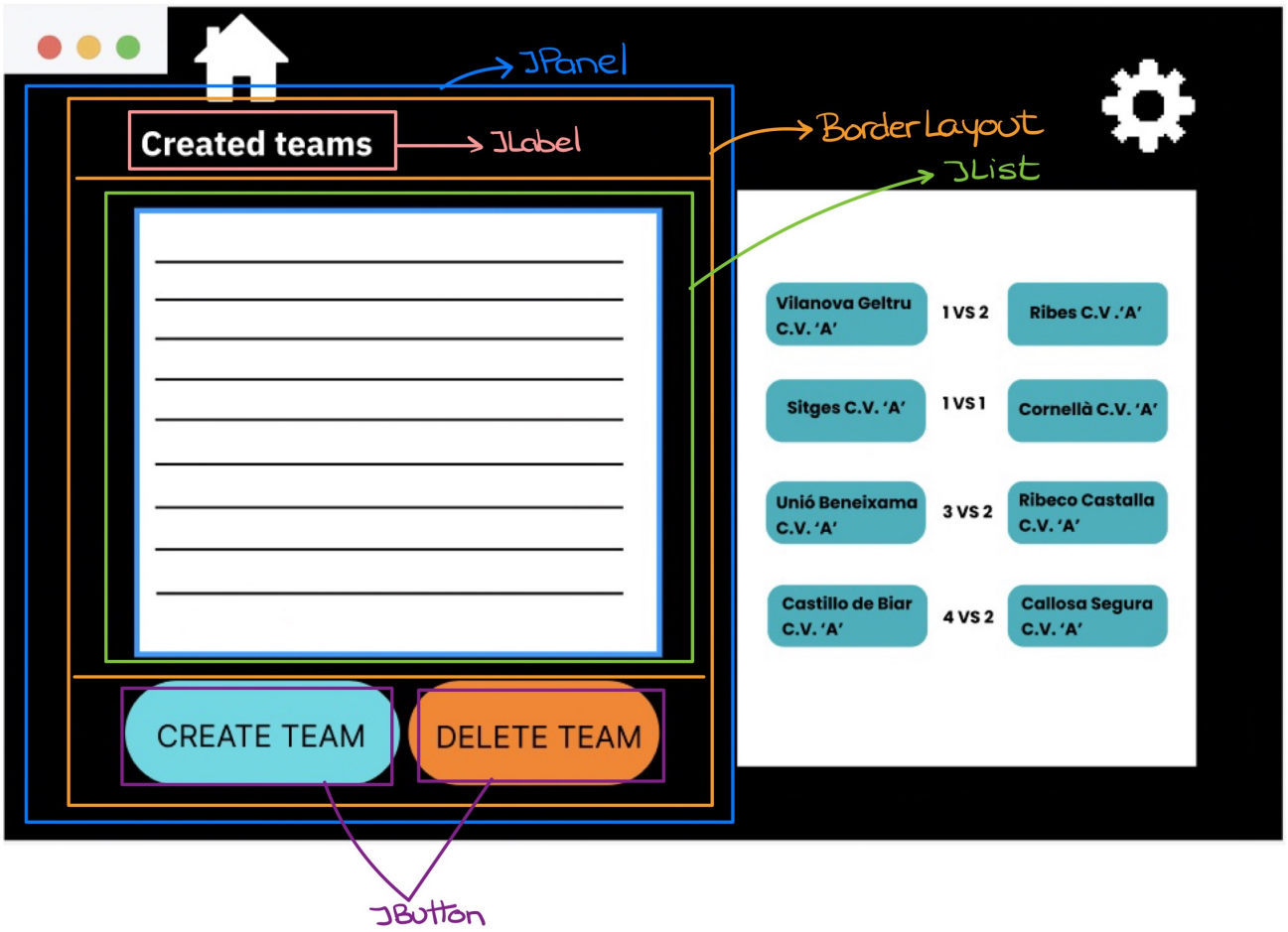
The Admin Main view is the first screen that the user will see once he logs in as an Admin. As in the previous views the base of this screen is the **JFrame** which basically comes from the CardLayout previously explained.

The first thing thing that we did was extending this class from a **JPanel** since we have to fit this screen into the baseView.

In this JPanel what we create is a **GridBagLayout** so we can place the two needed **JButtons** in their places and with the same distance between them and in the middle of the panel.

The first buttons corresponds to the next views that will be related to the Teams so once the user clicks on this button he will go to the next screen of the teams part. The second button as you can see it is related with the leagues and does the same as the previous button, it leads you to the next screen related to the leagues.

2.5 Teams or League View

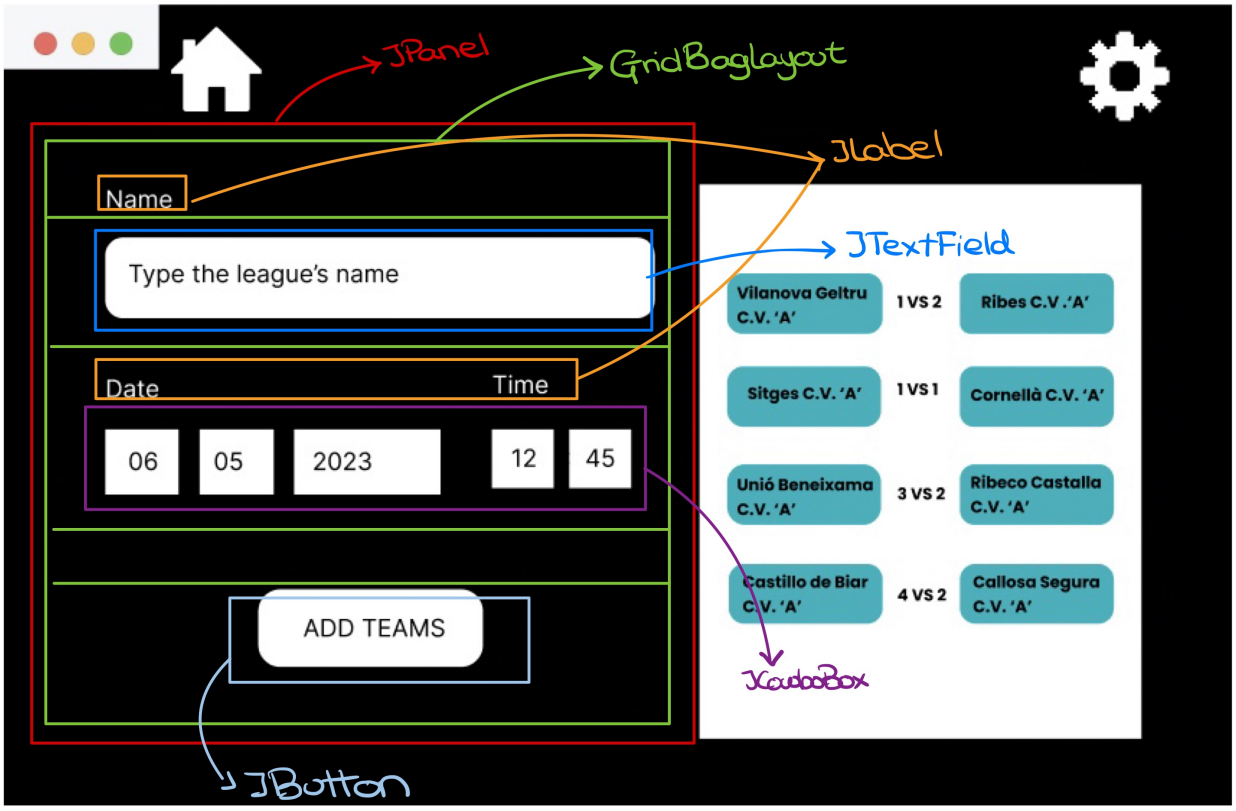


In this view the first thing we do is to extend the class of this one with a **JPanel** that will be the one that will go inside the baseView (as it is well seen in the image).

First we create a **BorderLayout** which we will divide in 3 sections. The first one is the north section where we will find a **JLabel** with the title "Created teams". Secondly, the center part inside which we will make a **JList** with all the names of the teams created so far. Finally, section 3, the south part where two **JButtons** are located, one to create a new Team and the other to delete one of the teams created.

There is another view that is exactly the same but changing the word Team for the word League but the rest is exactly the same so we don't think it is necessary to add an explanation on this one.

2.6 Create League View



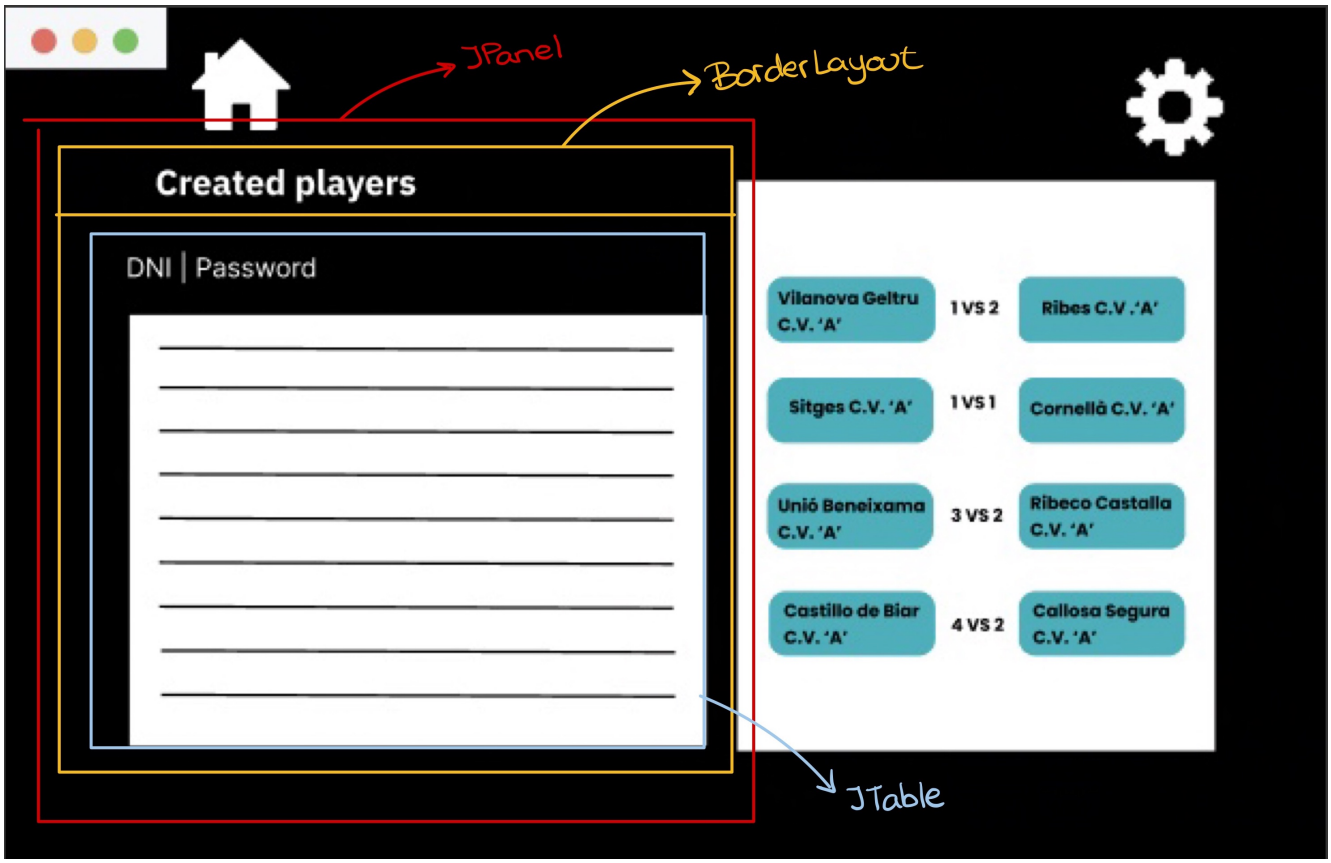
Once the user clicks on the create league button the application will take him to this screen where the user can create a new league.

Create League View is the window that asks the user the name, start date and time date of a league. To do so, we have done a **JPanel** that contains another panel with a **GridBagLayout**. This new panel contains respectively three main panels. The first one creates the name settings with a **JLabel** and **JTextField**. To place them, we use two vertical spacers between the components and use a **BoxLayout**.

The second panel uses a **GridLayout** with one row and two columns to show the date and time settings. Each one has a **JPanel**, the first one is located on the first column and the second one into the second column. The date settings contains one **JLabel** and another Panel for the spinner that will receive three **Combo Boxes**, they will correspond to days, months and year respectively. For the time panel, we have one **JLabel** and another **JPanel** that will have two **Combo Boxes**, one for hours and another for minutes. Finally, the last panel will contain the **JButton** that will go to the next view.

When the user clicks on the create team button a **JFileChooser** will appear so that the user can select a json file with the team and its team members and once he accepts on choosing that file the program will run to the next slide.

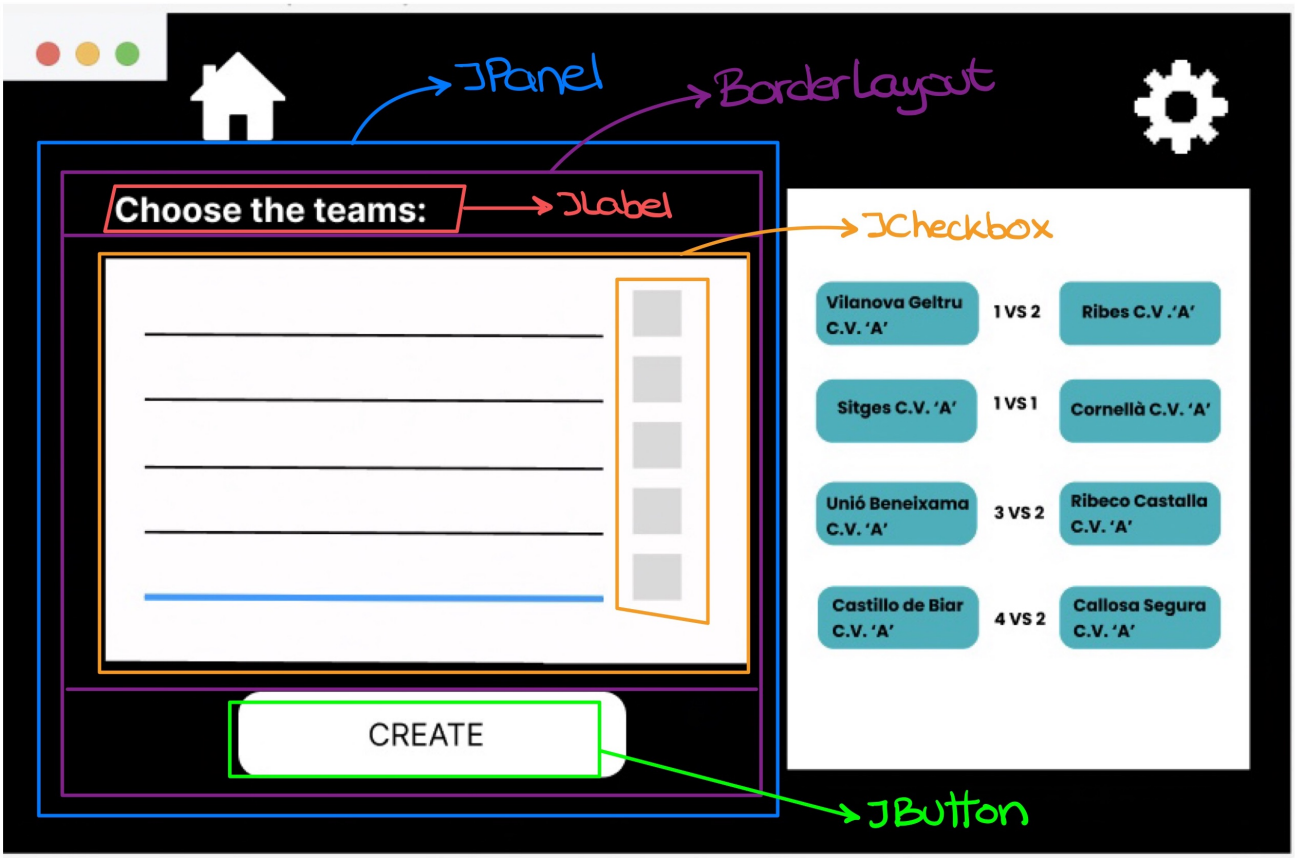
2.7 Created Players View



Created Players View is also a **JPanel** so that after we can add it to the cardLayout.

The first thing we do is to create a **GridBagLayout** or a **BorderLayout** to be able to place the different components such as the following. In the upper part we have a **JPanel** with a **JLabel** to be able to place the title "Created Players". Down this we have to place a **JTable** where the information of the players will appear.

2.8 Add Team to League View

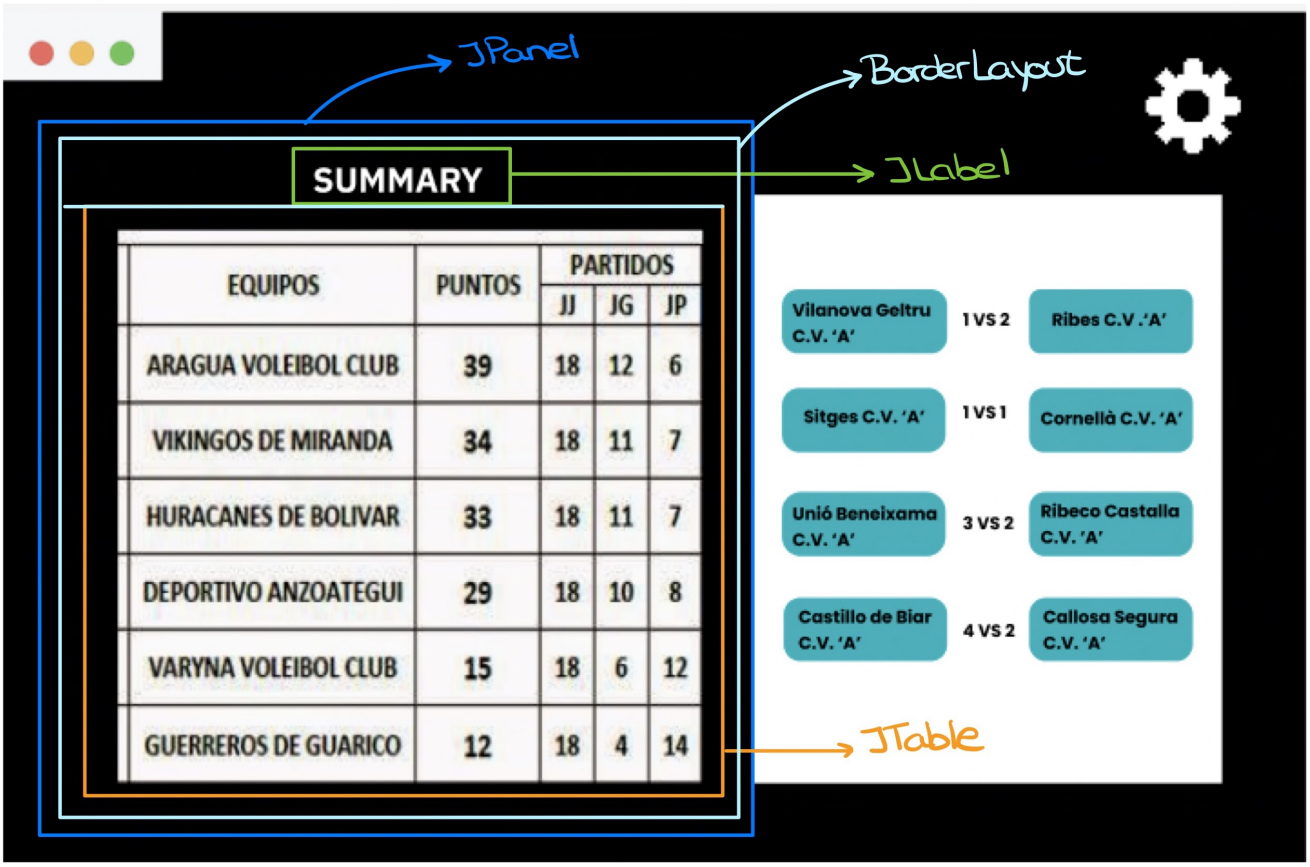


This is the view in which we will be able to add the teams to a league we are creating. To make this view, as in the previous ones we extend the class to a **JPanel**.

Inside this JPanel the first thing we do is a **BorderLayout** which we divide in the 3 typical sections: north, center and south. In the north part we place a **JLabel** that will put "Choose the teams" so that the user knows where he is and what he has to do more easily. In the center what can be seen is a set of **JCheckbox** because we want the user to be able to click on those team names that are in the list. Finally, in the southern part we will have a **JButton** that the user will have to click to finish the creation process. That is why the button has a text that says "create".

Similarly to that view, we have created another two. Delete League View lets the user delete the leagues he wants. The main difference is the **JLabel** to print the following message "Choose the leagues you want to delete: " and the **JButton** to "DELETE LEAGUE". The other view is Delete Team View that also changes the same components but this time to "Choose the teams you want to delete: " and "DELETE TEAM" for the button.

2.9 Player Main View



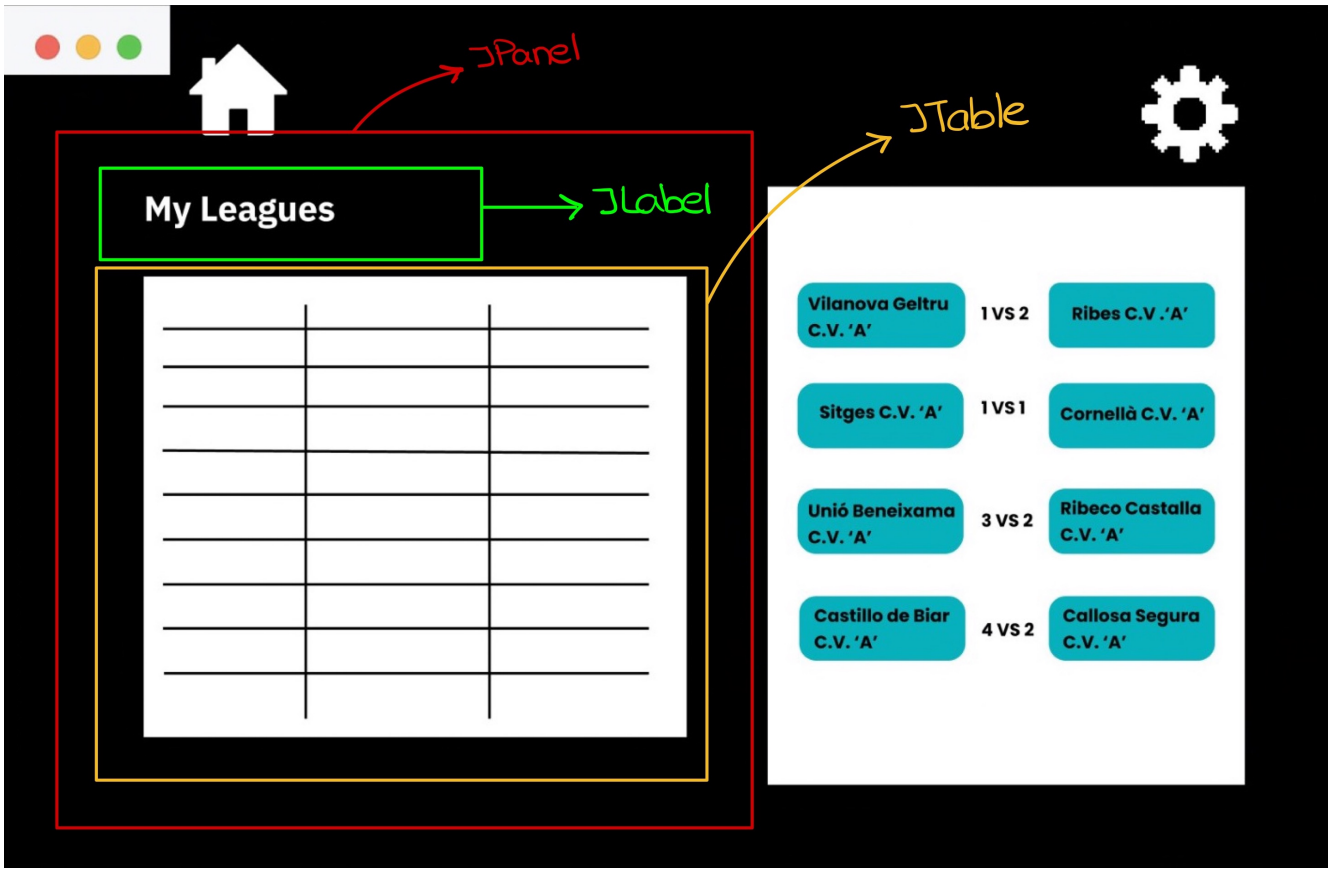
As in the previous screens we will also extend this one from a **JPanel**.

Again the first thing to do was a **BorderLayout** because it is the easiest and most effective way to divide the panel into different sections. In this case we will only have two: the north and the center.

In the north part there will only be a **JLabel** with some type of text, in this case we have put summary but in the program itself can appear any other type of text.

In the center part we find a **JTable** since by means of this we can make rows and columns for the different sections that we need since it is here where it will appear all type of information of the leagues in which the player is.

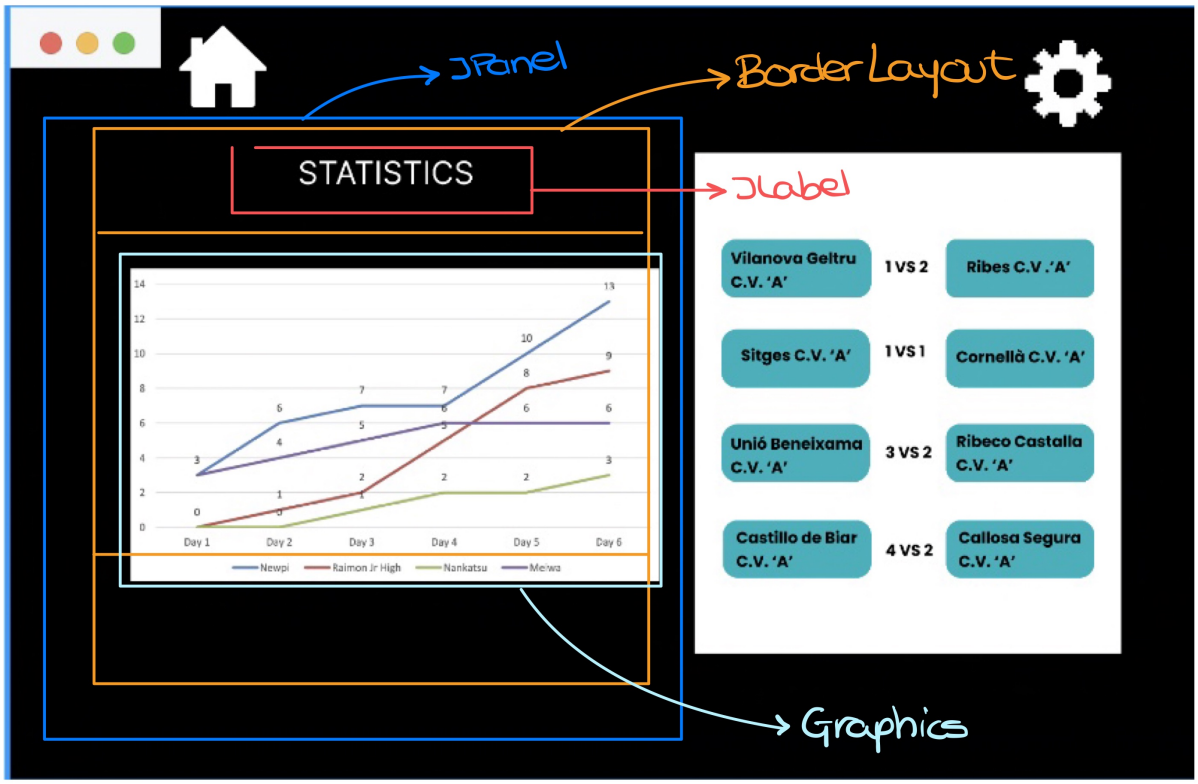
2.10 List Teams or Players View



This panel has the objective to show all the Players available inside a Team. To do so, we have placed a **JPanel** with a BoxLayout that places elements in y axis, everything inside the **JPanel** of the view. The **JTable** will contain a Data Model with the name of the player, dni, email, Squad number and phone. At the top, there will be a **JLabel** placed in the middle thanks to the method setHorizontalAlignment(SwingConstants.CENTER) and that will be set to the team name the user has entered. Below the table, we have placed a **JButton** to go back to the previous view.

List Teams View is quite similar to List Players View but changing the label and placing the team's name instead of the leagues one, and adding one button to be able to show statistics in real-time.

2.11 Statistics View



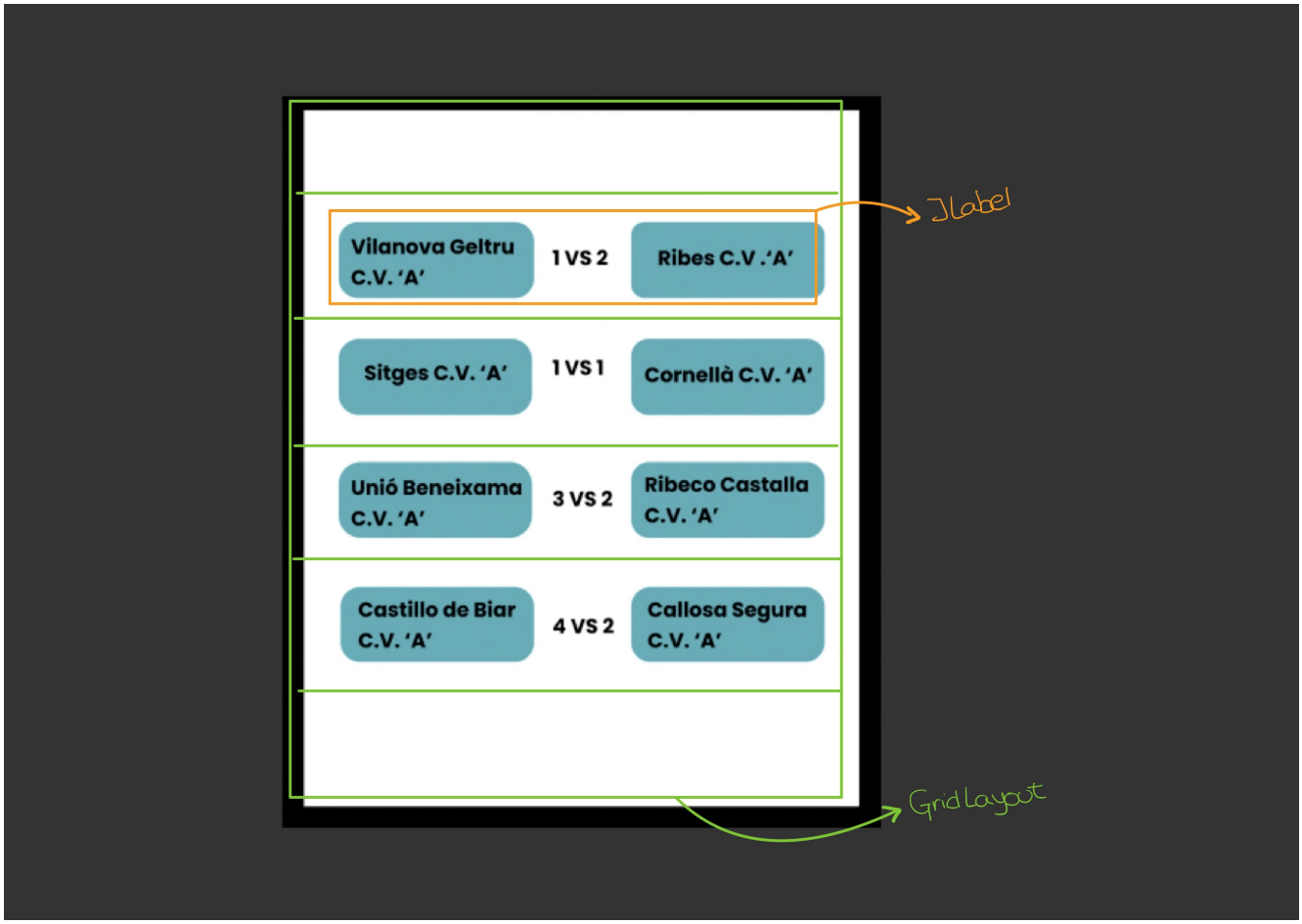
The StatisticsView class is a view that displays a graph representing statistics. It is designed to be used in a graphical user interface.

The StatisticsView class extends **JPanel**. It has a static public identifier to identify it, along with the width and height of the panel as well as the padding, x and y axis labels.

The class has several instance variables, including the chart data, colors used for the different data sets, team names and a maximum score. To fill the chart data, we use generateData method thanks to a list of provided statistics, generateColors() to initialize the array of colors used for the different data sets.

To be able to draw the graphic we use the library from SWING/AWT called Graphics, since it gives us the opportunity to use the paintComponent function.

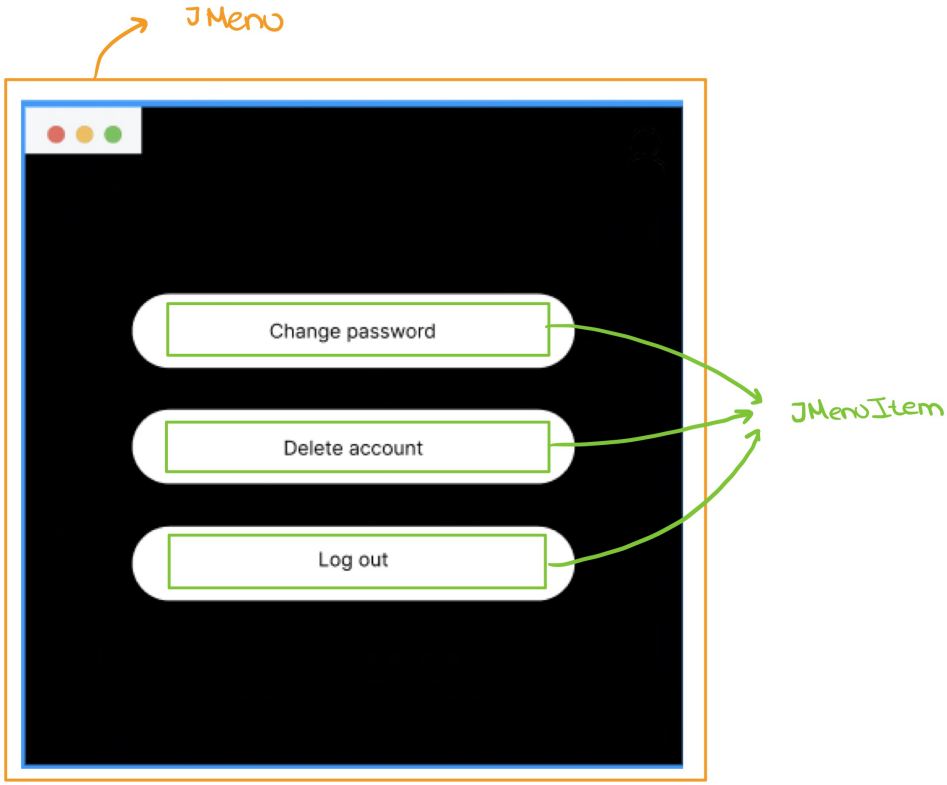
2.12 Live Game View



This is the section that will appear in all views of the application on the right side of the screen so that the user is constantly informed of how the matches are being played live.

To be able to do this we are going to make use of a **Grid Layout** since the distance between each one of them and the space for each match is going to be the same so we won't need a gridBagLayout. To put the names of each team and others, we will make use of the **JLabel** and, in case there are many matches at the same time what we are going to have to do is to use a component that is called **Scrollbar** or in its defect a container called **ScrollPane**.

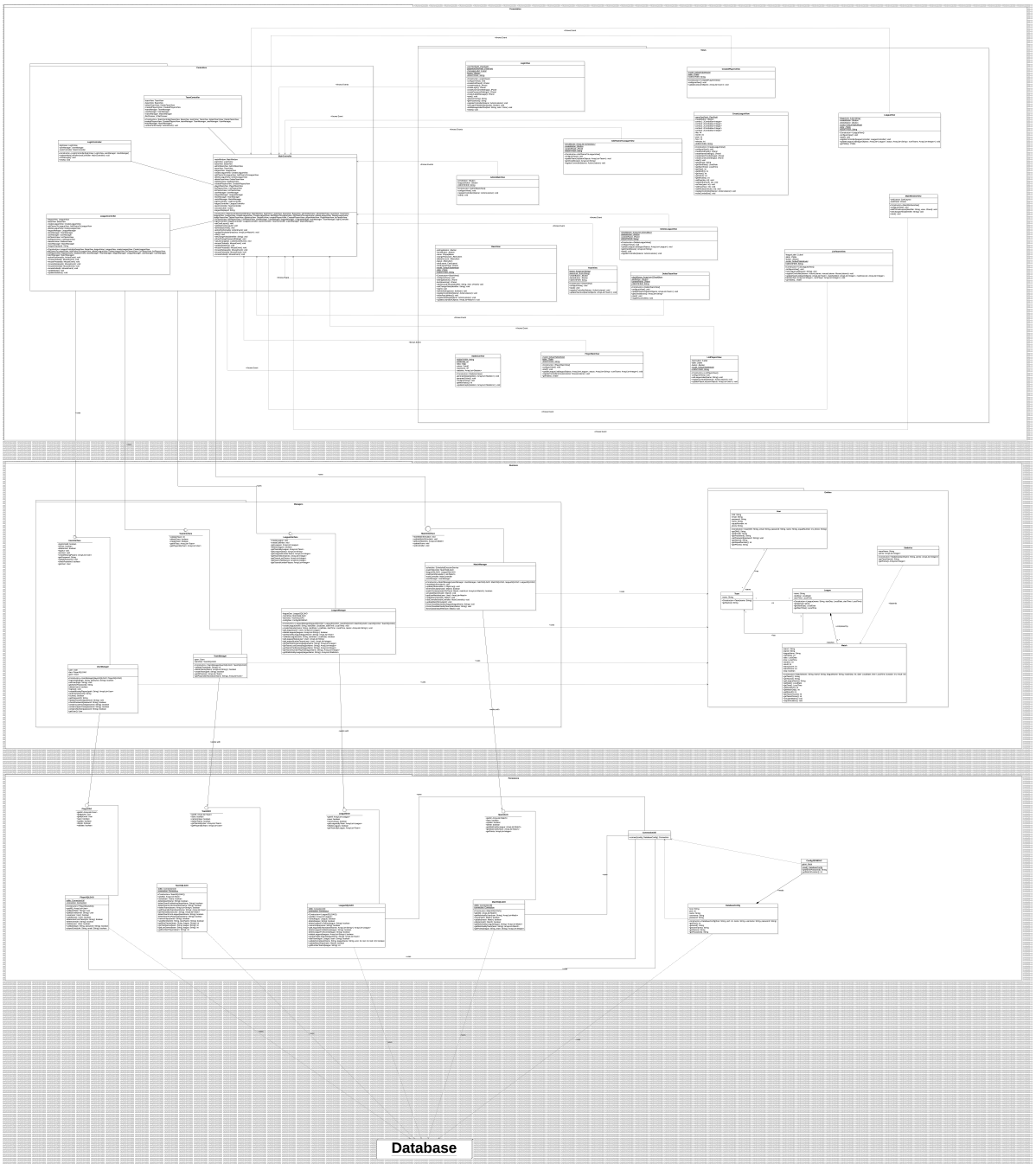
2.13 Settings Player



The Settings is not a screen but we believe that we must explain well what is in this **JMenu**. Basically, once the user clicks on the setting icon, 3 options will appear, which are the ones shown in the image (not in this form but in a dropdown). These options will appear as long as he enters as a regular user, otherwise only the "logout" option will appear.

To add these options to the **JMenu** we have had to make three **JMenuItem**, one for each option that we wanted to put. As we have said before, in the case that it is the admin who clicks on the settings icon there will only be one option and therefore we have only made use of one **JMenuItem**.

3 CLASS DIAGRAM



Our design is organised using layeredArchitecture. It consists of three major layers: presentation, business and persistence that communicate with each other.

Presentation includes all the controllers as well as all the views that are presented in the program. In total the program consists of 15 views which are: addTeamsToLeagueView, adminMainView, baseView, createdPlayersView, createLeagueView, deleteLeagueView, leagueView, listPlayersView, listTeamsView, logInView, mainWindow, playerMainView, statisticsView and finally teamView. We have 3 controllers (leagueController, teamController and loginController) that communicate with the mainController to decide which view should appear.

The business package contains on the one hand, all the managers with their respective interfaces (leagueManager, userManager, teamManager and matchManager) and on the other hand, all the necessary entities: Match, Statistics, User, League, Team.

Finally, persistence contains all the SQLDAOs of player, team, league and match with their respective interfaces. Also included are the connectionUtil, configJSONDAO and databaseConfig classes that allow us to connect java to the database.

Regarding the relationships between classes, let's start with presentation. In this layer, each and every view is connected to the mainController by means of a dependency relationship. This main controller, in turn, has an association relationship with the other controllers so that they can communicate with it and tell it which view to put.

To connect presentation with business, a dependency relationship is established from the whole block of controllers to the manager interfaces. The controllers depend on the managers to know what information to put in each view.

If we move into the business layer, the manager interfaces are connected to their respective managers through a realisation interface. In turn, all managers are connected through a dependency from the managers to the whole package of entities.

The most complex part lies in the connections between entities. On the one hand, we have that a user can belong to more than one team and therefore, a team can contain different users. A team can play in more than one league, and a league can contain several teams (although the same team cannot be added to the league twice). Each league consists of a series of matches from which statistics are generated. These statistics depend on the matches.

To conclude, business with persistence is related by associations from each manager to each sqlDAO interface in order to get the information from the database. These interfaces have a realisation interface to their respective DAOs. In turn, each DAO is associated with the connectionUtil class, which depends on both ConfigJSONDAO and databaseConfig. ConfigJSONDAO and databaseConfig are associated with each other.

4 DEVELOPMENT METHODOLOGY

4.1 SPRINT 1

VALERIA: For this first part of the project, I have shared the mockups and class diagram tasks with Laia. Each of us did half of the work. Eugènia also helped when designing the frames. First, we designed the mockups in Figma, and then, with the help of the iPad, we detailed the JComponents we were going to use. Regarding the class diagram, I was in charge of the business layer while Laia focused on the presentation one. Finally, once we had both the config.json and the database design, Laia and I finished the persistence layer and made all the missing connections. I also worked in the introduction for the final report.

EUGÈNIA: In this sprint, I created a database in PostgreSQL configuring its name, IP address and port, together with the Username and its password. I've also added some classes in IntelliJ to read the config.json file and join the project with the database. Finally, I have helped in some design layouts.

LAIA: In this first sprint, I did two different things since I have shared most of the tasks with my other classmates. In particular with Eugènia and Valeria. I have worked a lot with Valeria with the mockups design and layers but also with the diagram. Eugènia also did with us the figma mockup. For the diagram we decided to divide it into three main packages (presentation, business and persistence) and each of us three focus on one part. Since Valeria and I did most parts of the layouts from the figma prototype we decided to take the presentation and the business part and since Eugenia was going to work more on the configuration file she was going to do the persistence part of the diagram.

DANIEL: In this sprint along with my teammates we designed the database system to persist all of the data in our league manager program, for this we first theorized on the necessary tables, and we drew an entity relational model, after that we passed it to a relational model and implemented it using pgAdmin with postgreSQL language. Along with this we also created another .sql file to be able to insert some mockup information.

SEBASTIAN: In the first sprint I collaborated with Daniel in the design of the database based on the project requirements. We created a Miro desgin which functioned as a base for the later creation of the DAOs and their interaction with the postgreSQL database.

4.2 SPRINT 2

VALERIA: In this second sprint, together with Laia, we have been in charged of the design of the graphical user interfaces (GUIs). Our task has been to code each of the views that we previously designed in Figma. We started with the card layout as it was the basis for all the others and then, we proceeded to work on the ones that followed. Once we had them, we organized all the code so that we could navigate between them. I also worked with the report in the part of explaining the GUI.

EUGÈNIA: In this sprint, I helped doing the navigation view and card layout, and didn't advance too much on the tasks that were assigned to me, more logical ones.

LAIA: As Valeria has said I have been working with her with the implementation of the GUI's. We started thinking how we should do it and we thought that the best way was applying a CardLayout since almost all of the views had the same basic things and the only thing that was changing was the West part of the BorderLayout. So what we did was a class with the base view and then the other views implemented a JPanel instead of a JFrame since we were doing a CardLayout. Together with Valeria we started the report by explaining the GUI and how we were supposed to do them.

DANIEL: In this sprint I worked on the logic necessary for a user to log in, for this I first decided to implement the CRUD operations both for the PlayerSQLDAO and also for the MatchSQLDAO, adding the CRUD operations for the DAO's was a task that we divided amongst us so I did half of it. For the user log in we needed to check that the username and the password were correct and to determine whether the user was logging in as a user or as an admin. This sprint we also created interfaces for the DAO's so that we could decouple our work and so that people could create mock functions to test the other things that they were working on.

SEBASTIAN: Worked on the implementation of the Team and League DAOs and also designed their interfaces. These are now able to perform CRUD operations. Created the logic behing the login of the admin user. Coordinated with Daniel to be able to have a uniform design. We have also noticed that some changes had to be made to the database design so as to work with each more easily.

4.3 SPRINT 3

VALERIA: In this third sprint, I have been in charge of the navigation between views with Laia, although this time we have been quite delayed. We basically fixed things that didn't work before and added a home button to allow the user to go back to the main page. We also connected some of the logic functions with the GUIs in order to see if everything was working as it should.

EUGÈNIA: I continued with the card layout reordering it in order to have everything inside one view. For the CreateLeague View I found two libraries from git to add a TimePicker and DateChooser but then I realised practically everything was predefined and couldn't change much, so I opted for a JCalendar. However the listeners didn't work at all, so there's finally JComboBoxes.

LAIA: In this third sprint what we did was the navigation of the GUI's with Valeria. We know we were running late but we focused on doing this so after this we had to run and help the others as much as we could. I also worked on the statistics view and finding the way of doing a graphic without external libraries.

DANIEL: This sprint I implemented the team creation from a json file and also the league creation logic, for the team creation I created three functions in which the path of the file must be sent, the first one validates the team, returning 0 if there are no errors, 1 if the team name is not unique, 2 if the players email format was incorrect, 3 if any of the players squad numbers were negative and 4 if the dni formats were incorrect, this was important to keep in mind as it helped us later on when displaying the different pop ups to the user. The second function creates the missing players for a team, and it returns an arraylist of the newly created players. And finally the create team function that simply adds this team to persistence. This spring I also implemented the league creation in which I need to create the league, add the teams to the league and also create all the necessary matches in a double round robin fashion.

SEBASTIAN: Worked on the implementation of several methods for the Teams and the League's managers. A user can now delete its account and when this happens the user is also deleted from the teams it belongs to and from the RAM. The system can also delete teams and all their information both in the database and in RAM. Also worked on the logic behind the retrieval of leagues differentiating between whether the user is an admin or a regular user.

4.4 SPRINT 4

VALERIA: In the last sprint we worked more as a group because the deadline forced us to put together everything we had done. To start with, we continued linking the logic with the visual part (specifically the statistics and the live streaming that included the threads part). Finally, Laia and I improved the class diagram including the modifications we made with respect to the initial design and we finished writing the report.

EUGÈNIA: In this fourth sprint, I have created ListPlayersView and ListTeamsView to be able to connect it with the logic and database. Also, in a meeting we did, Valeria and I changed the LeagueView into a JTable instead of a JList to be able to print everything. This also made us move some listeners and change the controller depending on the user entered. Moreover, I've been doing some views descriptions for the final report.

LAIA: In this fourth sprint I mostly worked with Valeria and Sebastian and Daniel everyday on connecting everything to be able to finish the project on time. Another of the things I have worked on has been in the statistics view making a hardcoded graph to see how it worked and then with the help of Daniel and Sebastian we managed to draw the graph with the data from our database because at the beginning it did not draw anything but then we realized the error and we managed to fix it.. I also keep doing the report. Another thing that I have been working on is the new diagram that is the first diagram that we did but adapted since while we were coding we thought it was better to do different approaches. I did the diagram with Valeria and Sebastian.

DANIEL: This sprint I implemented all necessary logic to simulate a match, this meant checking which matches needed to be simulated and when, simulating them in real time, and also I implemented the necessary functions to be able to retrieve the matches that were being simulated which was used for the visualizing match simulations. I also did all necessary logic to display the leagues information in the league view, then the logic for the information to be displayed in the teams view, which shows when you click a league in particular. Then I also worked on the statistics view and the updating in real time of this view and all other views that required to be updated in real time. Finally I also worked on a set of functions that were necessary to be called when deleting a league or a team so that any match currently in simulation gets deleted and stops simulating.

SEBASTIÁN: This last sprint we have been working all together. I have mainly focused on the logic related to the backend of the project, specially creating additional getters based on the different classes that we have like retrieving the teams based on the league name or retrieving the players based on the team name. However, I have also assisted in the implementation of navigation between views, the design of the views and general design of the system as a whole. We have been gathering presentially to give support between each other and manage to finish the project in a timely manner.

5 TIME COSTS

In this section of the report we will record the hours we worked on each section of the project as well as the total hours for each of us and the sections. The sum of hours will be the total time invested in this project.

5.1 Use of resources

First we present a table where you will be able to see the breakdown of hours per team member for each important section of this project. In total there are 7 sections and each column is a team member. The **TOTAL** column on the right is the total number of hours we have been working on each of the sections while the **TOTAL** row at the bottom of the table is the total number of hours invested by each team member.

Stage	Laia	Eugènia	Valèria	Daniel	Sebastián	TOTAL
GUI Design	3	1	2			6h
DDBB Design				3	3	6h
DDBB Implementation				22	20	42h
Class Diagram	6		5		4	15h
GUI Implementation	17	20	22	3	3	65h
Logic Implementation	39	25	48	45	40	196h
Reporting	9	2	3	1	1	16h
TOTAL	74h	48h	76h	77h	71h	346h

On the other hand, we also provide a graph where you can better see the total number of hours invested by each member of the group in this project.

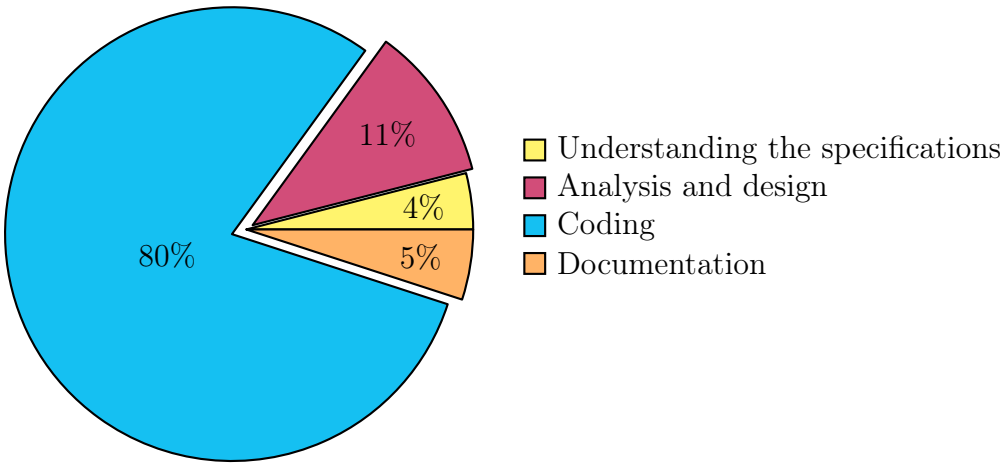


Figure 1: Pie Chart

6 CONCLUSIONS

This project has not been easy at all. Everything has its complications, but thanks to this, we have learned several things, both personally and educationally.

Speaking at a personal level, we can highlight the fact of being able to trust others when tasks are assigned to them, as well as the importance of communication among all of us. At first, this aspect was a bit challenging because we simply did the tasks assigned to each one of us, and on the day of our meeting, we would see what each person had done, and that was it. This approach stopped working when we had to implement much more code. That was when we started sharing what we were working on and, if we ever got stuck, we would let each other know, and together we would find a solution without any problems. We also had joint meetings on Mondays (when we didn't have a meeting with the professor) to catch up with each other, know what was still missing, and what had already been completed and functioning.

In Sprint 4, the final one, since none of us knew much about threads to work on a task individually, we decided to tackle it together because if five of us were brainstorming and researching, it would be much easier to understand and apply it in the project.

In terms of knowledge, this project has helped us to know how to better use applications that we had never used until this year, such as Jira and Bitbucket. Some of us did not have much practice with these tools and we have had to learn as we did the project as well as when it comes to fixing bugs like now the merge conflict or others like it.

For some things that we did not understand very well how they worked we made use of the artificial intelligence ChatGPT to understand how the CardLayout worked and other theoretical questions we had.

We also made use of the oracle java documentation web page as it provided us with both information and examples of what we wanted. With this we learned a lot in terms of java knowledge and all the functions it has.

It has been a first contact to use Swing since we have never used it and we have seen that what seems easy is really not so easy. And it has also helped us to realize that working in a team is not easy either if there is no good communication between team members as well as trust in others.

7 Bibliography

References

[1] OpenAI (2022), OpenAI Chat: http://chat.openai.com

[2] Laing Raven (2021), Time Picker: https://www.youtube.com/watch?v=fRib6RpGmiE&t=283s

[3] Unknown (1995), Oracle Java Documentation: https://docs.oracle.com/javase/tutorial/uiswing/layout/card.html

[4] Jeff Atwood and Joel Spolsky (2008), Stack Overflow: https://stackoverflow.com