Matteo Belenchia, Marcin Massalski

Politechnika Łódzka, international faculty of engineering  ul. Żeromskiego 116, Łódź

Admin panel component

Contents

[Purpose and range 3](#_Toc535357938)

[Requirements 3](#_Toc535357939)

[Functional requirements 3](#_Toc535357940)

[Non-Functional requirements 4](#_Toc535357941)

[Use case diagram 5](#_Toc535357942)

[Component Diagram 5](#_Toc535357943)

[Class Diagram 6](#_Toc535357944)

[Use case implementation 7](#_Toc535357945)

[Join chat 7](#_Toc535357946)

[Read chat 8](#_Toc535357947)

[Send message 11](#_Toc535357948)

[Delete message 12](#_Toc535357949)

[Leave chat 13](#_Toc535357950)

[Set max players 14](#_Toc535357951)

[Set minimum bet 15](#_Toc535357952)

[Close Game session 17](#_Toc535357953)

[See running Games 18](#_Toc535357954)

[See connected Players 19](#_Toc535357955)

[Edit Player 20](#_Toc535357956)

[Ban Player 21](#_Toc535357957)

[Refresh stats 23](#_Toc535357958)

[Show game specific stats 23](#_Toc535357959)

[Show today’s stats 24](#_Toc535357960)

[Show player stats 25](#_Toc535357961)

[Show total stats 26](#_Toc535357962)

[Show history 27](#_Toc535357963)

[Activity diagrams 27](#_Toc535357964)

[Delete Message 28](#_Toc535357965)

[Send Message 28](#_Toc535357966)

[Show Player statistics 29](#_Toc535357967)

[State transition diagrams 29](#_Toc535357968)

[AdminPanel Class 29](#_Toc535357969)

[Chat Class 30](#_Toc535357970)

[DataViewer Class 30](#_Toc535357971)

[Timing Diagram 31](#_Toc535357972)

[Package Diagram 33](#_Toc535357973)

[Composite structure diagrams 33](#_Toc535357974)

[Chat Class 33](#_Toc535357975)

[DataViewer Class 34](#_Toc535357976)

[Interaction Overview Diagram 35](#_Toc535357977)

[Conclusions 36](#_Toc535357978)

[User documentation 36](#_Toc535357979)

[Chat 36](#_Toc535357980)

[Functional description 36](#_Toc535357981)

[User manual 36](#_Toc535357982)

[Complete description 38](#_Toc535357983)

[Installation description 38](#_Toc535357984)

[Admin Panel 38](#_Toc535357985)

[Functional description 38](#_Toc535357986)

[User manual 39](#_Toc535357987)

[Complete description 44](#_Toc535357988)

[Installation description 44](#_Toc535357989)

# Purpose and range

The Admin Panel component is designed to work as part of the Online Casino system, the purpose of which is the development of a web application that provides the basic functionalities of a Casino, such as playing slots, BlackJack and Roulette, withdraw and deposit money, chat with other Players and more.

The purpose of the Admin Panel component is to provide the administration and monitorization of Players and visualization of the relevant data and information present in the system, both as runtime data and database data. The component offers various administration tool such as banning Players, closing Games, limiting the number of connected Players, setting a minimum bet requirement and more, as described in the requirements section and use case diagram. At the same time, the list of logged Players and currently running games is also visualized. There also is a great deal of variety in the kind of data that can be visualized, that spans Player personal statistics to Game specific statistics to global total or daily statistics of the system as a whole. Statistics can also be visualized day by day.

Additionally, we decided to develop also a Chat, which does not belong to the Admin Panel component, but is in fact part of the User Management component. To simplify things, the Chat doesn’t use the Database Management component and does all its work internally to its own package, which is fully developed by us.

The range of the component is limited by its strong dependencies on User Management and Game Management components. All operations of the Admin Panel need to use the interfaces provided by the aforementioned components, and thus by itself the component is merely visualizing received data or formatting input data to be used in the appropriate interface, and dealing with the communication between the front-end GUI and the back-end Server.

# Requirements

## Functional requirements

The server management component shall:

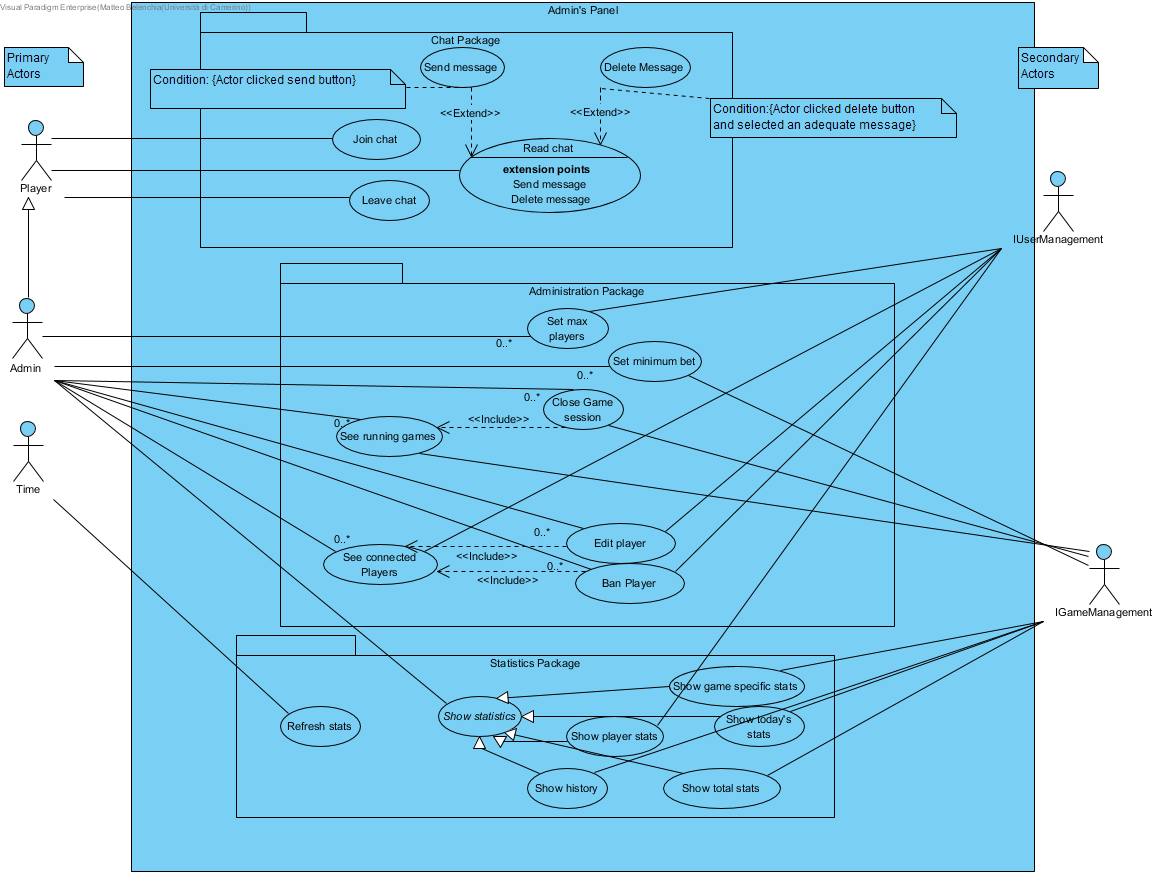
1. Let the admins ban a player until a certain time in the future.
2. Allow the admin to close game sessions
3. Allow the admin to change any attribute of any user
4. Allow the admin to change the minimum allowed bet
5. Allow the admin to set the maximum number of concurrently connected users
6. Show connected users and admin(s)
7. Show currently running game sessions
8. Show, for each game type:
   1. House losses and gains total
   2. Average and minimum bet
   3. Number of game sessions
   4. Profitability
9. Show today’s :
   1. Aggregated house losses and gains
   2. House losses and gains per game type
   3. Number of game sessions
   4. Profitability
10. Show total:
    1. Aggregated house losses and gains
    2. Number of game sessions
    3. Profitability
11. Show player’s :
    1. Anagraphic data
    2. Total losses, gains and current balance
12. Show all daily stats (history) in the last month
13. Refresh the shown data periodically without any user input
14. Allow joining and leaving the chat
15. Read latest messages in the global chat
16. Let admins and users alike send messages in the global chat
17. Let users delete their own messages as long as they are shown in the chat
18. Let admins delete any message as long as they are shown in the chat

## Non-Functional requirements

The server management component shall:

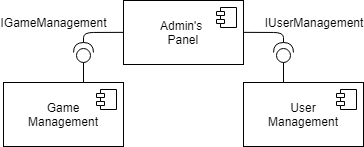
1. Let players be banned with a zero duration
2. Not allow a less than 1 minimum bet
3. Not allow to edit a Player into having an already used username
4. Not allow to edit a Player into having an already used email
5. Not allow a less than 1 maximum number of connected Players
6. Refresh shown statistics once every 5 minutes
7. Assign an ID to each chat message for identification and selection
8. Set the chat message timestamp as the server's local time upon receipt
9. Limit the number of messages shown on the chat
10. Highlight admin messages in the chat
11. Delegate the DB access for the editing of the users' data, kicking players, setting server capacity, query Player statistics and visualize connected Playersto the User Management Component by using the appropriate interface
12. Delegate the DB access for closing games, setting the minimum bet and querying statistics to the Game Management Component by using the appropriate interface
13. Have a user-friendly graphical user interface

# Use case diagram



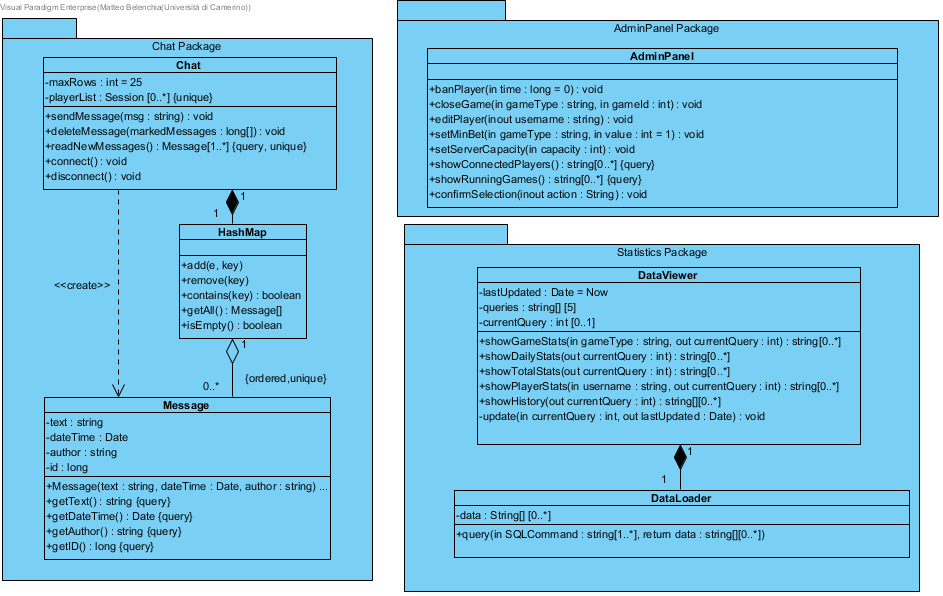
In the Use-Case diagram actors on the left side are primary actors while those on the right are supporting actors. The actor Admin inherits from the more general Player actor as it is a more specialized role with additional use cases but which can still use all the features reserved for Players. The Time actor is used to trigger the Refresh stats use case. The supporting actors are associated with the use cases that depend on them for their functionality. The use cases are split in 3 packages. The package Chat contains the use cases relevant to the chat. Send message and Delete message are use cases which provide extended functionality to the Read chat use case, as in our analysis these functionalities only make sense after having read the chat. In the Administration package there are various use cases related to administration functionalities. We modeled Close Game session to include See running games because after closing a game we’d need to refresh the running games screen, and for the same reason Edit player and Ban player include the See connected players use case. The last package contains the statistics functionalities and features an abstract use case Show statistics which is a generalization of the other more specialized use cases.

# Component Diagram



This fragment of the component diagram here shows the Admin’s Panel connections with the rest of the system (which is not shown fully). The component uses the IGameManagement interface to relate with the Game Management component and the IUserManagement interface to communicate with the User Management component. The component does not provide any interface.

# Class Diagram



The class diagram is split into 3 packages.

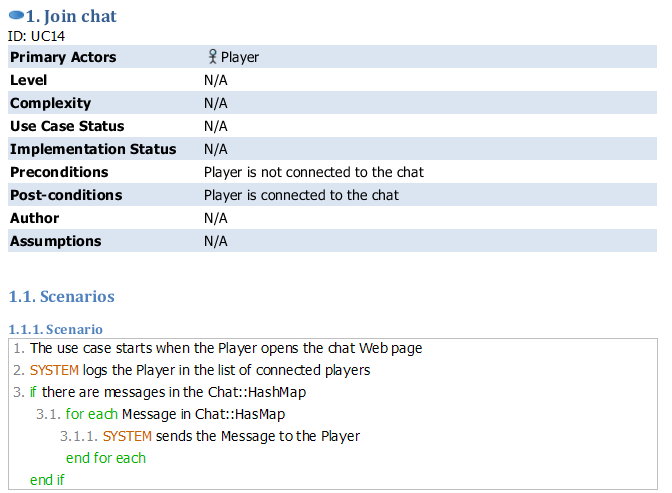
The Chat package contains the Chat class which is the main class for the Chat functionality. The Chat class provides methods to connect, disconnect, send, delete and read messages. Its attributes specify the maximum number of lines the chat has and a list of Sessions to hold information about the Players logged to the chat. The class contains exactly one HashMap object to hold the messages; the chat does not use a database connection. In the HashMap are stored Message objects which are mantained in a certain order and are at all times unique thanks to the id attribute. The Message class contains getters and a costructor method. The Message objects are created by the Chat class but are aggregated into the HashMap.

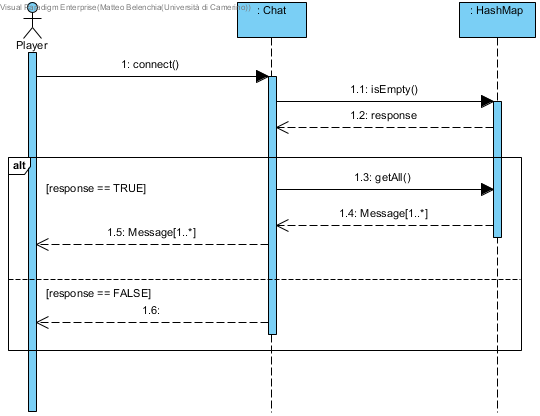
The AdminPanel Package contains only one class “AdminPanel” which contains a series of methods. Notice that the confirmSelection() method is used to confirm the changes made with the closeGame() and editPlayer() methods because they can be used to perform multiple changes.

The Statistics Package main class is DataViewer which contains the methods for the various statistics type that can be shown. These methods call update() which in turn calls the query() method of the DataLoader class, the single instance of which is contained in the DataViewer class. The various SQL queries are stored in the attribute “queries” and the one currently selected index is saved in the attribute “currentQuery”, which is used as argument of the update() method. The “lastUpdated” attribute is used to determine if the 5 minute timeout for refreshing data has elapsed. Some methods also require the type of game or the username of a player to run.

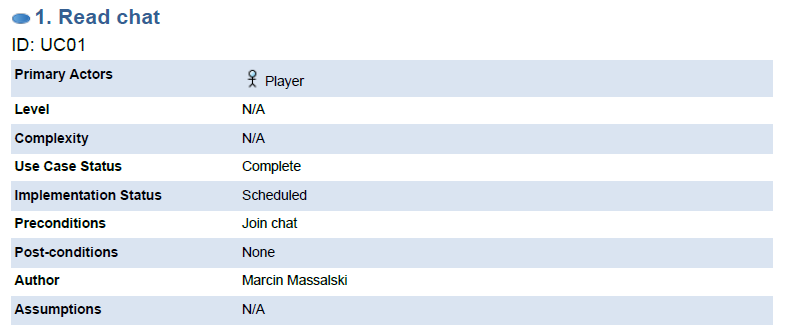
# Use case implementation

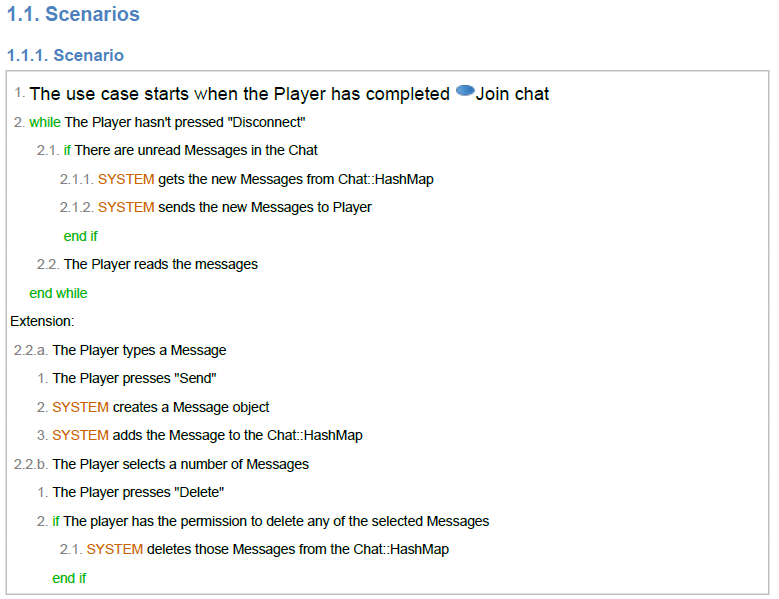
## Join chat

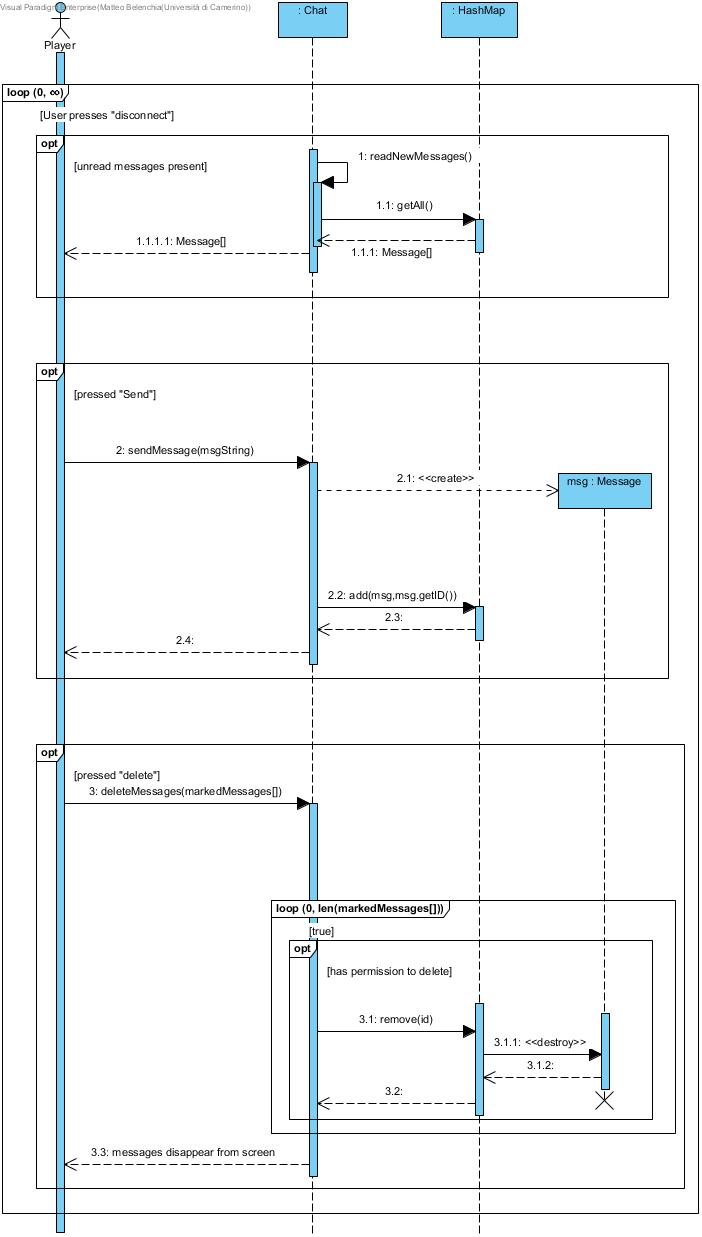




## Read chat

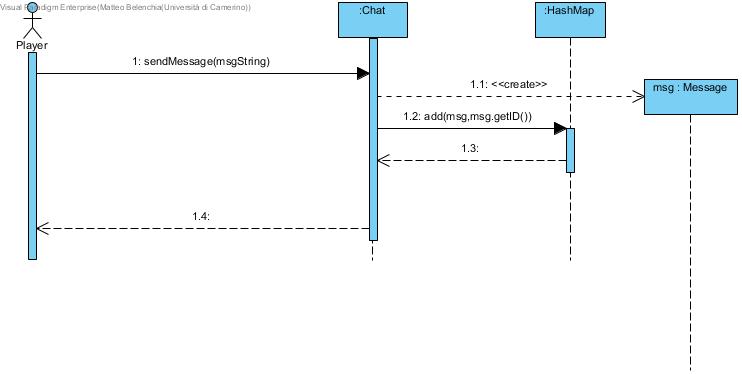




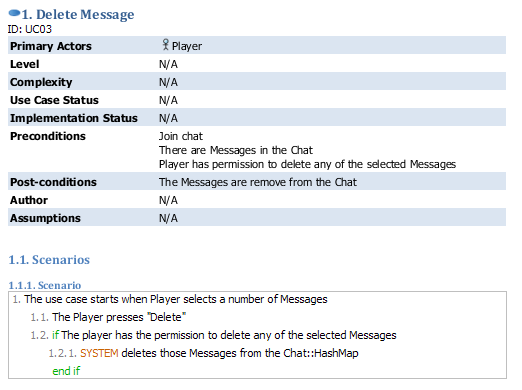


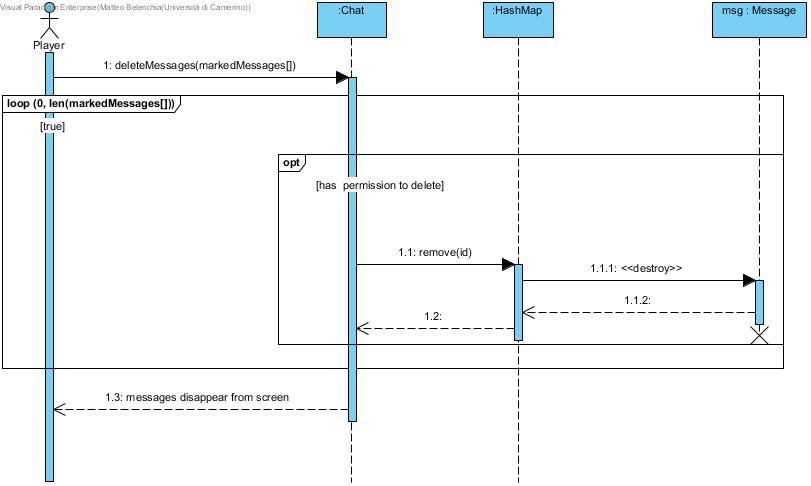
## Send message



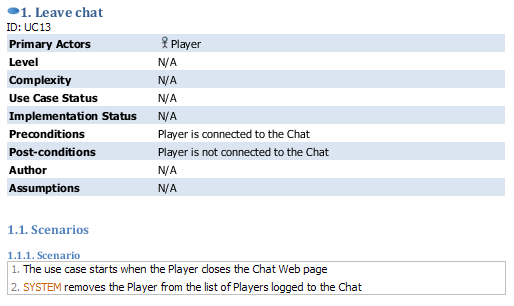


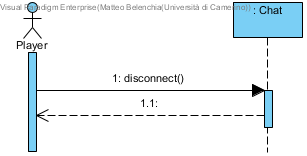
## Delete message



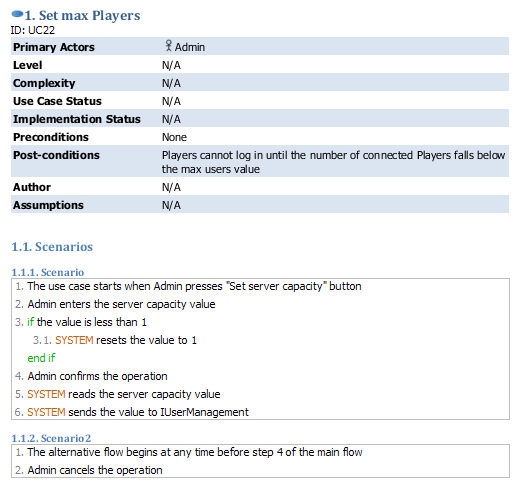


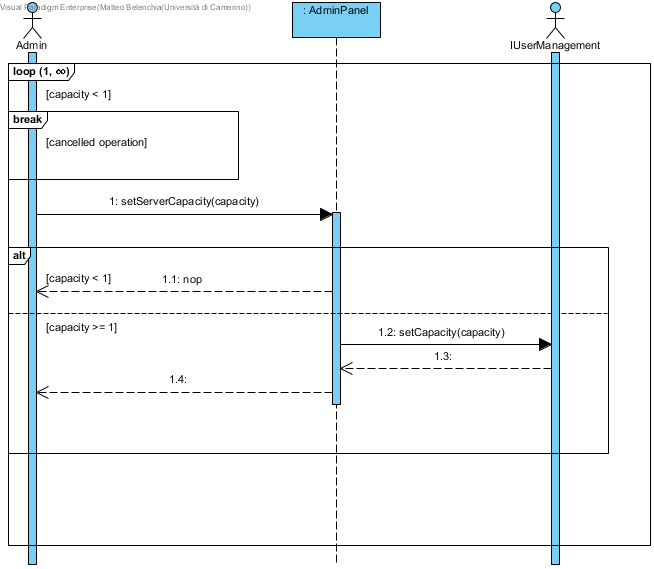
## Leave chat



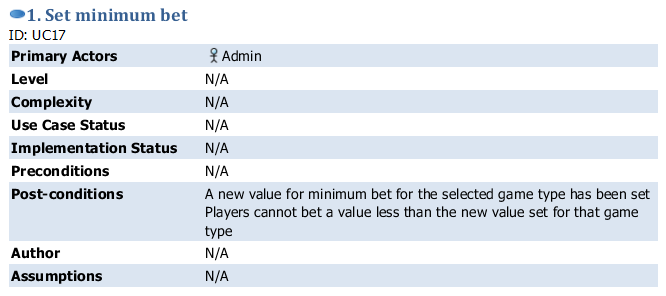


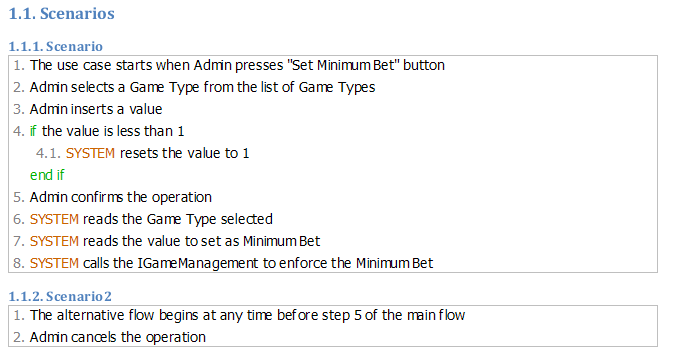
## Set max players

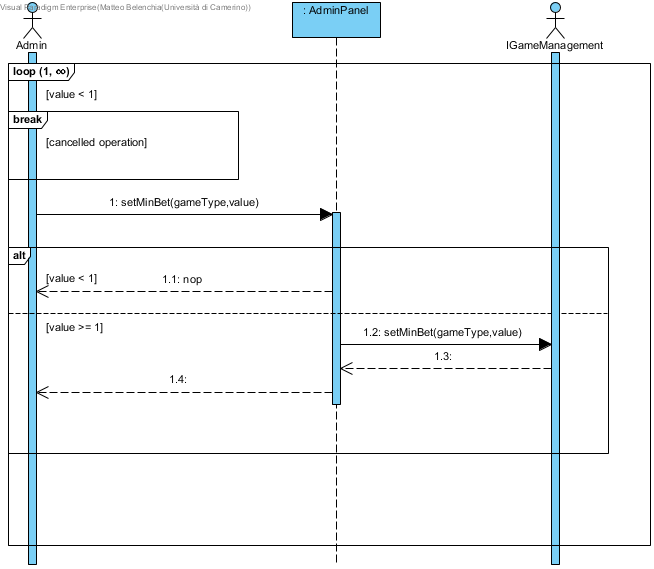




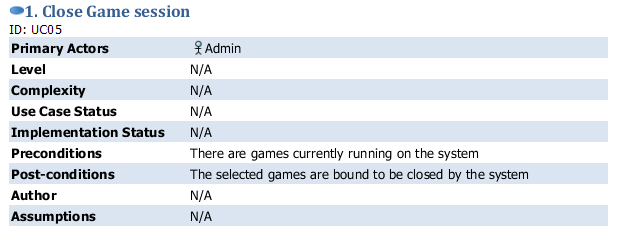
## Set minimum bet

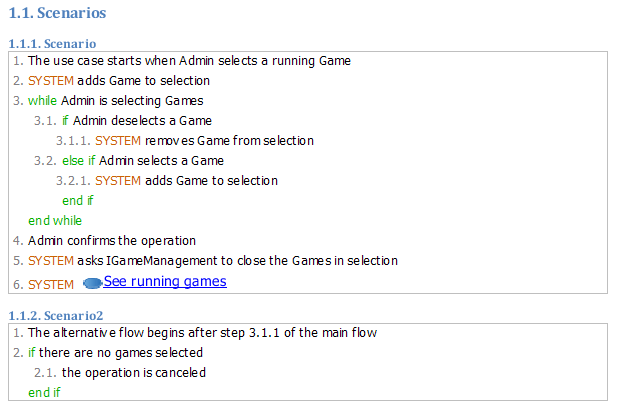


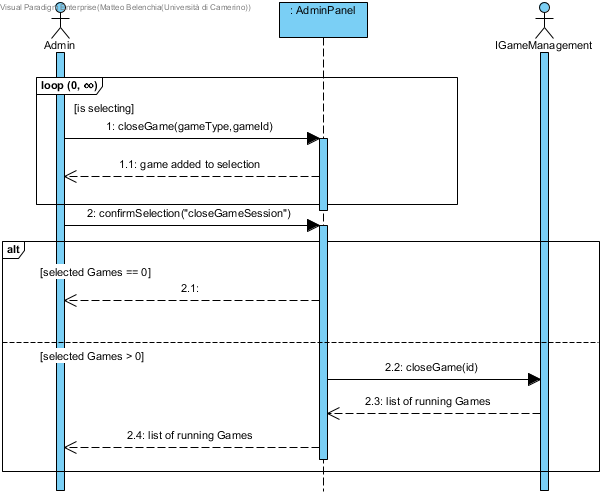




## Close Game session

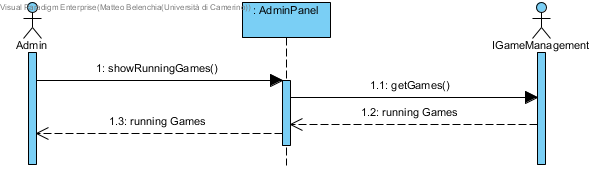




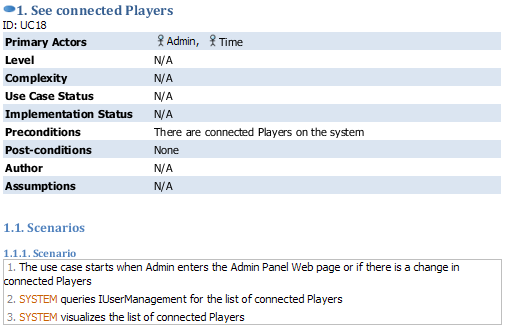


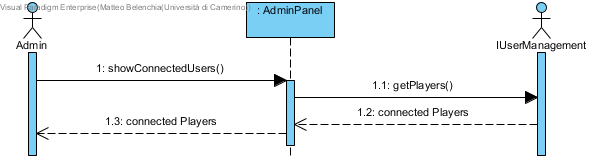
## See running Games



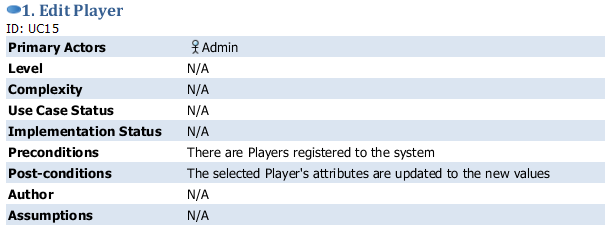


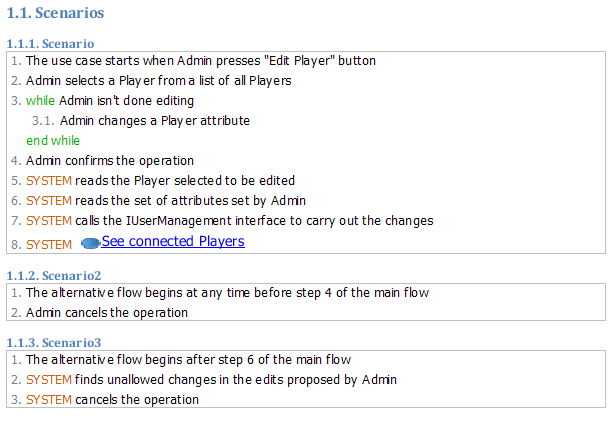
## See connected Players

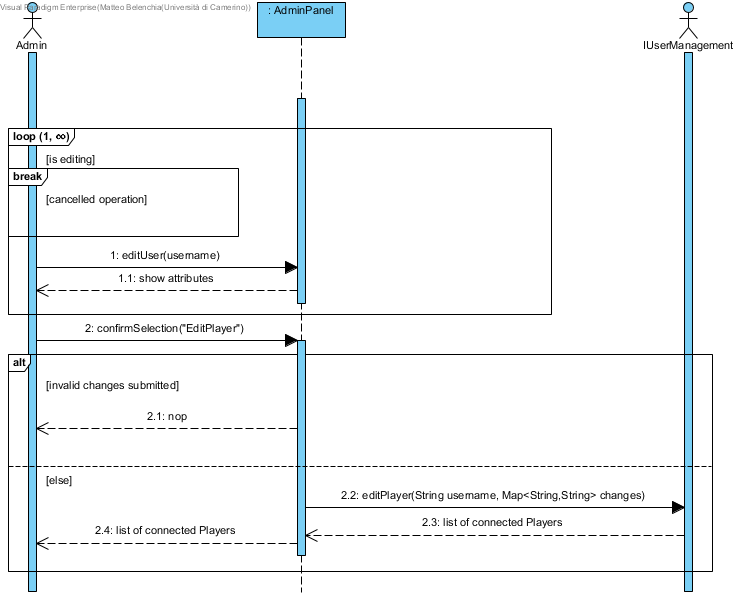




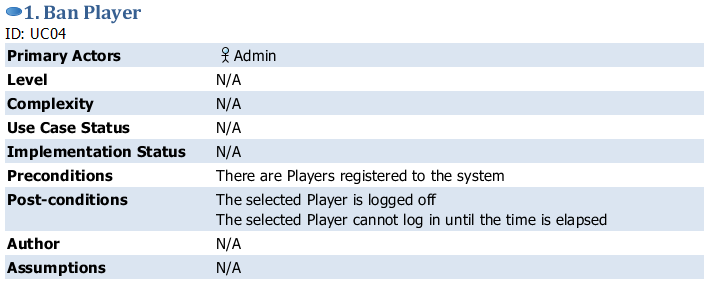
## Edit Player

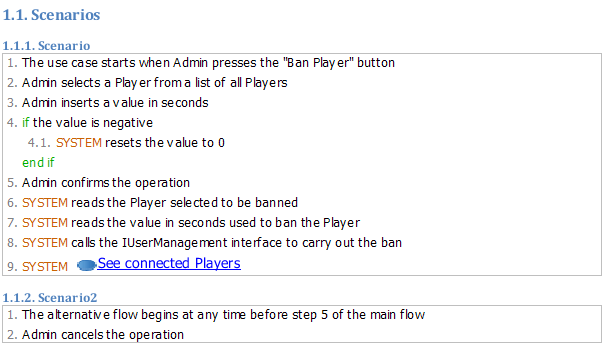


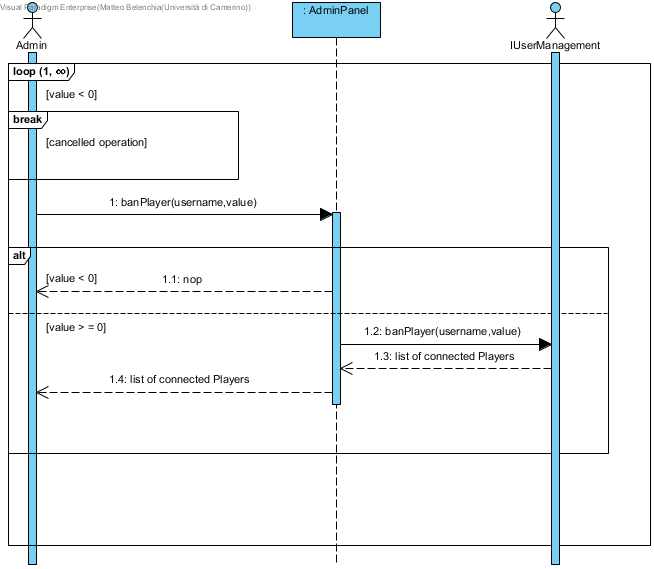




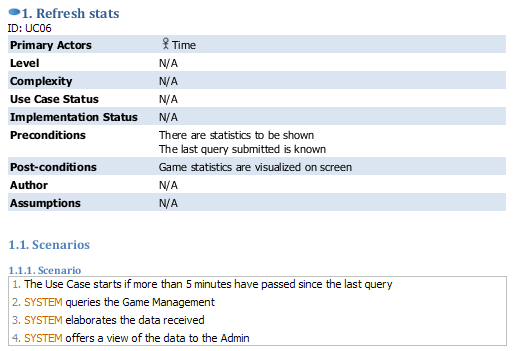
## Ban Player

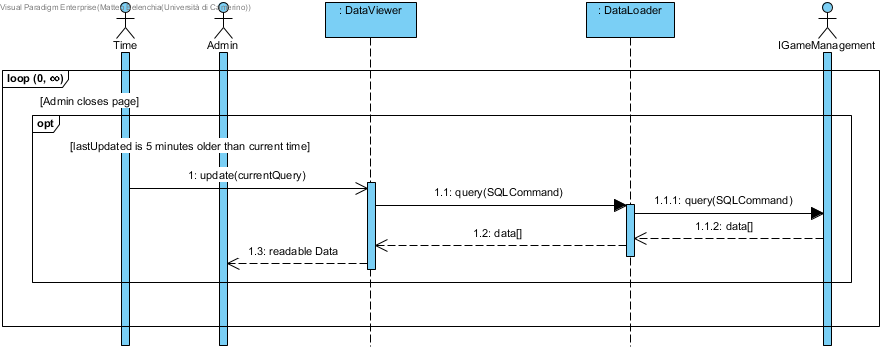




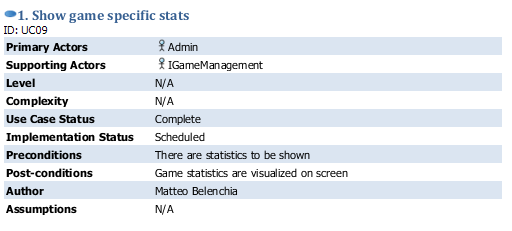


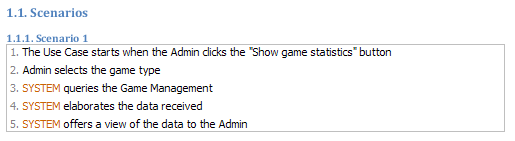
## Refresh stats

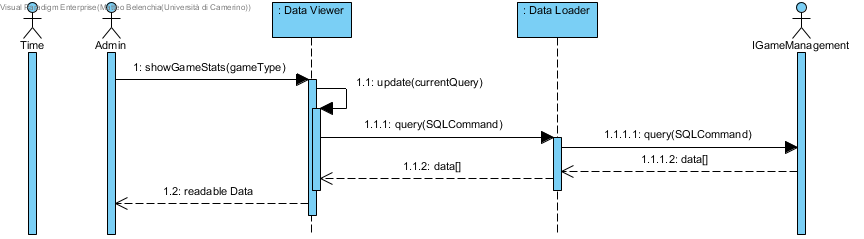




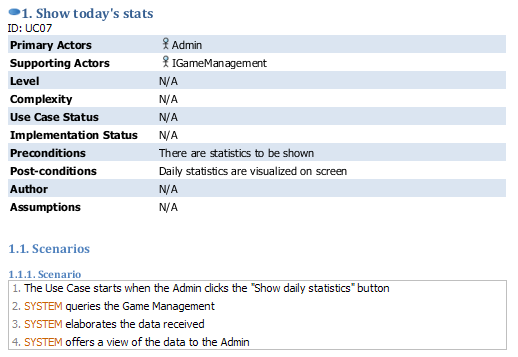
## Show game specific stats

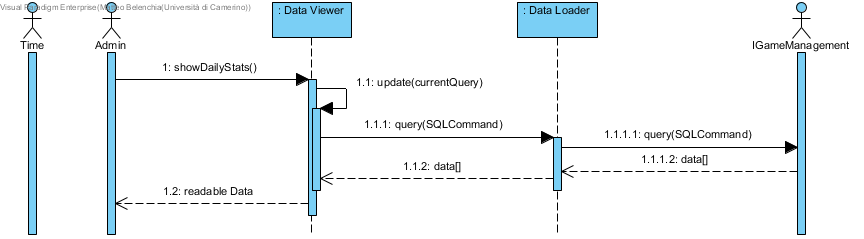




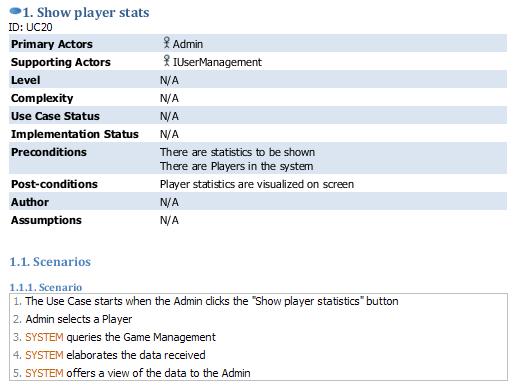


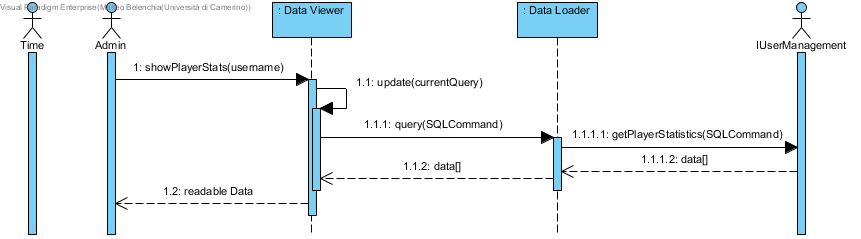
## Show today’s stats



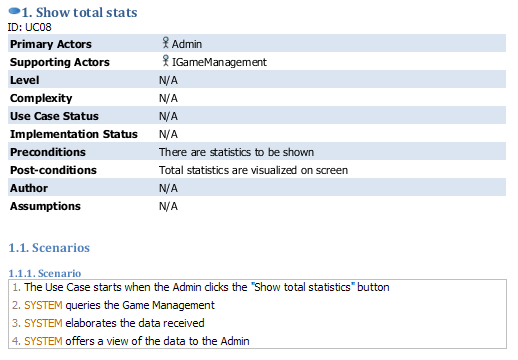


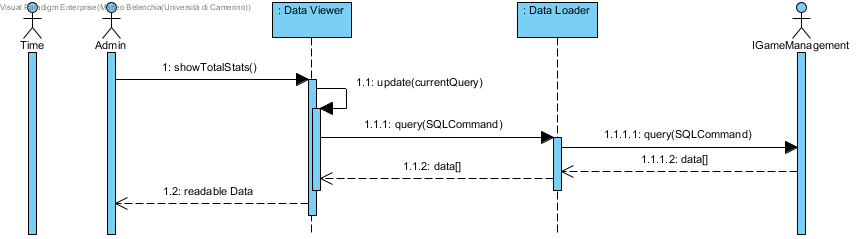
## Show player stats



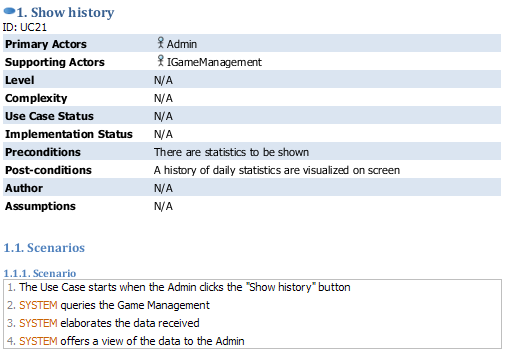


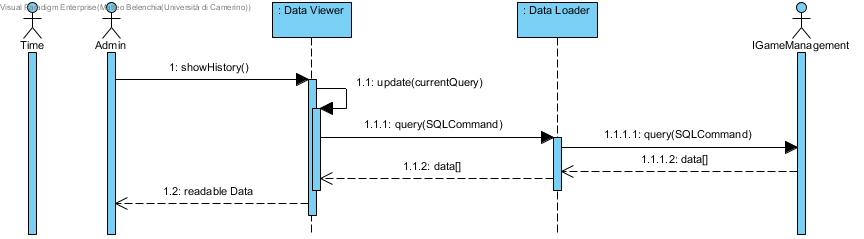
## Show total stats





## Show history

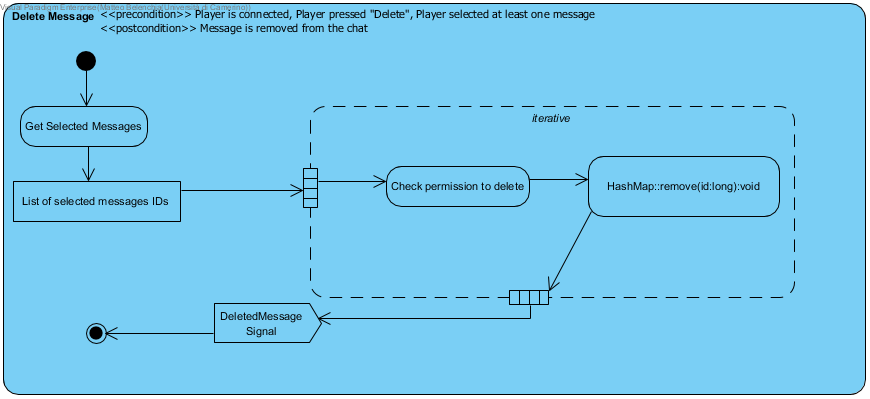




# Activity diagrams

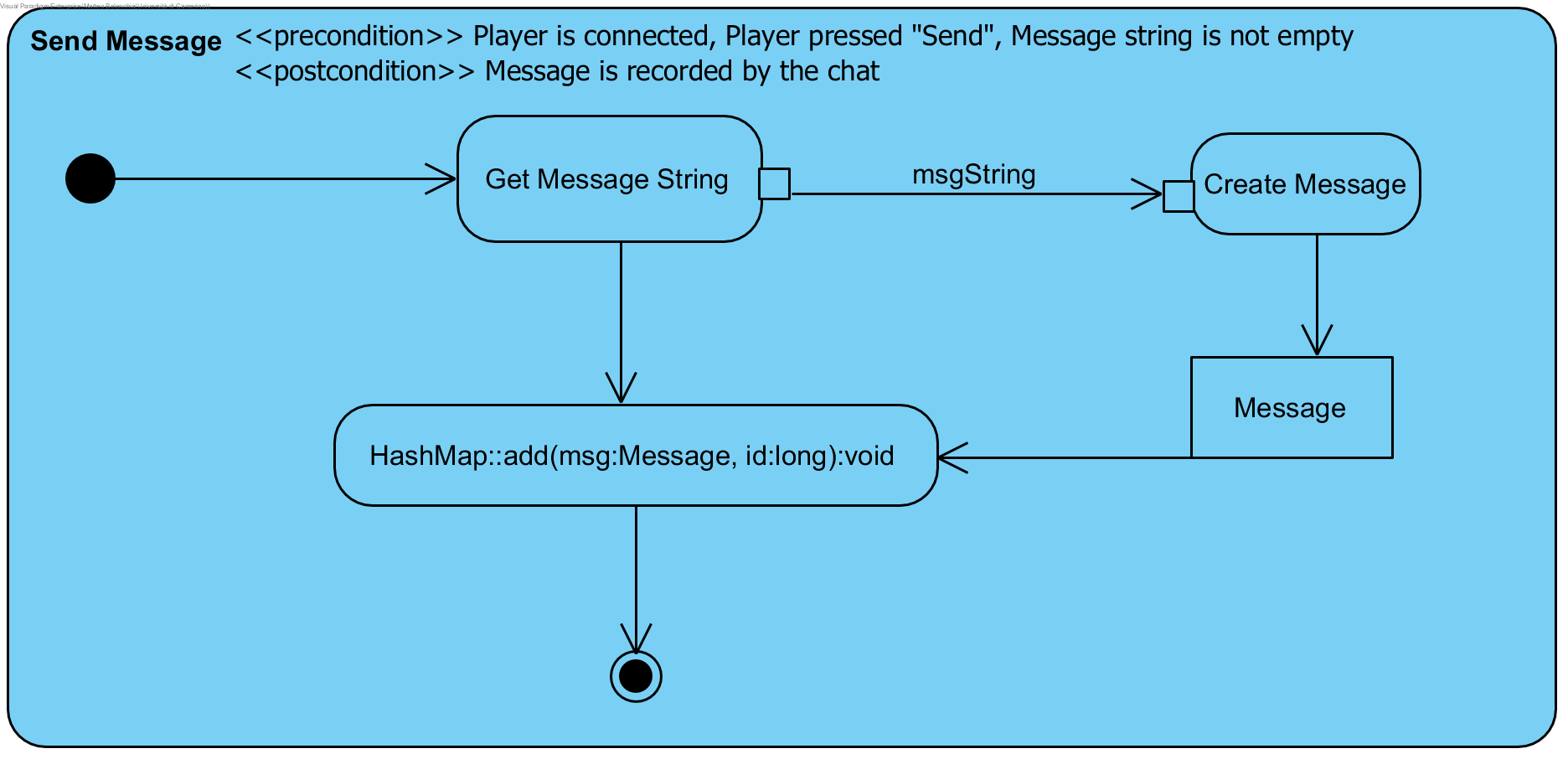
The following activity diagrams show the execution flow of the use cases “Delete Message”, “Send Message” and “Show Player statistics”.

## Delete Message



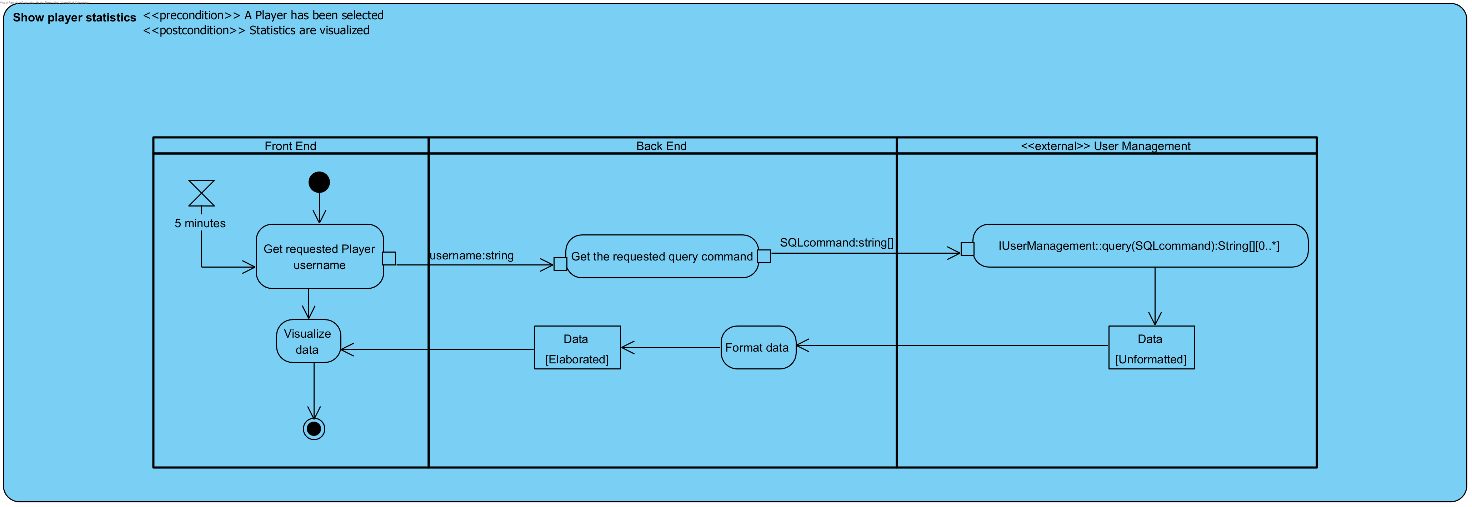
The preconditions before the activity can start is that the Player is connected to the chat, he has pressed the Delete button and selected at least one message for deletion. Once the list of selected messages is retrieved, we pass the list of the Message IDs to iteratively check for permission to delete and if permission is granted the message is removed from the HashMap. Then a signal is launched to mark that the chat has been modified and it signifies that the current chat has to be sent to all connected Players. The post conditions is that the Messages for which there was permission to delete are removed from the chat.

## Send Message



The precoditions is that the Player is connected to the Chat, it has pressed send and the Message he or she composed is not empty. The Message string is read and used to create a Message object which is then passed to the HashMap to be added to it. The postcondition is that the Message is saved to the chat.

## Show Player statistics

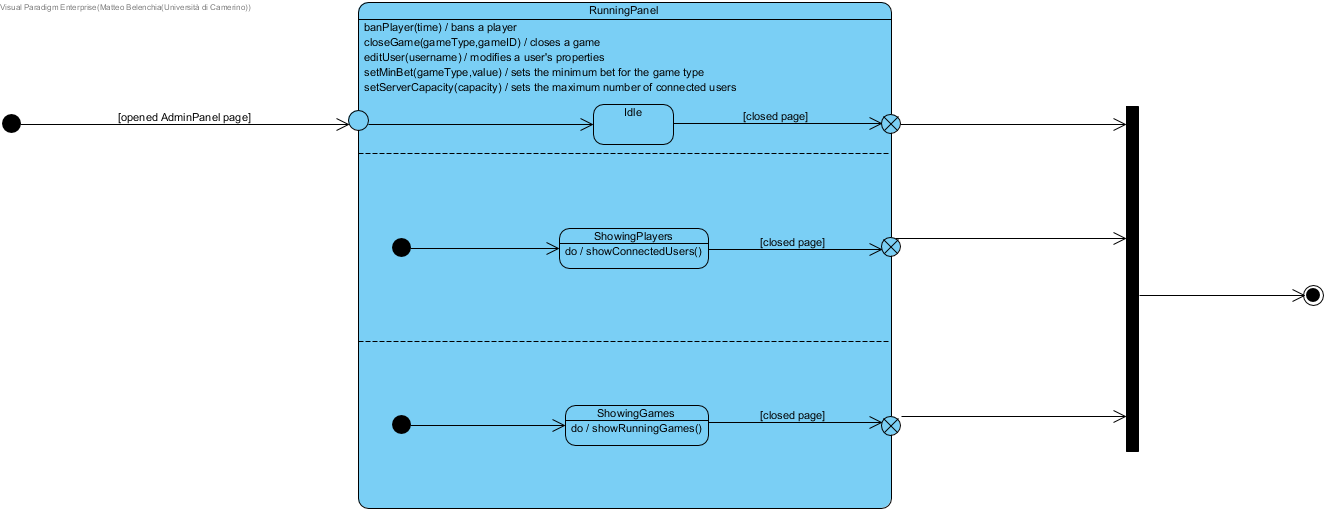


The precondition is that there is a Player currently selected. The Front End application gets the Player username and passes it to the Back End where the username is added to the appropriate query string which is then passed as argument to the external User Management component. The component then returns some raw Data which is processed in the Back End and is visualized in the Front End. Additionally, every 5 minutes the process is repeated again. The postcondition is that statistics are currently visualized in the Front End application.

# State transition diagrams

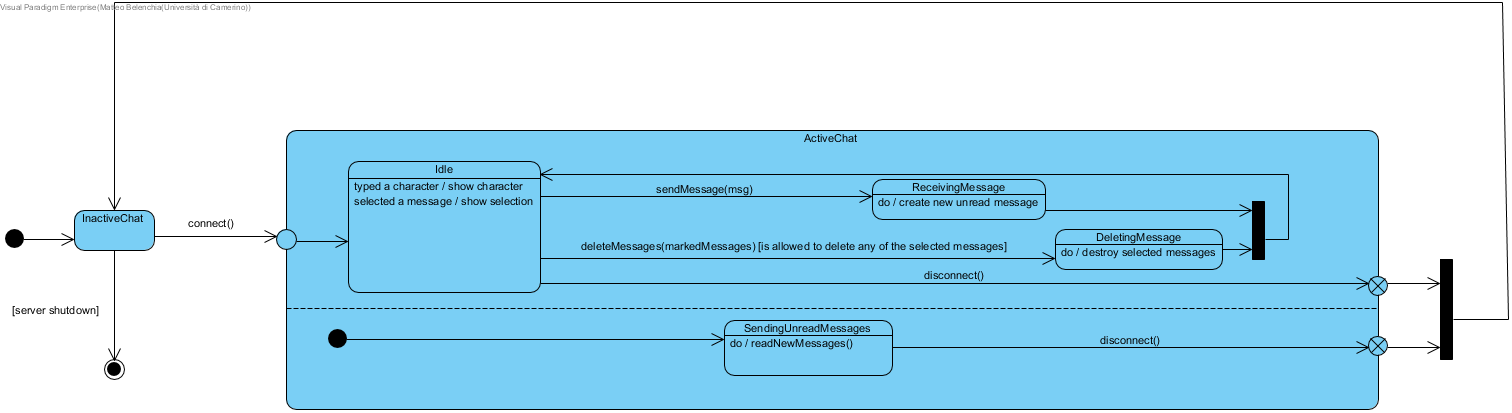
The state transition diagrams are relative to the most important classes of the component, Chat, AdminPanel and DataViewer.

## AdminPanel Class

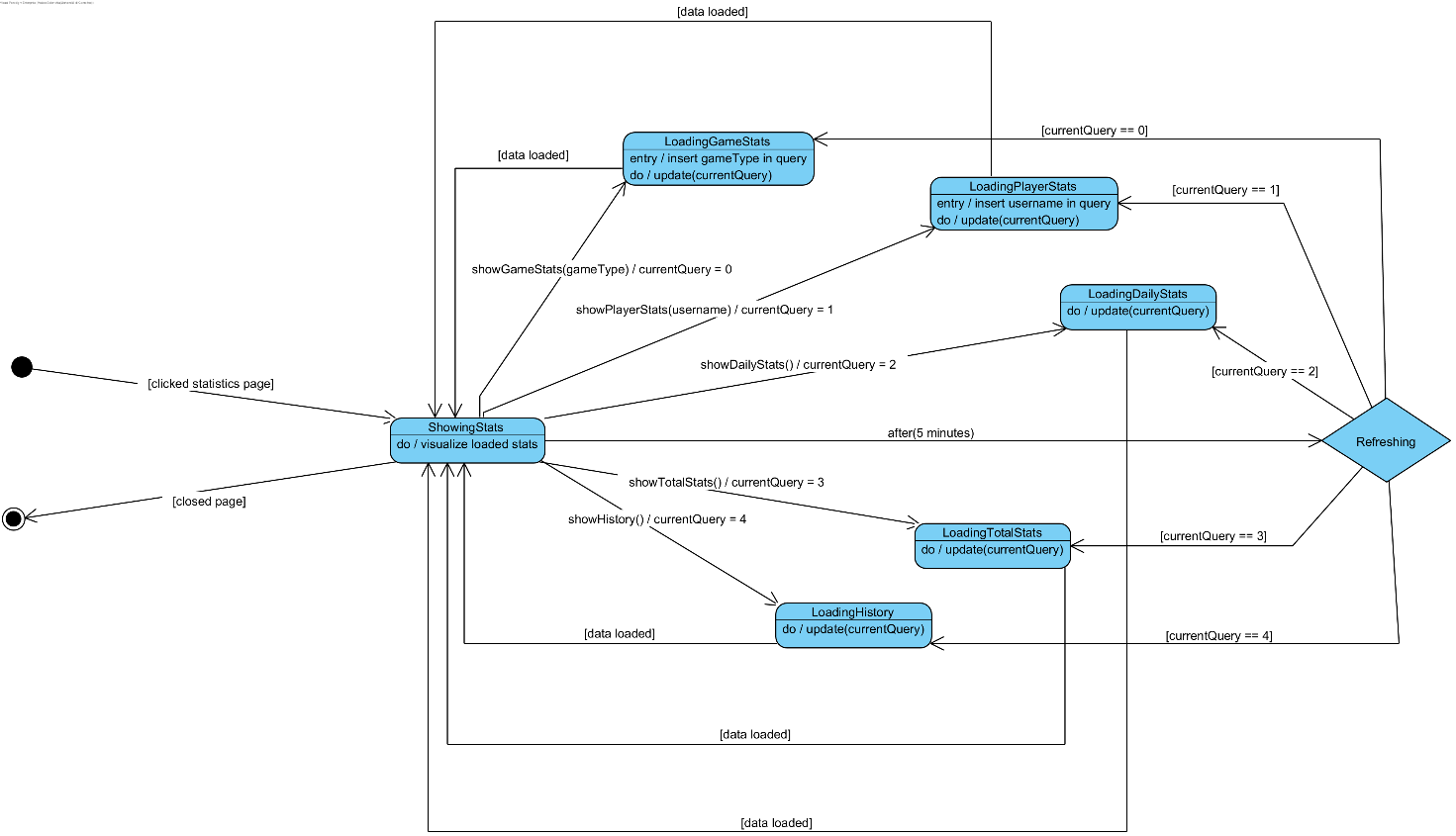


The AdminPanel class is modeled with a single composite state RunningPanel which is entered when the webpage is opened. This state is composed by 3 parallel running substates. The Idle substate inherits all the actions defined in the RunningPanel state, which correspond to the various actions defined in the AdminPanel class, and these actions do not provoke any state transition but can be repeated as often as the Admin wants. Concurrently the substates ShowingPlayers and ShowingGames run the methods showConnectedUsers() and showRunningGames(). On page close all the substates terminate and the state transitions out from RunningPanel.

## Chat Class

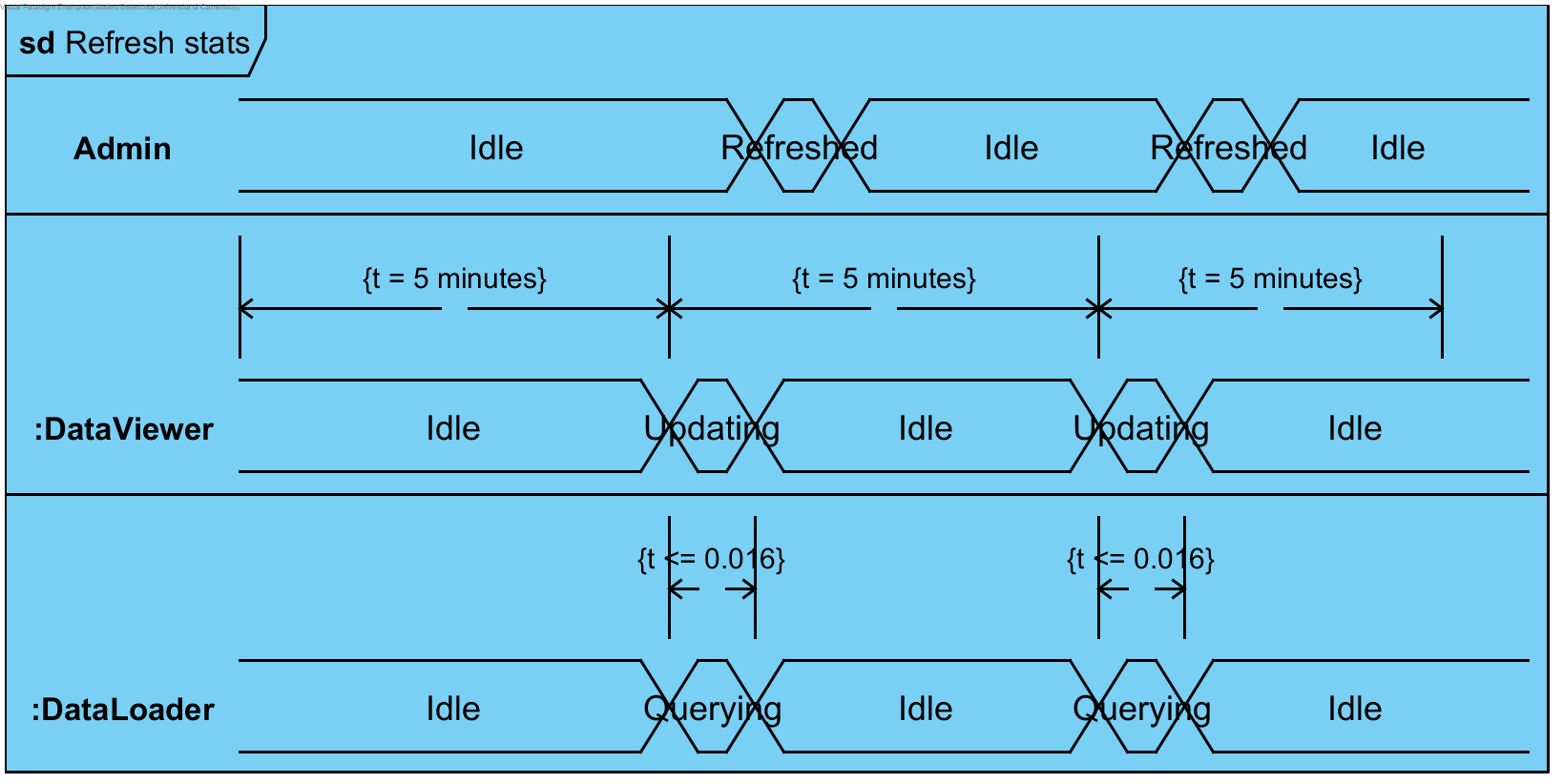


The Chat class starts in the InactiveChat state which will transition to ActiveChat when the method connect() is called. The ActiveChat state is composed by two section that run in parallel. In the top section the control goes to the Idle state which contains actions related to typing a message and selecting messages. Then when calling the sendMessage() method the state transitions in the ReceivingMessage substate in which the chat is creating the Message object and adding it to the HashMap, while instead calling deleteMessages() when the caller has permission to delete at least one of the selected messages fires the transition to DeletingMessage substate in which the Messages are removed from the HashMap. Both DeletingMessage and ReceivingMessage transition back into the Idle substate. In parallel in the bottom section runs the SendingUnreadMessages substate which runs the readNewMessages() method. When calling the disconnect method both sections exist the composite state and the current state becomes once again the InactiveChat state.

DataViewer Class

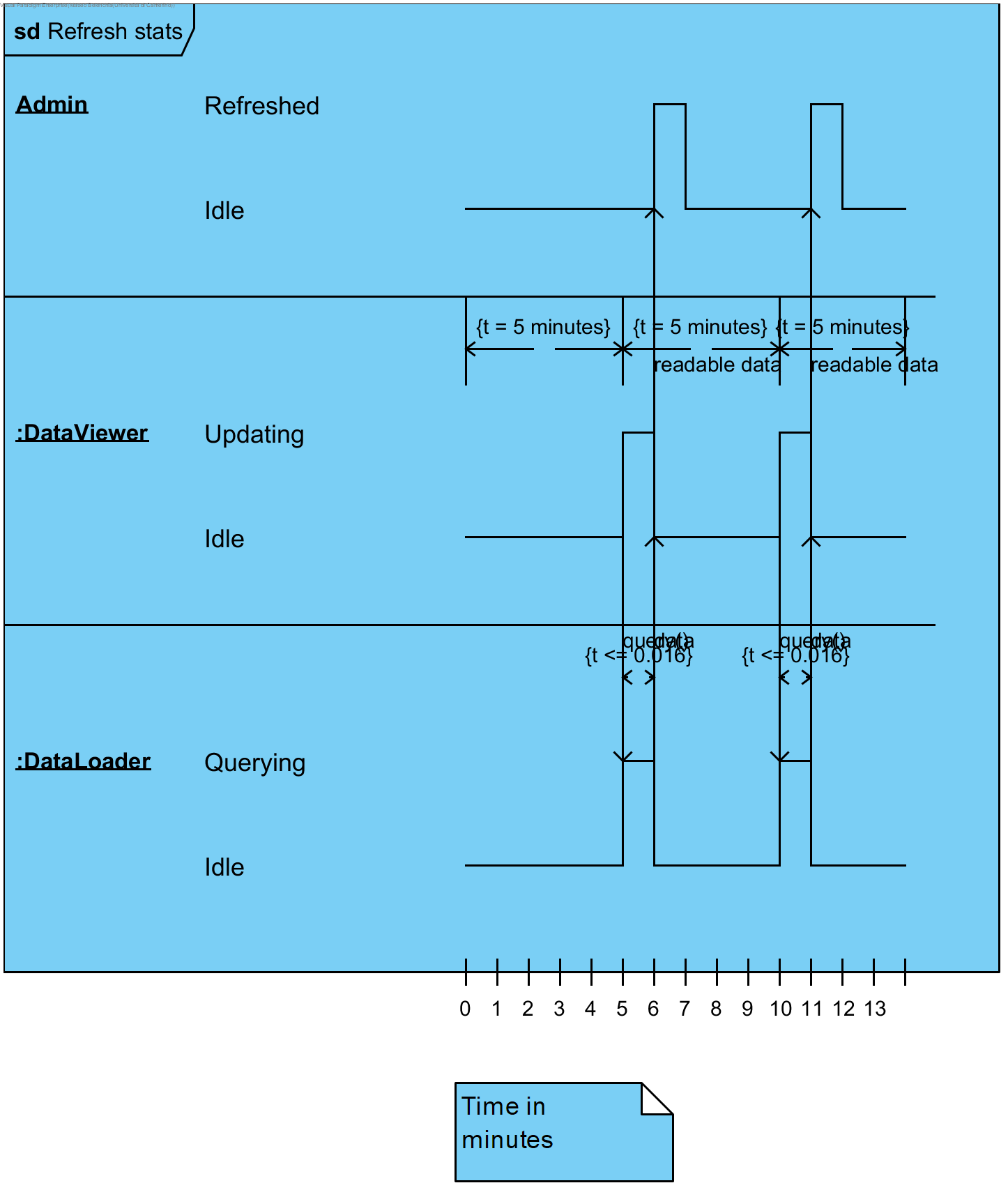
The initial state is reached upon opening the statistics webpage. In the ShowingStats state the statistics currently loaded are displayed if any. From such state, calling any of the following methods trigger a transition to the corresponding state : showGameStats() to LoadingGameStats, showPlayerStats to LoadingPlayerStats and so on. In each transition the variable currentQuery is set to a particular number which represents the corresponding index in the array of SQL queries. This value is then used in each of these states as argument to the update() method. When the data has finished loading the state transitions back to ShowingStats. From such state, when 5 minutes have passed since the last call to update(), the state transitions into the Refreshing choice, where according to the value of currentQuery the state will change into any of the Loading states previously mentioned. From ShowingStats then the State diagram terminates when the webpage is closed.

# Timing Diagram

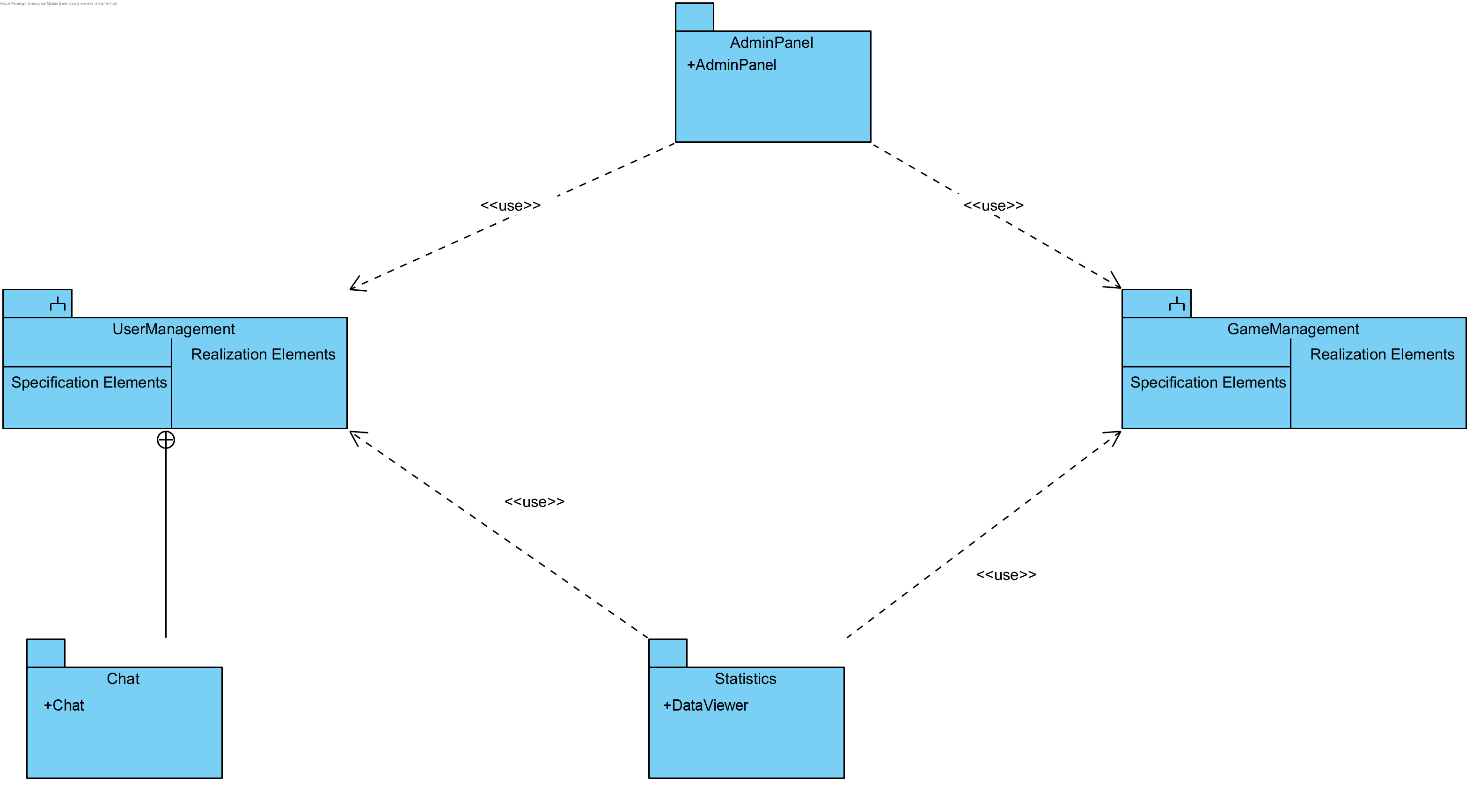


The timing diagram shows the timings relative to the use case Refresh stats. This timing diagram is in compact view because it’s easier to visualize. Looking from the central lane every 5 minutes the state of DataViewer changes from Idle to Updating, which triggers the change of state in DataLoader from Idle to Querying. Since time is expressed in minutes, there’s a constraint on Querying to finish in less than a second or in other words 0.016 minutes. This is in order to show that the Querying state is not actually taking a full minute but is a rather fast operation. After Querying and Updating states transition back to Idle, now the state of the Admin actor changes from Idle to Refreshed with the meaning that the visualized stats are now refreshed.

The following diagram is the same timing diagram but in expanded view, although the constraints make it harder to read. On the arrow going from DataViewer to DataLoader there’s the method call query() which returns data, while on the arrow going from DataViewer to Admin there’s readable data as return value. On the bottom the timing is expessed in minutes.



# Package Diagram

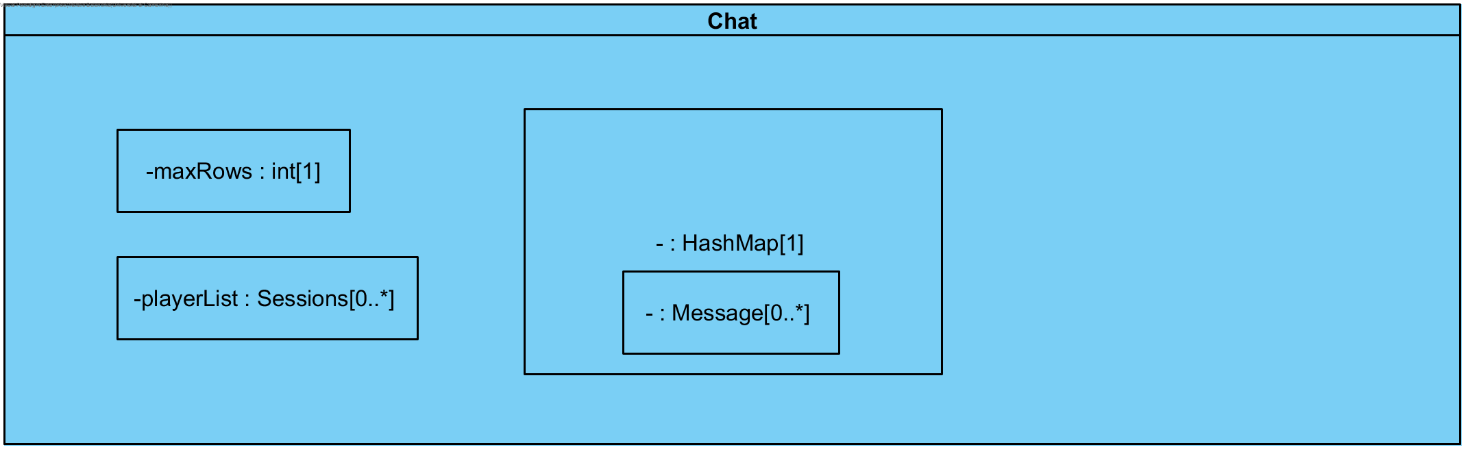


The package diagram shows the relationship between the various packages of the class diagram and the public members of such packages. The package AdminPanel contains a single AdminPanel class which is public and has use dependencies on the subsystems UserManagement and GameManagement, which are not part of our component. Likewise, the Statistics package has only DataViewer class as public member and has the same dependencies on UserManagement and GameManagement. Different is the case for the Chat package. It shows the Chat class as its only public element but it is logically contained in the UserManagement component as shown by the containment relationship.

# Composite structure diagrams

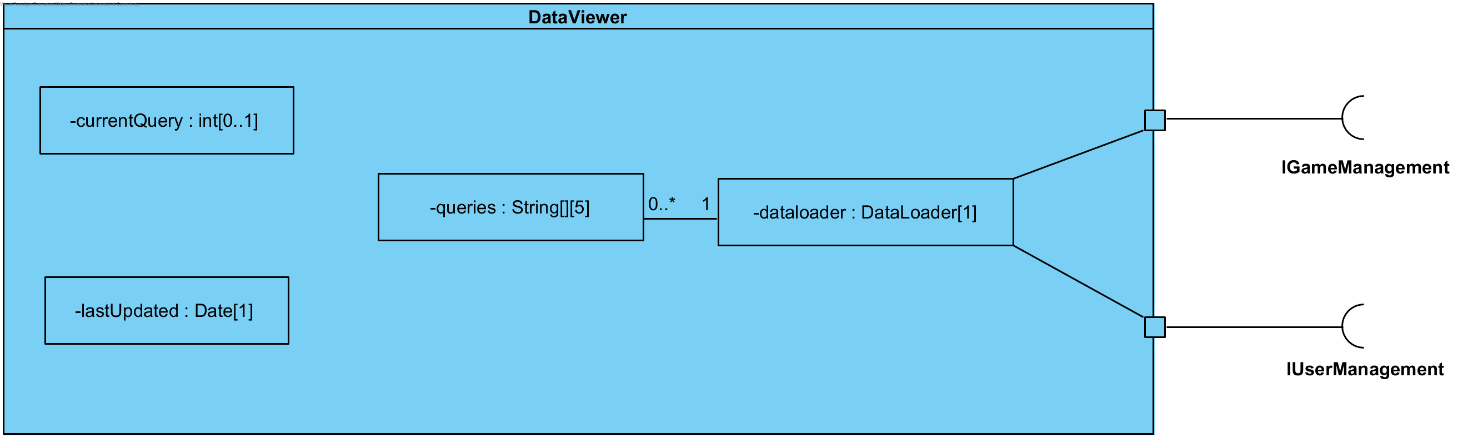
The composite structure diagram show the inner workings of the classes of the component. The following diagrams are relative to the Chat class and the DataViewer class.

## Chat Class



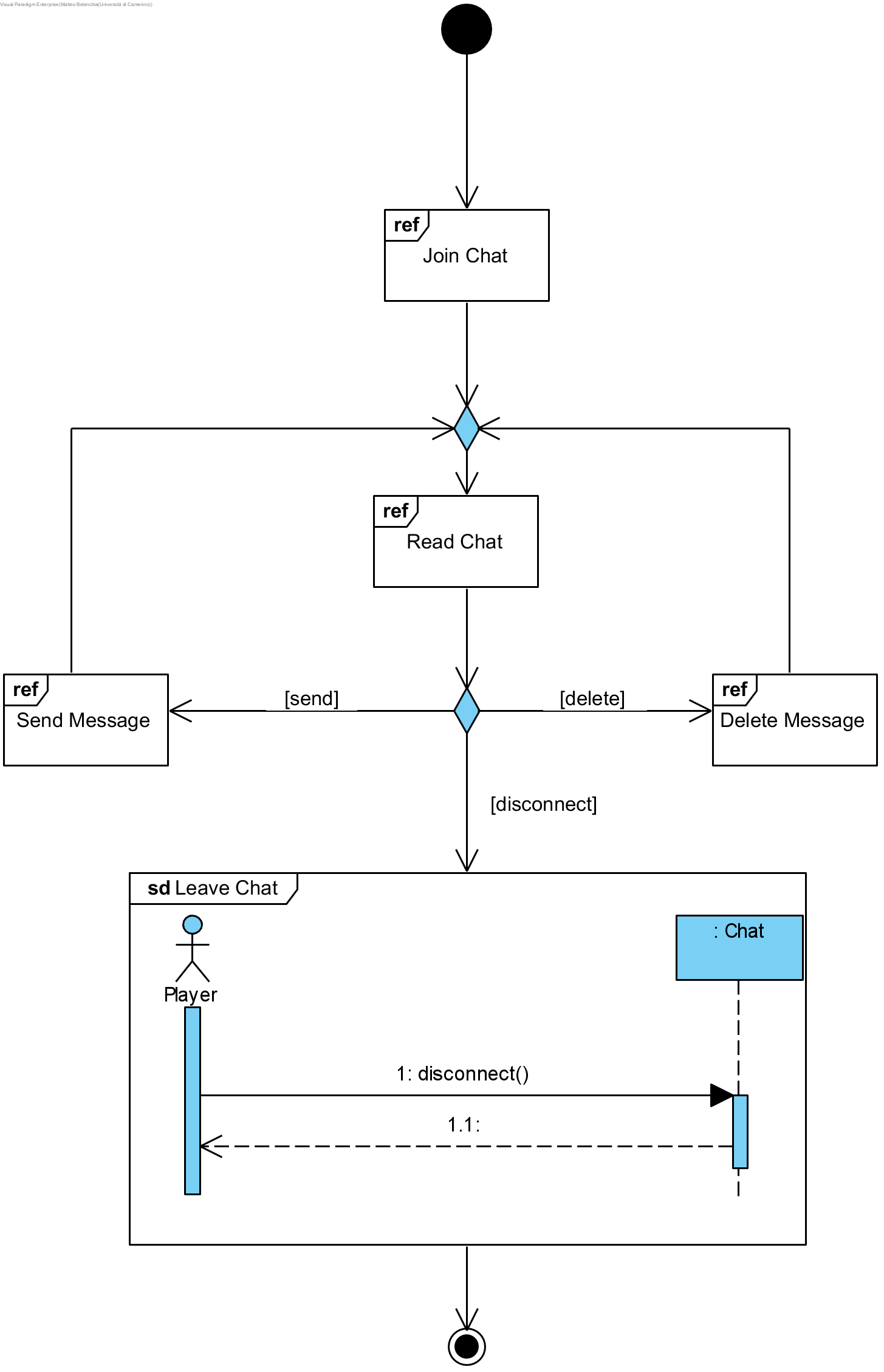
The Chat class contains the attribute maxRows with a multiplicity of 1, and the attribute playerList which can contain 0 to many sessions. Additionally it contains an HashMap object with molteplicity 1 which in turn contains 0 to many Message objects. The Chat doesn’t have foreign dependencies.

## DataViewer Class



The DataViewer class contains a currentQuery attribute with molteplicity 1 but which can be null at the beginning, and a lastUpdated attribute with molteplicity 1. Then it contains a queries attribute which is a collection of 5 String arrays containing SQL queries. These queries are associated with a single DataLoader object with molteplicity 1 which is connected to 2 pins mapping to external interfaces IGameManagement and IUserManagement, provided by the corresponding components. These interfaces are required by the DataLoader object in order to work.

# Interaction Overview Diagram



The interaction overview diagram shows an overview of the operations of the chat. The first activity is a referenced sequence diagram Join Chat that describes the connection to the chat. From there the next activity is described by the Read Chat sequence diagram and from there the flow can split in different ways: if we go for sending a message we reference the Send Message sequence diagram while if we go for deleting messages we reference the Delete Message sequence diagram. From both these activities the control is merged to run again the Read Chat activity in a loop. If, instead, we choose to disconnect, then the Leave Chat sequence diagram describes the disconnection activity and it is reported in full for its semplicity. After that, the interaction overview diagram terminates.

# Conclusions

The component has been developed in all the requested functional and non-functional requirements. The major problems with it is that it has been developed using a rather outdated technology of Java websockets (we would have favoured a Python approach or using the Node.js framework) and that the user interface could have used some better graphics. The integration of the component would also be rather difficult because the Statistics pages require a strict data format to be followed in order to work and lack of coordination with the other groups will prove to be decisive in this issue. We decided to use a very broad data format for messages between client and server making the component much more reliant on following a strict format of data compared to having had multiple types of messages (which would have been prohibitively many for the scope of the component).

# User documentation

This user guide will provide the relevant information in order to use the component.

## Chat

### Functional description

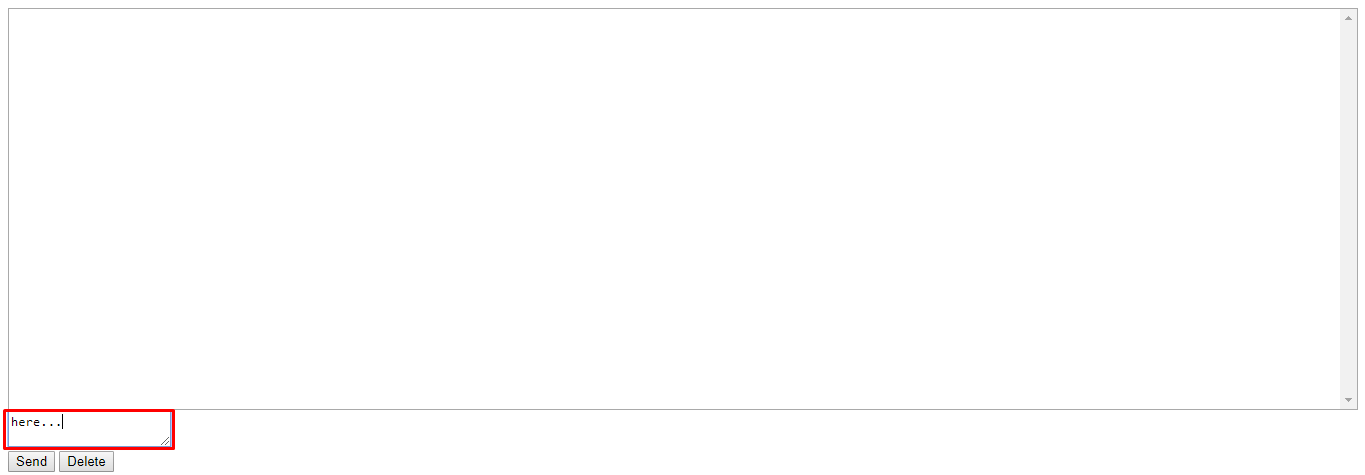
The chat provides the functionality of reading, deleting and writing messages. Admin messages are written in red, and Admins can delete any message while normal Players can only delete their own.

### User manual

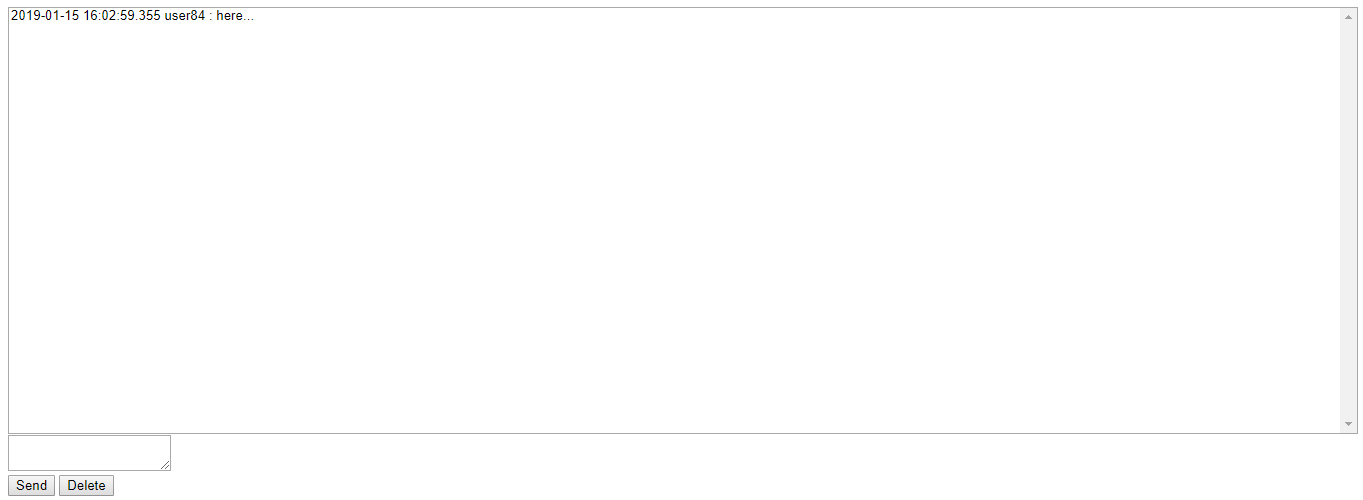
#### Connection and Disconnection

The chat resides on the following webpage: <http://localhost:8080/SE_Chat/chat.html>   
Simply accessing it will connect the Player or Admin to the chat. Disconnection is likewise simple and it’s enough to close the webpage. On connection all messages are downloaded.

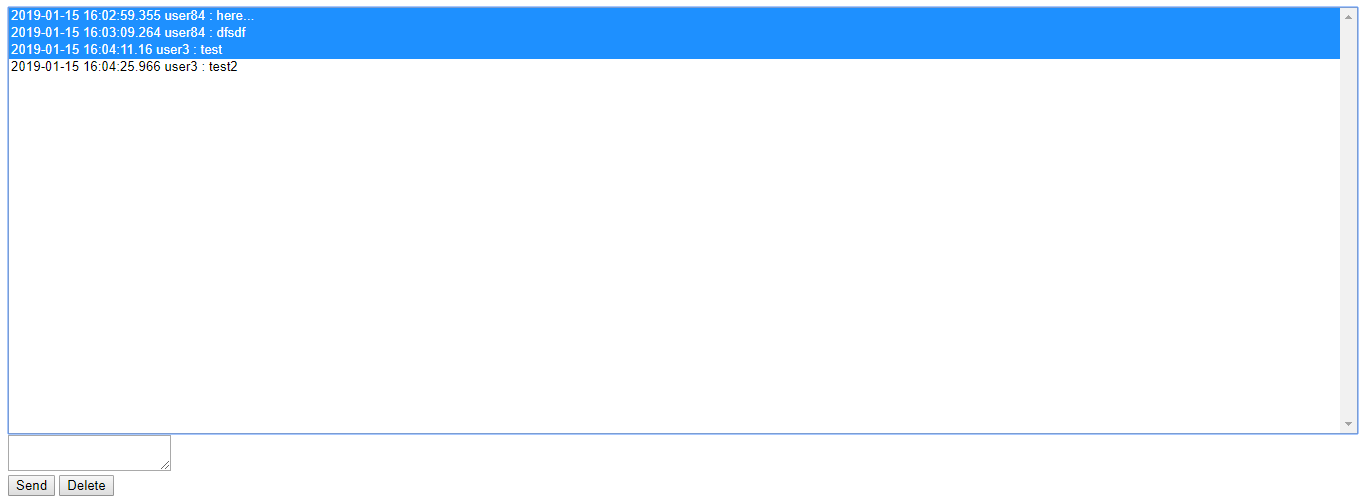
#### Sending a message



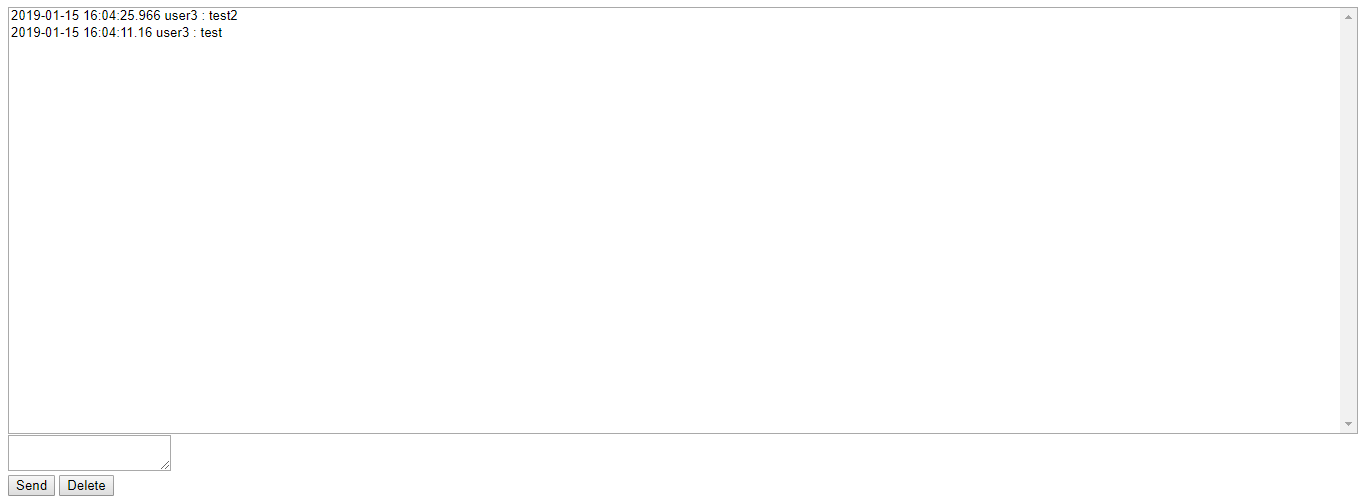
In order to send a message, simply type text in the highlighted textbox and press “Send” button. The message will appear in the chat.



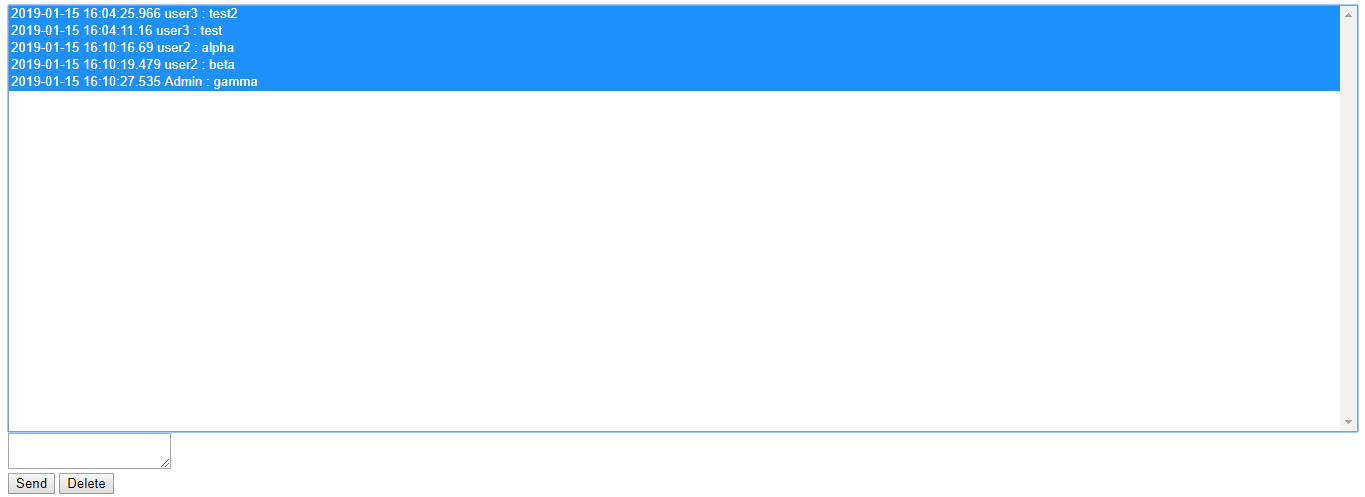
#### Deleting a message

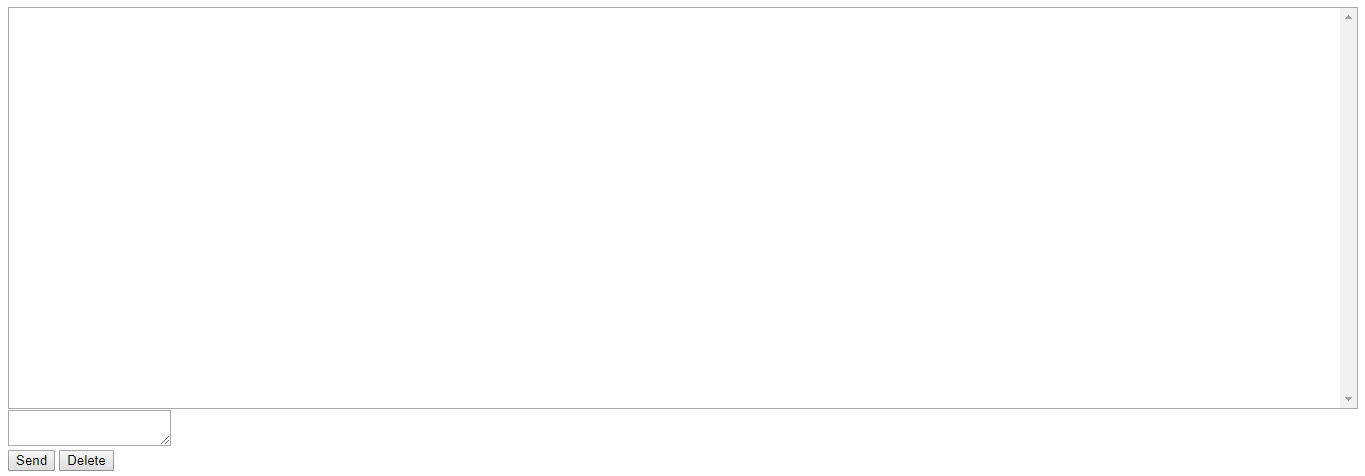


Before deleting a message, select any number of messages in the chat. Selecting a message is a simple as clicking on it, and to deselect a message press CTRL + Click on the message. When finished selecting, just click the “Delete” button. If the client is a Player, like in the picture above (user84), then only its own messages will be deleted.



Otherwise, if the client is an Admin, all of the selected messages will be deleted:





### Complete description

The chat can display messages, receive new messages and delete messages. Messages are received containing only the type of message, the text and the username, and an ID and timestamp is then given on the server and relayed to all clients. Messages are deleted by sending a special type of message containing a type and a list of integers representing the message IDs to delete. When a new client connects or there is a deletion, the whole chat is sent again to the client or clients, in this case the message contains a type and a list of messages.

### Installation description

The chat should be integrated in the User Management component. The installation will need to remove the index.html page which is used to provide a random username and instead provide the Player username stored in the webpage session. It also needs to set the Admin username as “Admin”.

## Admin Panel

### Functional description

The Admin Panel offers functionalities to ban Players, not letting them log in until a certain amount of time expires. It lets the Admin to edit Player properties such as username and email. Admins can set a server capacity, not letting any Player log in until the number of connected Players is below the threshold. Admins can close game sesson, prompting an end to any game. A minimum bet for every game time can be set, forcing Players to bet from that amount upwards.

On the statistics page, we can see statistics for each game type, statistics for each registered Player, daily statistics, total statistics, and an overview of all daily statistics for each day saved in the system.

### User manual

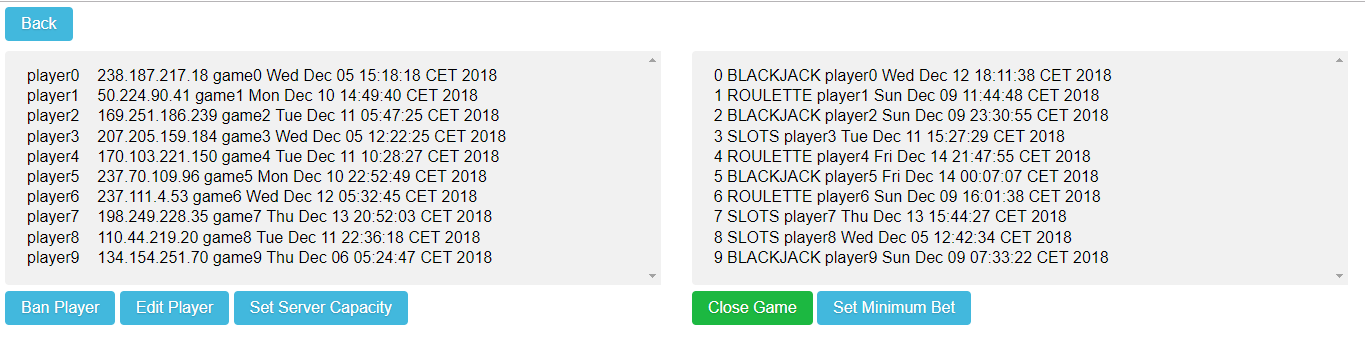
The entry point for the component is the page <http://localhost:8080/Administration/>



On this page, there are 2 buttons to access the relative component: Admin Panel (<http://localhost:8080/Administration/adminPanel/> ) and Statistics (<http://localhost:8080/Administration/statisticsPanel/>).

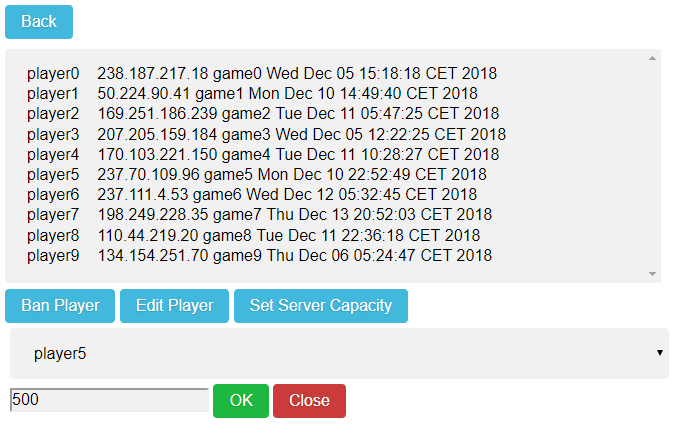
#### Admin Panel page

On the Admin Panel page, we visualize the currently connected players on the left and currently running game sessions on the right. The top-most button “Back” brings you back to the initial page.



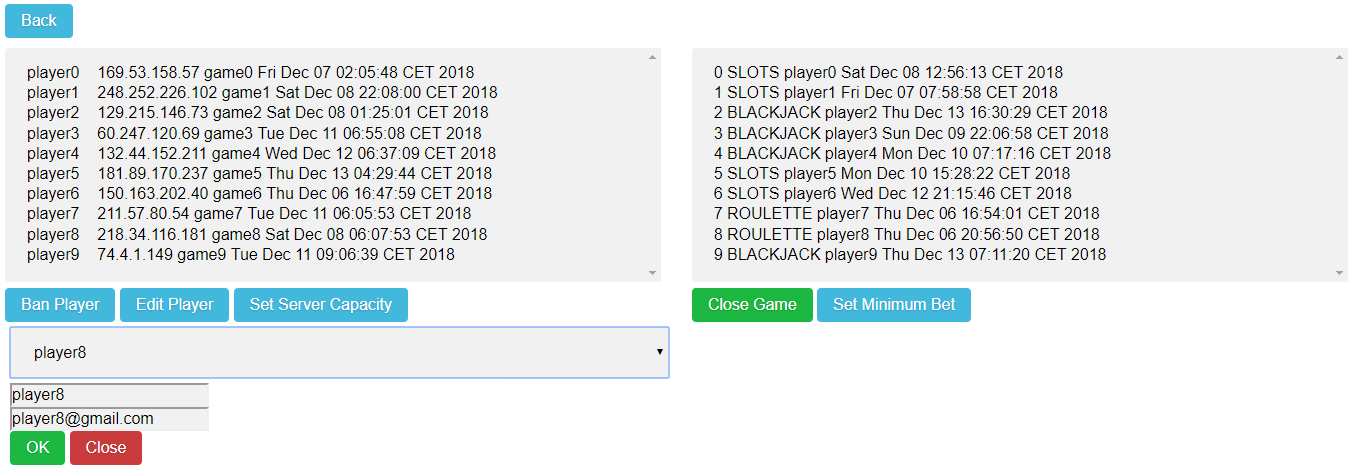
##### Ban Player

On the left, the button Ban Player summons a form in which you can select any player (even if it’s not currently connected) and ban him for the specified number of seconds. The OK button is used to confirm, and close button is used to cancel.



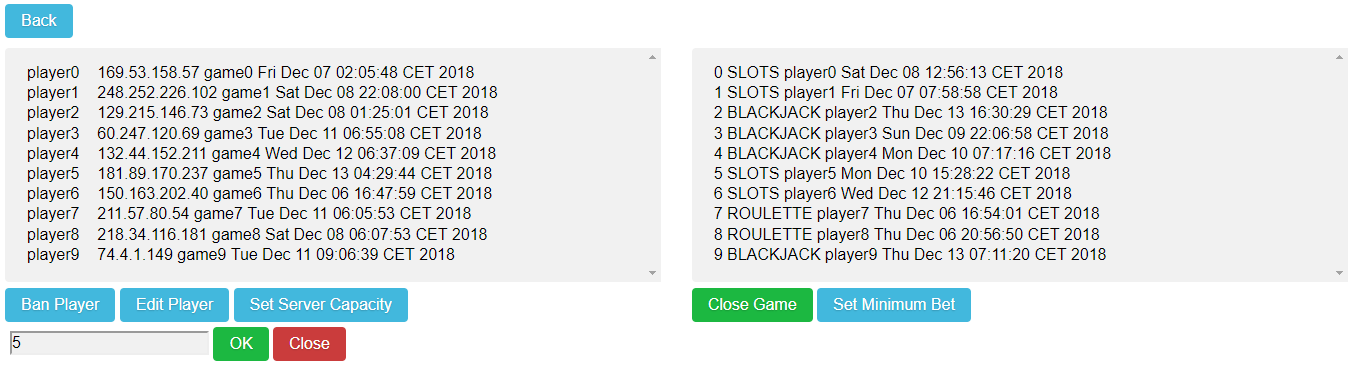
##### Edit Player

The edit player button, once clicked, shows a form to edit the selected player data. The player is selected by using the select form, which shows all the existing players in the system. You can edit the data then press OK to confirm, or Close to cancel. The username and the email must be unique, if another player shares either of them, the changes are discarded.



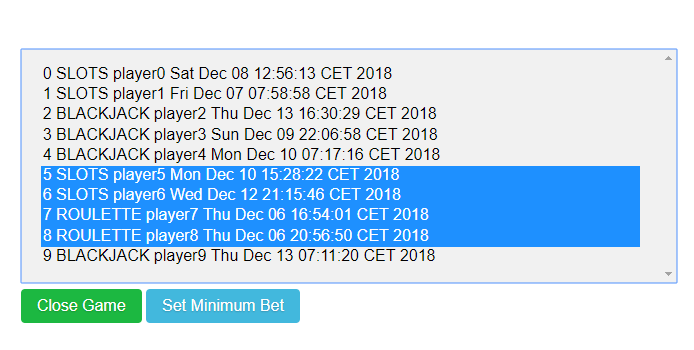
##### Set Server Capacity

The Set Server Capacity button lets you enter a number which will be the maximum number of concurrently connected players allowed. Players are not kicked if the number is lower than the number of currently connected users though. In fact, no one will be allowed to log in until the number of connected players is lower than the number being set. Press OK to confirm or Close to cancel.



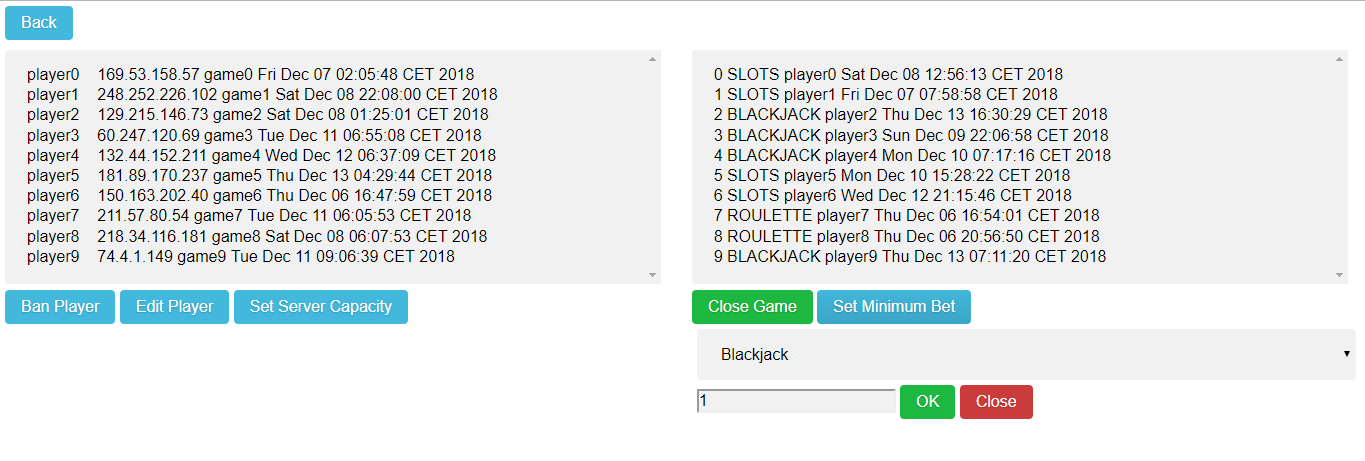
##### Close game

On the right, the Close Game button closes the games that are selected from the above form. You can close more than one game at the same time, by selecting multiple games by dragging or holding CTRL. CTRL + Click on a game to deselect.



##### Set Minimum Bet

The Set Minimum Bet button lets you set the minimum bet for each game. Press OK to confirm or close to Cancel.

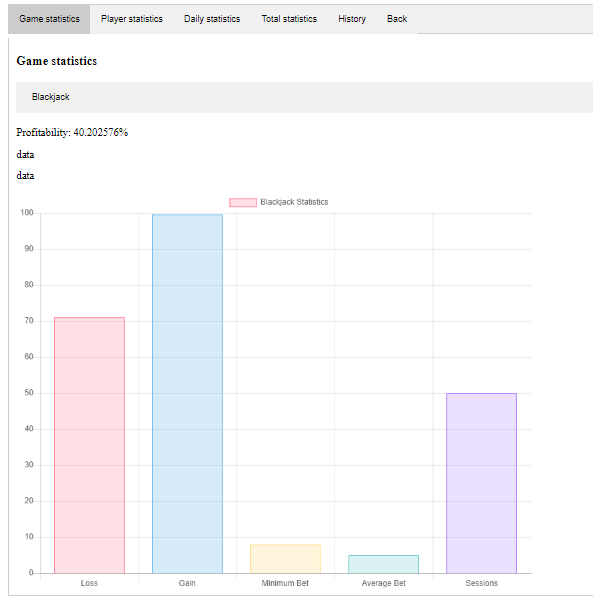


#### Statistics Page

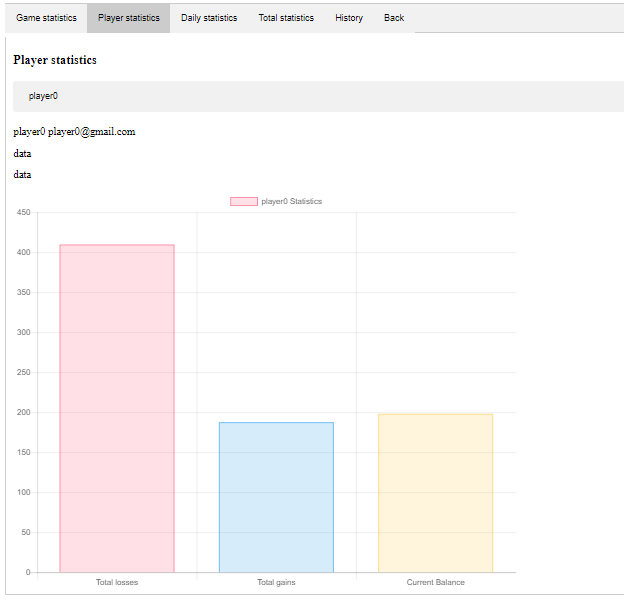
On the Statistics page, there are 5 different tabs to show different statistics, plus a final Back tab that brings you back to the initial page.



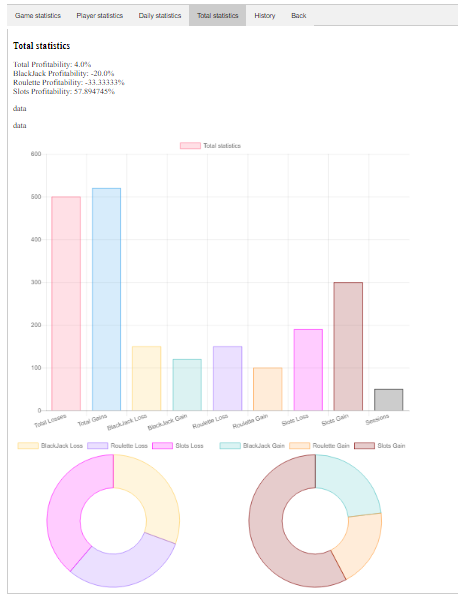
Clicking on the Game Statistics tab opens a view which shows statistics for each game type, selectable with the select form.



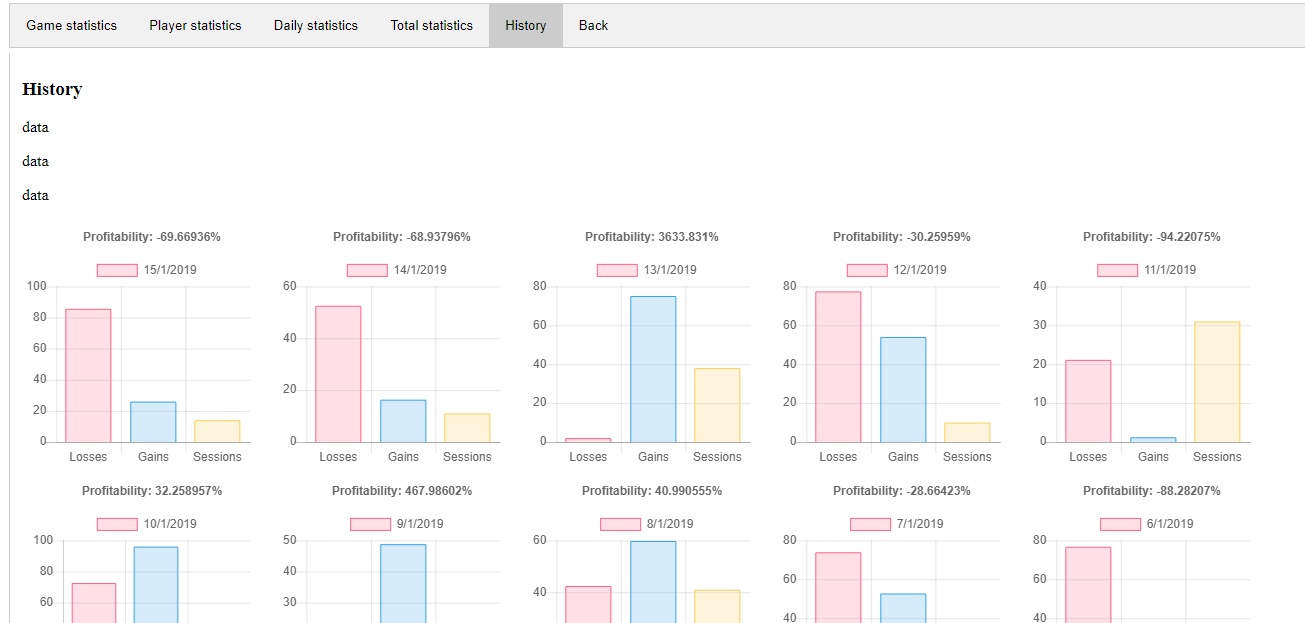
Clicking on the Player Statistics tab opens a view which shows statistics for each player in the system, selectable with the select form.



Clicking on Daily statistics or Total statistics shows either the data for today or the data for all the values present in the application database.



Lastly, the History tab shows an overview day by day of all the days the application has been running in the last month.



### Complete description

The component uses a very broad data format. Messages sent to the server are decoded with having a type and then all the data contained in a List of String arrays, which we resoluted to be the most flexible data container. Messages from server to client are instead JSON arrays with a type in first position and a variable number of data positions. For this reason the data format must be respected very precisely for it to work.

The AdminPanel page can ban players, close game sessions, edit players, set the minimum bet and set the maximum number of connected players. It also shows live the players connected and the games running.

The statistics page shows the Player statistics, with the anagraphic data, the total gains and total losses. The game statistics show for each game gains and losses for the house, minimum and average bet, the total number of sessions and the profitability. Daily and total statistics pages show total losses, total gains, loss and gains for each game type and number of sessions. Additionally donut charts show the distribution of loss and gains per game type. Also total and per game profitability is shown. The difference between daily and total statistics is that the former is about the current day while the latter queries all the data in the database. The history page shows a basic gains, losses and number of session chart for everyday present in the database, along with a profitability value.

Profitability is calculated as

and represents how much was gained or lost after repaying the losses.

### Installation description

In order to be installed it is necessary to connect the actual implemented interfaces in place of UserManagementPuppet and GameManagementPuppet. Furthermore, return values of statistics method must match exactly those of the puppet classes because of the message formatting issues described in the previous section.