Figure 1

Table of Contents

1. Introduction

This is the full documentation of the first release of project RBSG. This project is about building a game, which works and feels like advance wars seen in figure 1. Advance wars is a round based strategy game.
The students got split up into eight different teams consisting mostly of eight members. These eight members got also divided into six developers one scrum master and one product owner. This constellation will swap each release and will force every student to take on the developer role and also either the product owner role or the scrum master role. This should teach everyone in the team about agile programming, working in a team and working all by himself figuring out stuff and reading into subjects which had not been targeted before.

2.1 First Ideas

The first ideas of the team were pretty straight forward and simple. Everyone agreed with having a more clean and tidied up design. Also the team had the same opinion about the structure of the project, which means they all wanted the release to be separated into two categories. Front end and backend.

2.1 Mockups

These screens are the first rare realization of the team's first ideas.

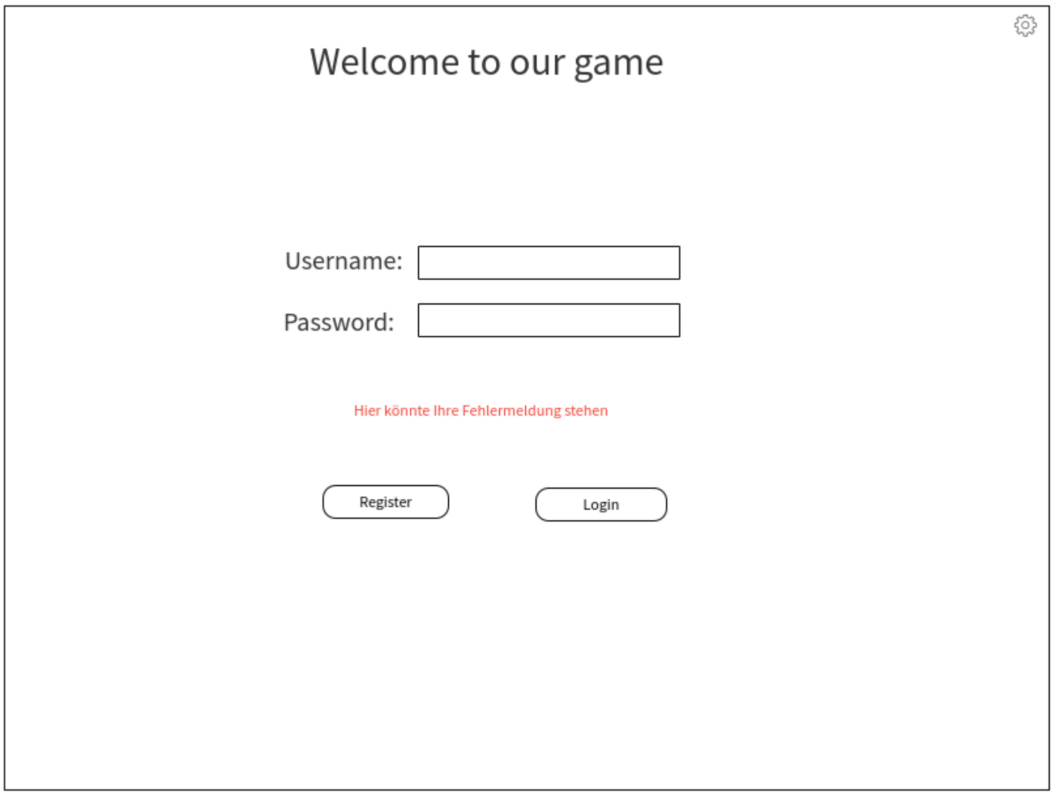


Figure 2.1

2.1.1 Welcome Screen

In the „Welcome Screen", the team thought that the screen does not need a lot of features, which distracts from something that is actually pretty simple. So they created this simplistic login mockup screen that basically only functions as login and register that is it. There are two ideas that the team thought would look good in the screen, which was one, the settings button in the top right, enabling more options to the user, like dark mode and log out and such and two, let error messages fade in and fade out instead of opening an annoying window for the user to close.

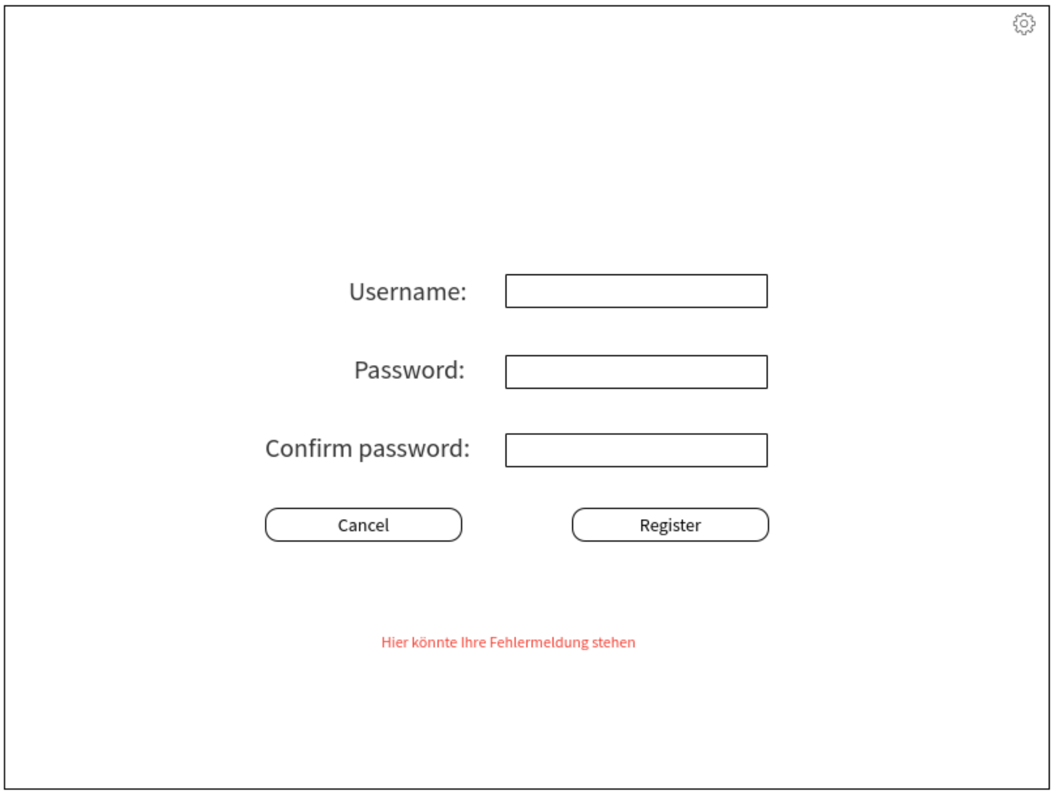
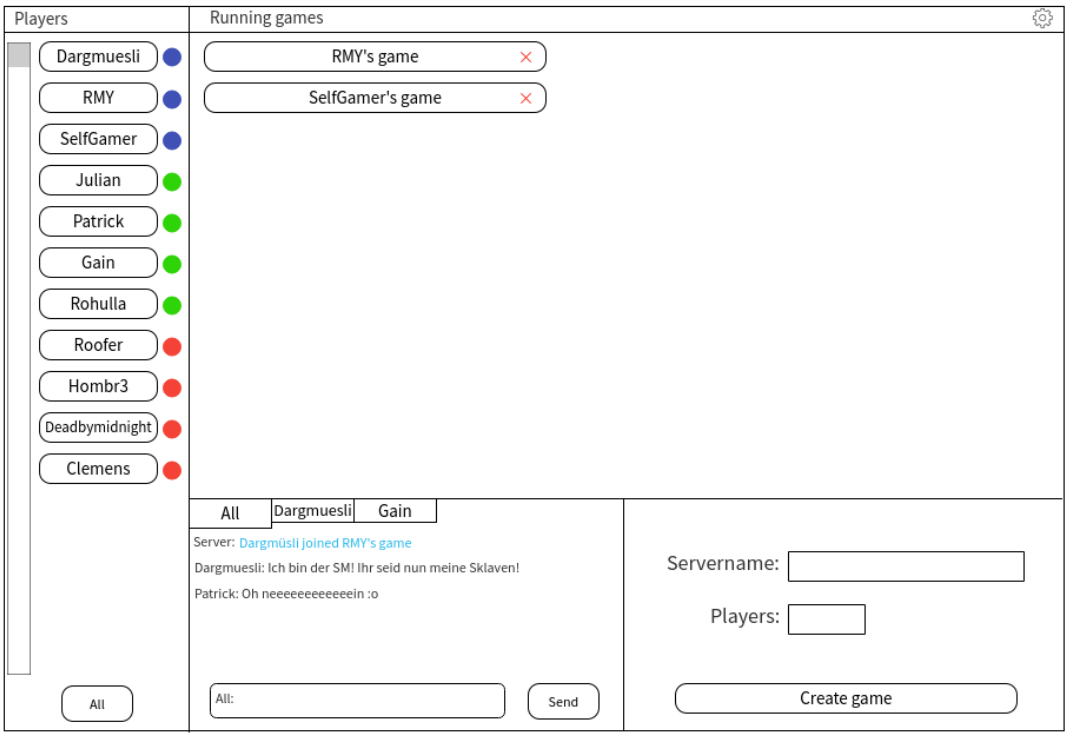


Figure 2.2

2.1.2 Register Screen

With the same mindset the product owner created the „Register Screen", still trying to make it as simplistic as possible because it is just a register screen and should not distract and stop the user from registering. Again they thought it would be a good idea to put a settings button on the top right to enable even more options instead of only a button which gives the user the option to log out. Also they wanted to make the error messages the same way, fades instead of pop up windows.



Server: Dargmüsli joined RMY's game
Dargmuesli: Ich bin der SM! Ihr seid nun meine Sklaven!
Patrick: Oh neeeeeeeeeeeein :o

Figure 2.3

2.1.3 Main Screen

In the „Main Screen“the team had a little more in mind, instead of just making it simplistic. Of course they had the same mentality creating this main screen mockup, keep it simple and clean as possible to make it look modern. First things first, they wanted to set the focus of the main screen to be the running games, because that is what this screen is all about, give the player the option to join games, view them, create them and delete them. So this is a pretty big part of this screen. They also made every game that is listed clickable in the mockup so that the user can simply click on a game and join. Of course the game has to have free player slots which are not displayed on the mockup but were meant to be part of the screen.
Now getting to the second part of this screen, which is the chat. The team wanted the logged on players, which are shown on the left of figure 2.3 to be buttons that are targetable as chat targets. If the user clicks on a player, a window should pop up enabling the option to private chat, but also giving the option to whisper in all chats, so that you can overview everything that happens while chatting privately with your chat target. Furthermore they wanted the server messages to get a special focus and implemented it to be a different colour and be sent to every chat, so that nobody will miss a server shutdown due to server maintenances for example. The last feature that the team thought about in the chat were those coloured circles. They wanted to make the user see, if players are in-game (blue), in the lobby (green) and occupied (red). As mentioned before they wanted to make everything to look clean and simple, so they did not want to implement more and make it look overloaded with features and such. Lastly the settings button got its appearance on this screen as well.

2.2 Domain Stories

Domain stories are visualisations about use cases that the user encounters using the client of team B. Theses use cases are shown in figure 3.1 - 3.11.

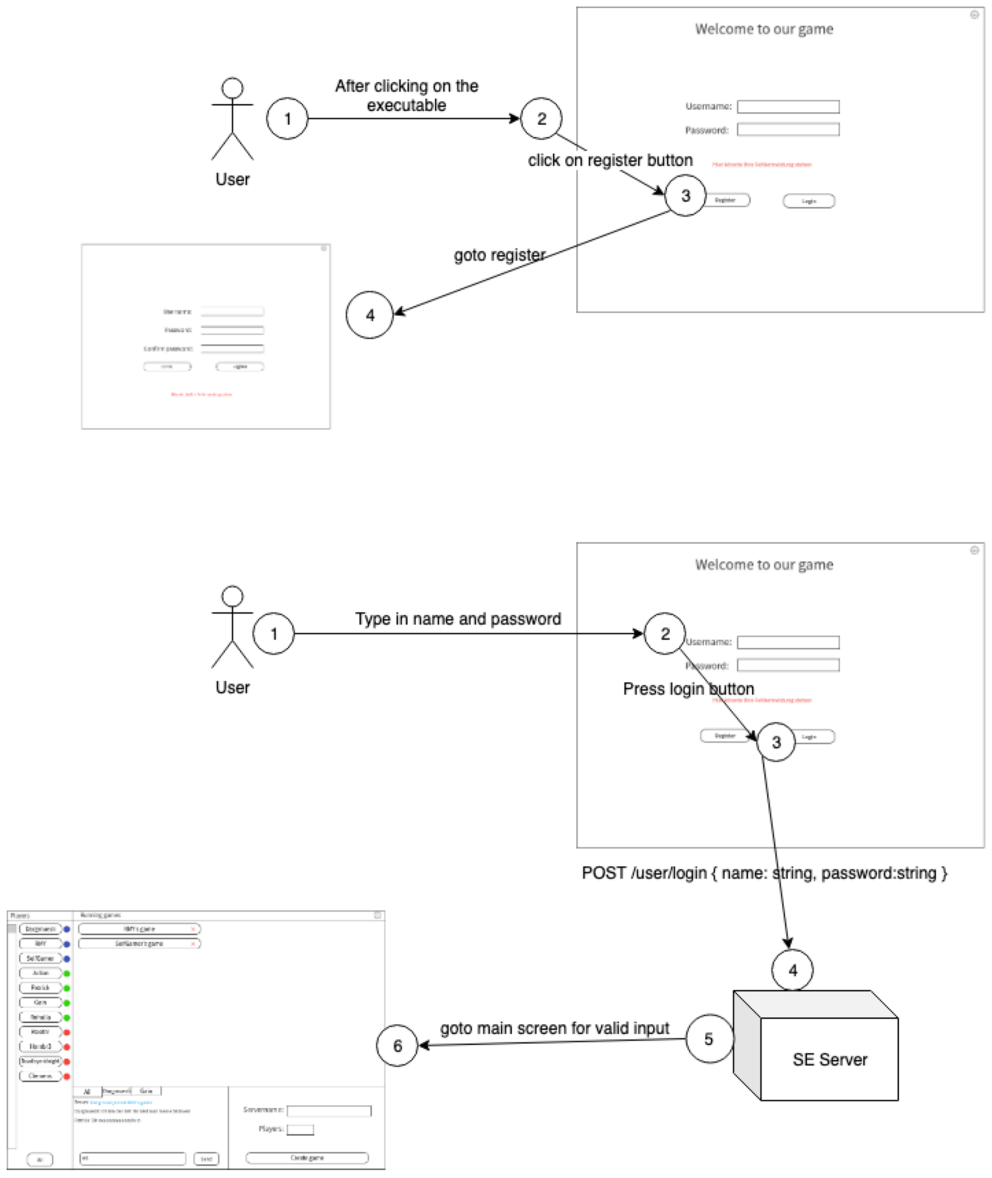


Figure 3.1

First panel showing the register process. (1) After you click on the executable jar, the client will open up and the login screen will show up. (2) Click on the register button to be directed to the register screen (4). Second panel shows the process of an already registered user typing in his login data. (1) Type in your login name and password. (2) Press the login button to be directed to the main screen (6). (3) After you pressed the login button you will send out a message to the SE game server as shown in figure 3.1 .

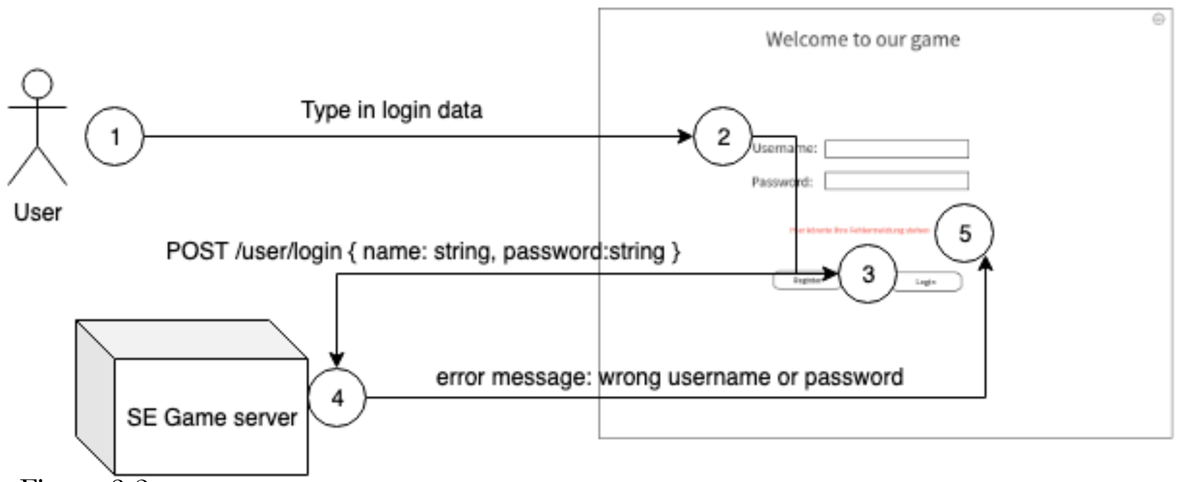

Welcome to our game
User
1
Type in login data
2
Username:
Password:
POST /user/login { name: string, password:string }
3
5
Register
Login
SE Game server
4
error message: wrong username or password

Figure 3.2

(1) Follow the same steps seen in figure 3.1 and try to log on with your login data. (4) This time your name or password will have a typo for example or the server is not reachable for some reason. (5) The client will show an error message stating the problem that occurred.

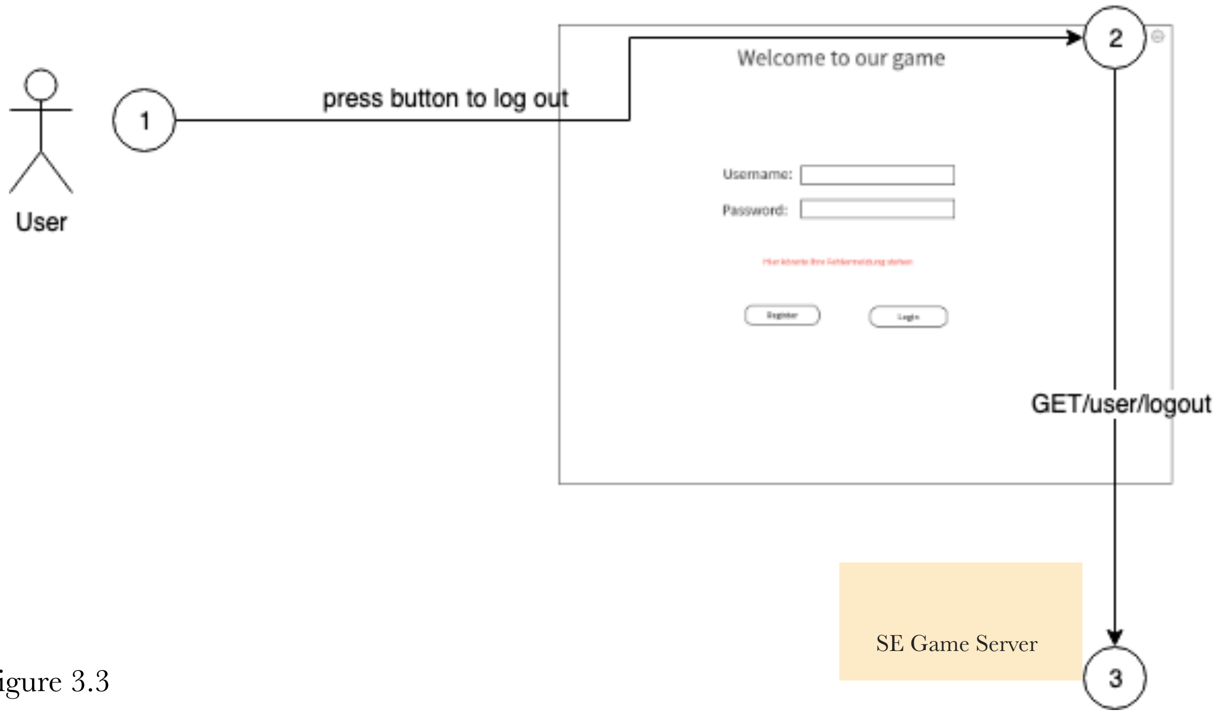


Figure 3.3

(1) Press the button on the top right to log out. (2) A message will be sent to the SE game server and the user will be logged out.

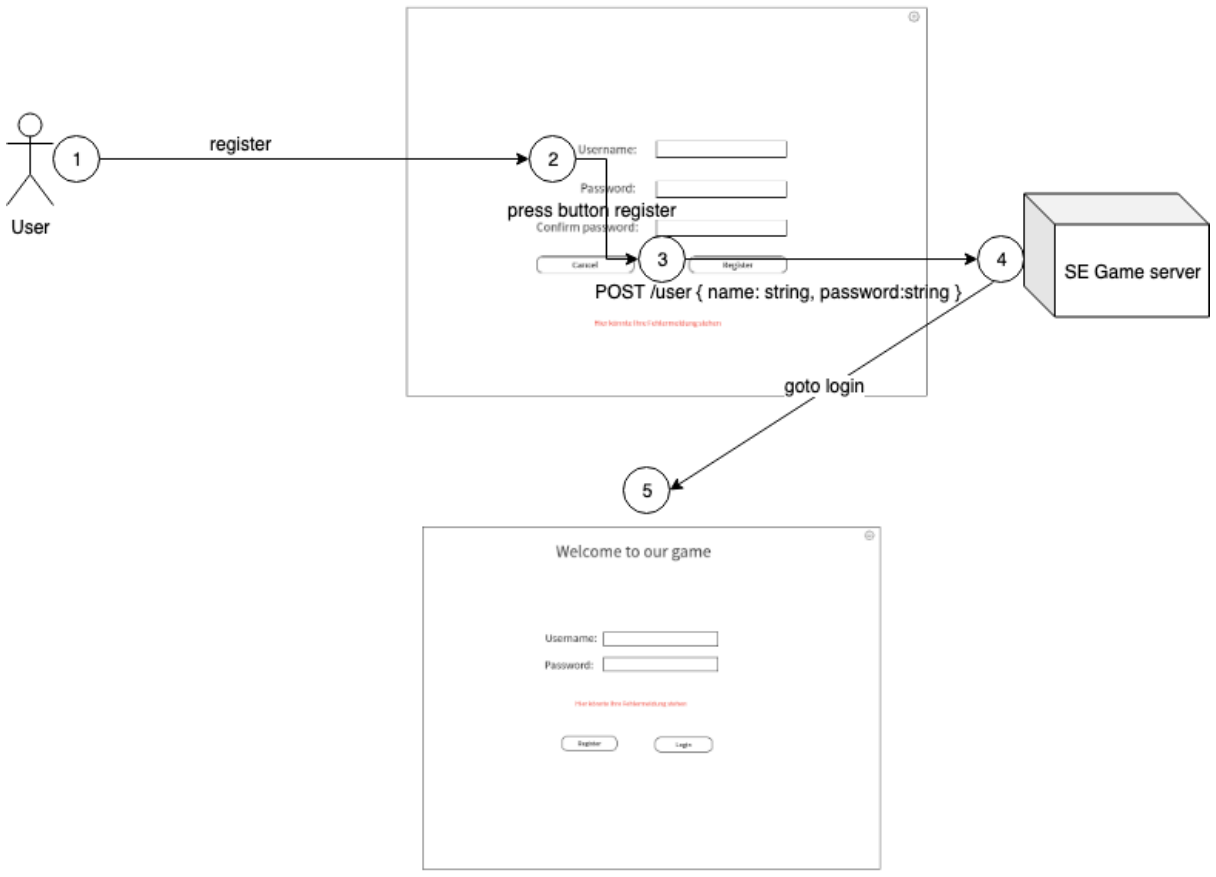


Figure 3.4

(1) In the register screen you can type in your username, password and confirm your password, confirm password is there for protecting the user from accidental typos. (2) After typing in your data, press the register button. (3) The server will get a message and register a user (4). (5) The user will be headed back to the login screen, so that he has the option whether he wants to login directly and use the client or log off after registering.

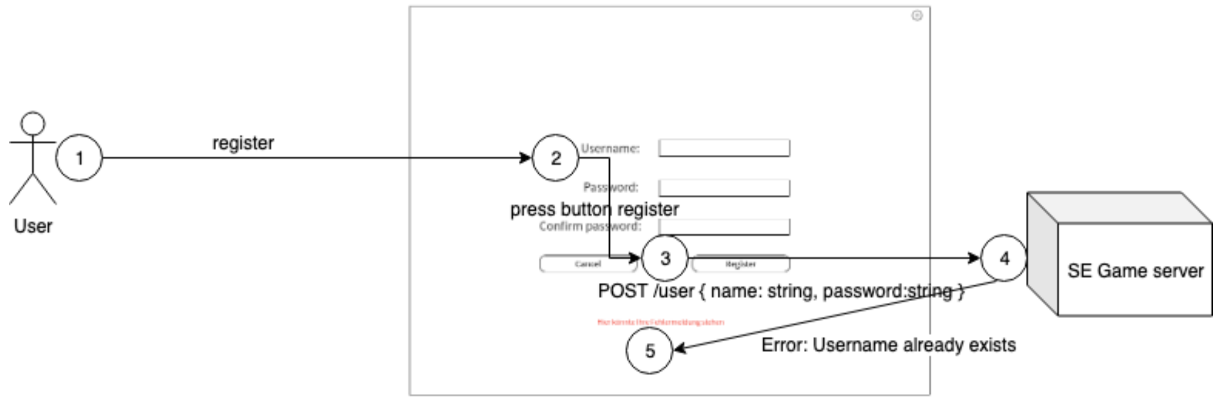


Figure 3.5

(1) Repeat the process seen in figure 3.4 and try to register. (4) This time the server will either tell you something like „that username already exists“ or the server isn't even reachable. (5) An error message will show up and tell the user what problem he is facing.

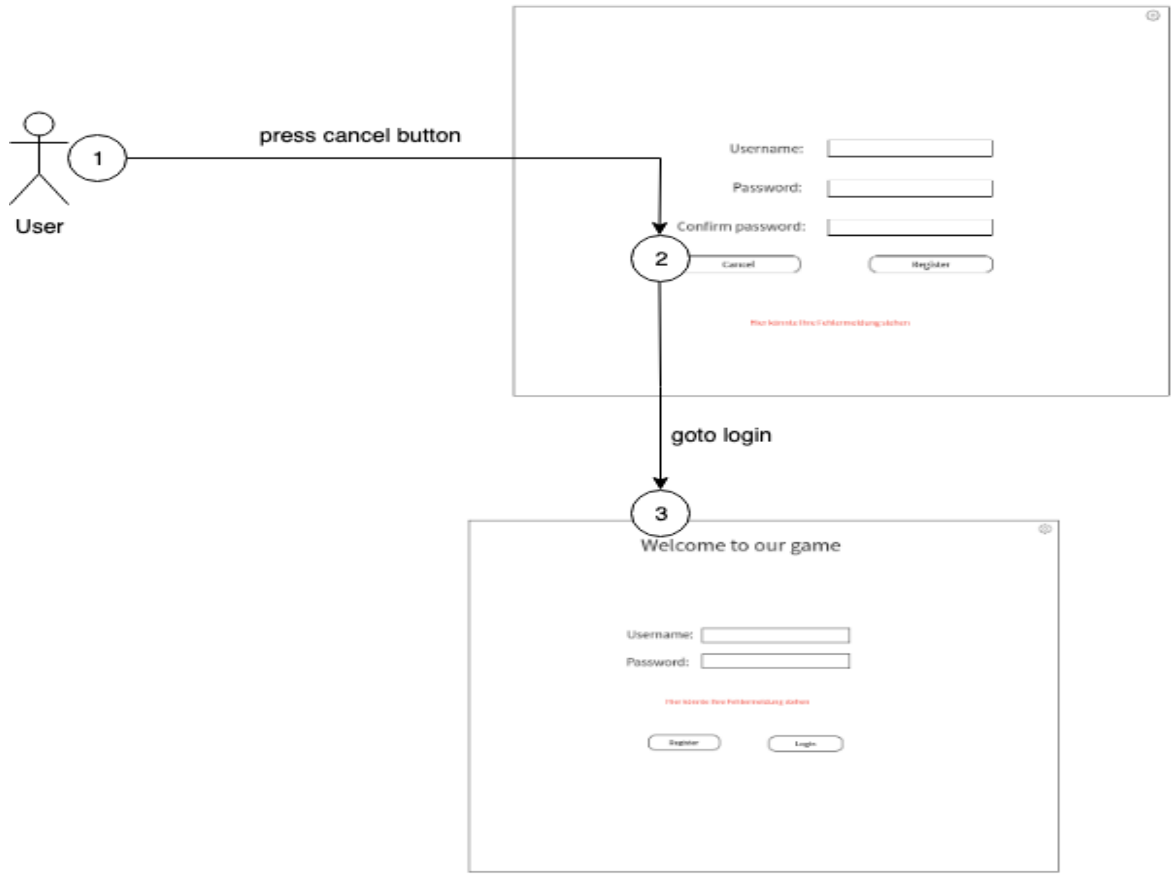
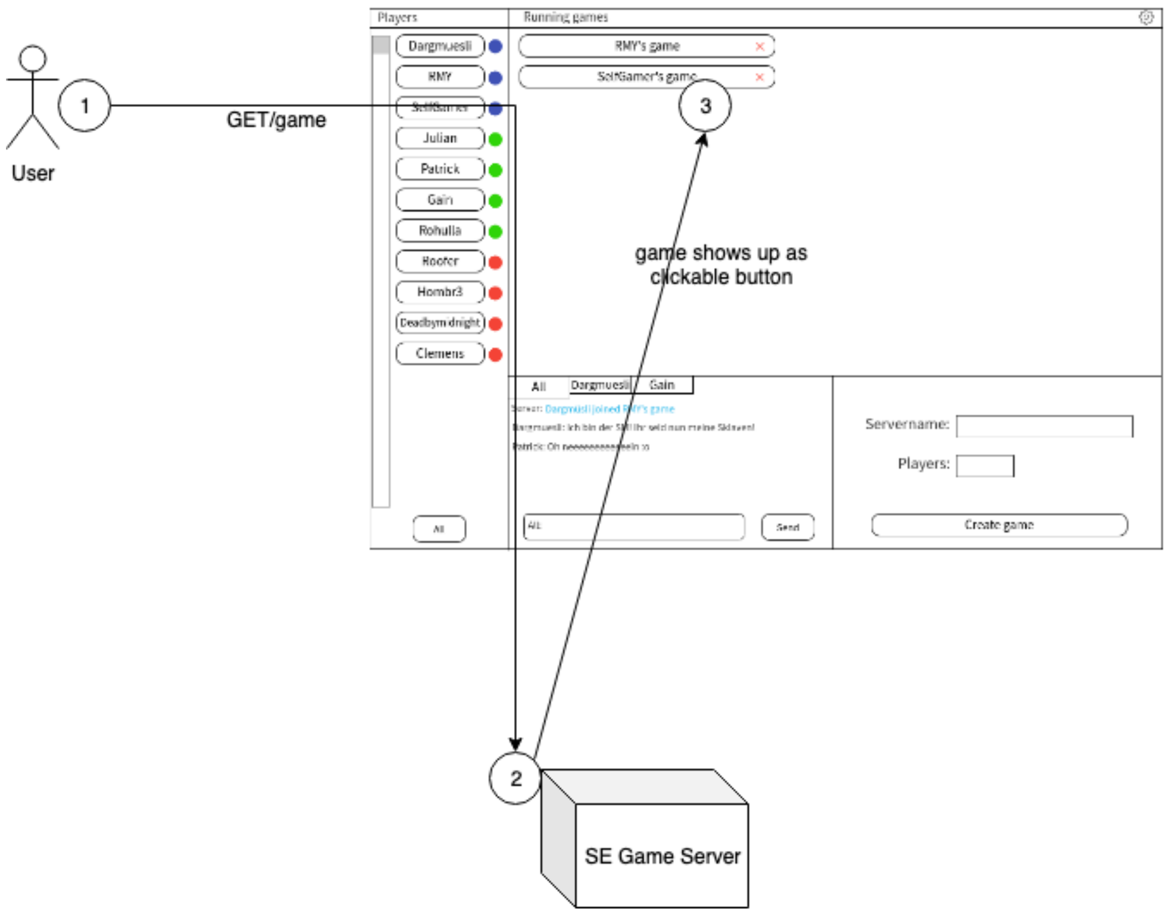


Figure 3.6

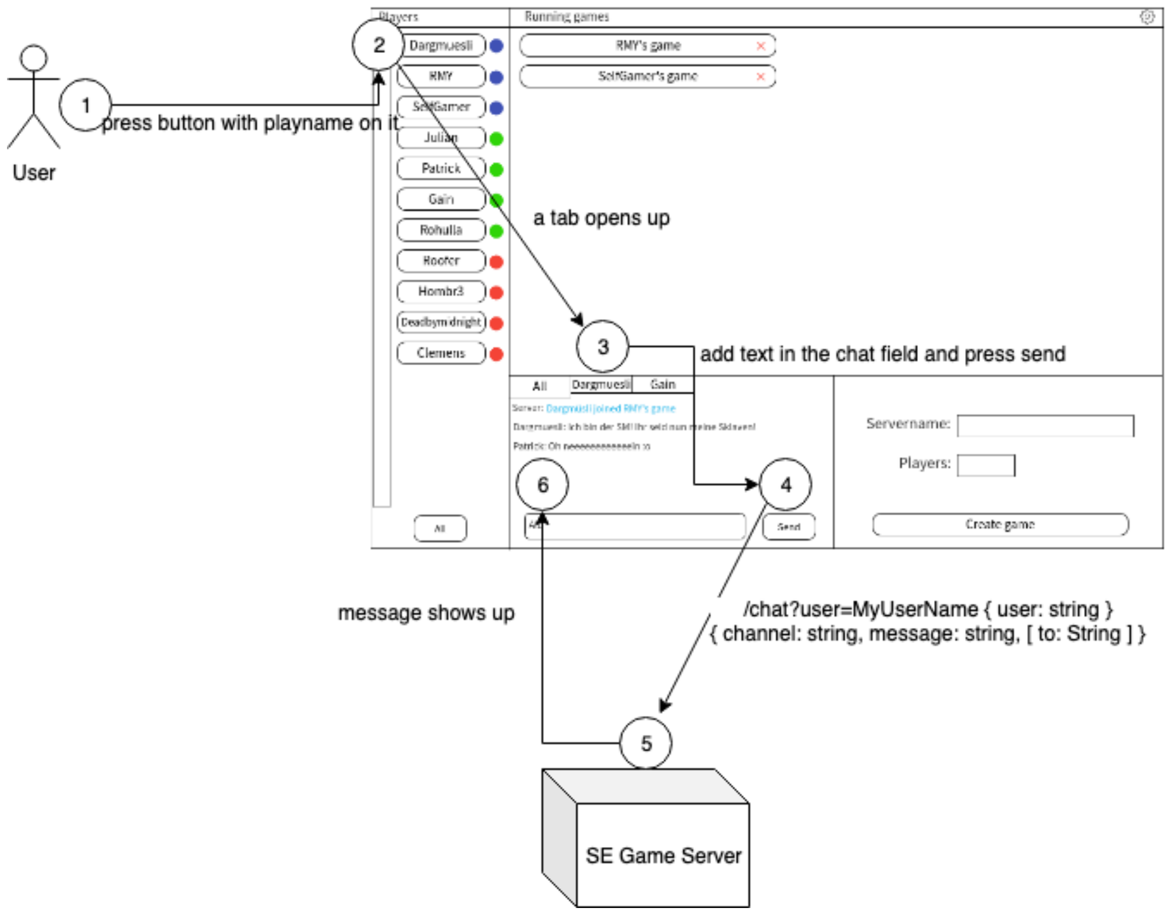
(1) Press the cancel button if you accidentally pressed the wrong button. (2) After pressing the cancel button the user will be directed back to the login screen (3).



Server: Dargmüsli joined RMY's game
Dargmuesli: Ich bin der SM! Ihr seid nun meine Sklaven!
Patrick: Oh neeeeeeeeeeeein :o

Figure 3.7

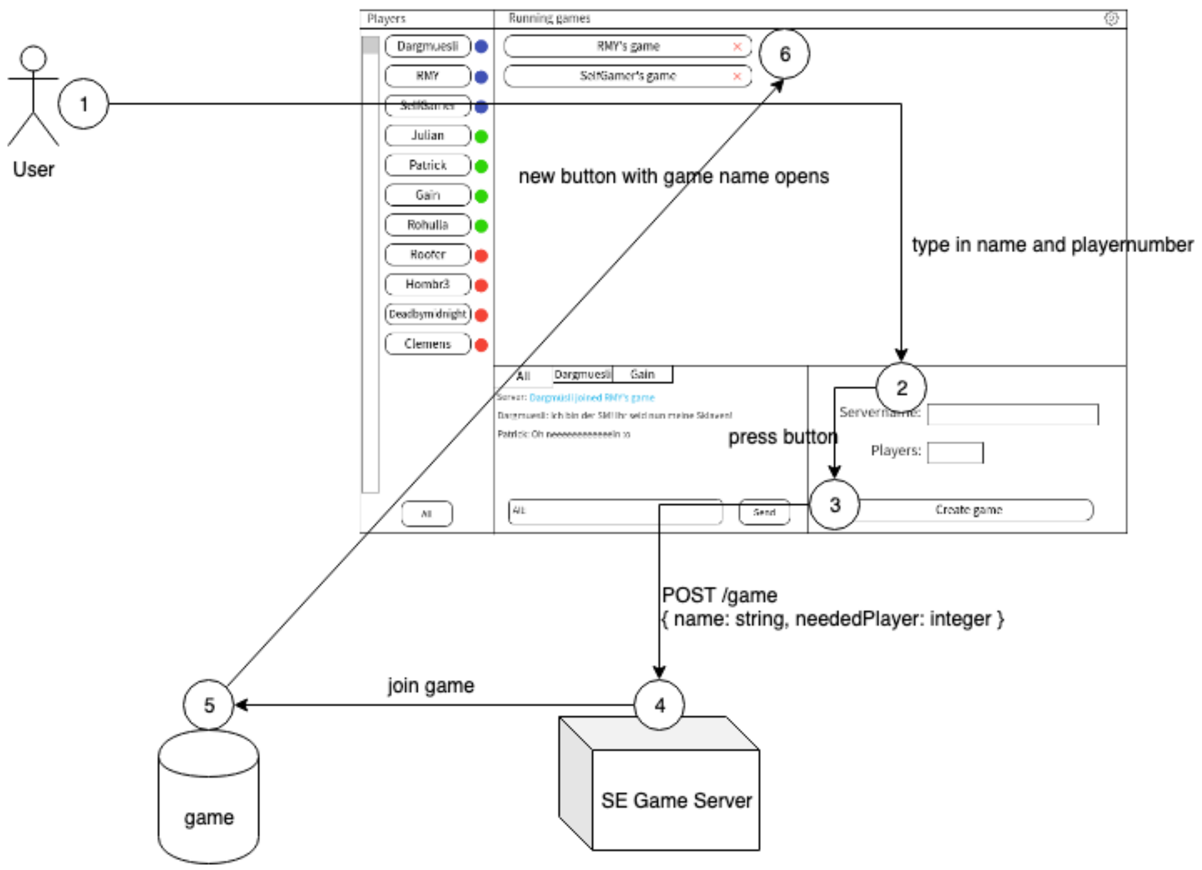
(1) The client of team B will send out a message, that gives the client information about the games, which are live on the server. (2) Active games will be shown as clickable buttons to give the player the option to join any game that has been listed.



Server: Dargmuesli joined RMY's game
Dargmuesli: Ich bin der SM! Ihr seid nun meine Sklaven!
Patrick: Oh neeeeeeeeeeeein :o

Figure 3.8

(1) The user can press a button with a name of an online user, to chat and interact with him. That also counts for the „All“ button which works the same way, with the only change that it can not be closed. (2) After clicking the button a new tab will open up if there is no existing one yet. (3) The user can now type in his message and press the „Send“ button. (4) A message will be sent to the server containing information about both users and strings about the message and channel. (5) The server will process the message and will direct the message back to the main screen, which then shows the message in the chat field (6).



Server: Dargmuesli joined RMY's game
Dargmuesli: Ich bin der SMI Ihr seid nun meine Sklaven!
Patrick: Oh neeeeeeeeeeeein :o

Figure 3.9

(1) The user can type in a „Server-name“ and a player number to determine how many players are allowed in your game and what the name of your game should be. (2) Press the button „Create Game“ (3). (4) A message to the server will be sent which contains information about the just entered strings. (5) The server will then open up an instance of the game and let the user join. (6) The game also gets listed with the associated name and player number, so other online users can click on the game.

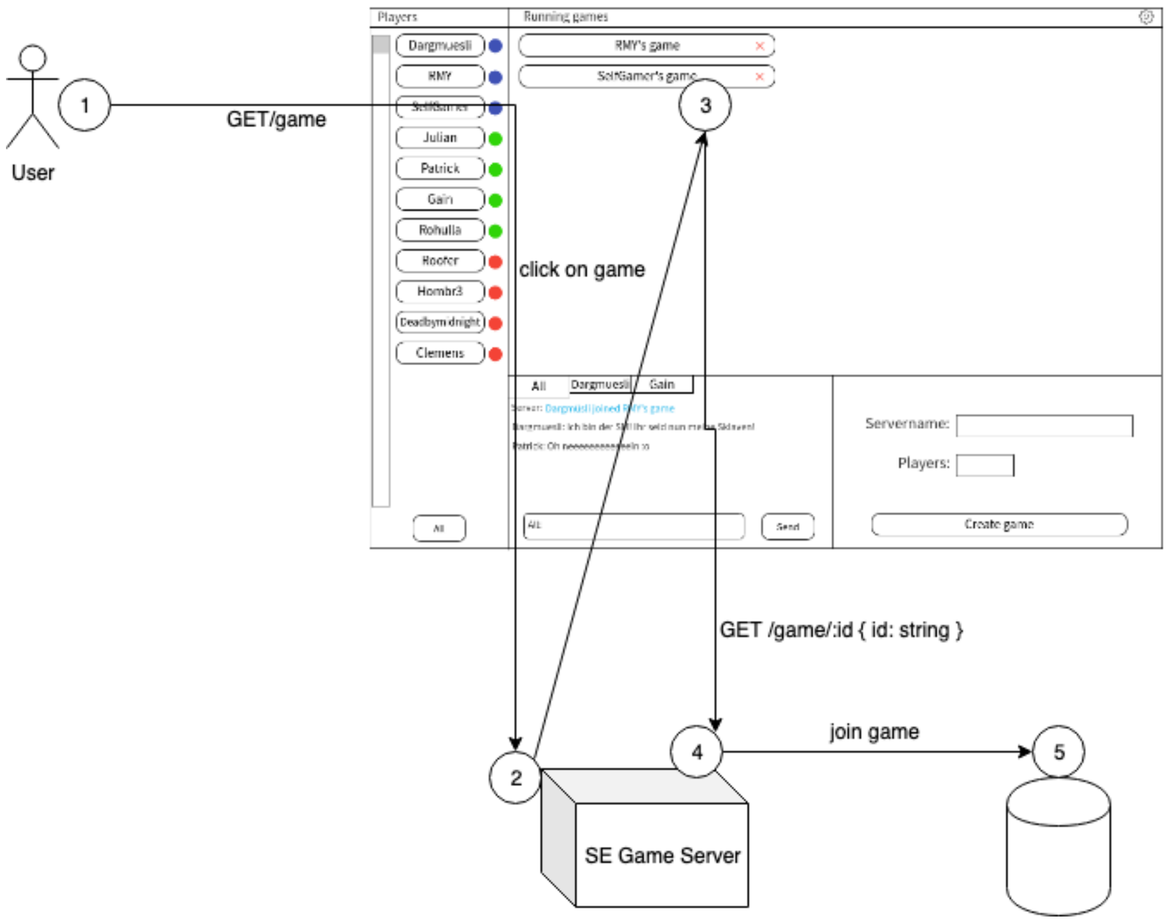


Figure 3.10

(1) After getting information about the active games the player can click on a game via a button (3). (4) The server will get the information about the game that has been chosen and clicked on. (5) The user will join the server and will be able to play the version of team B's advance wars.

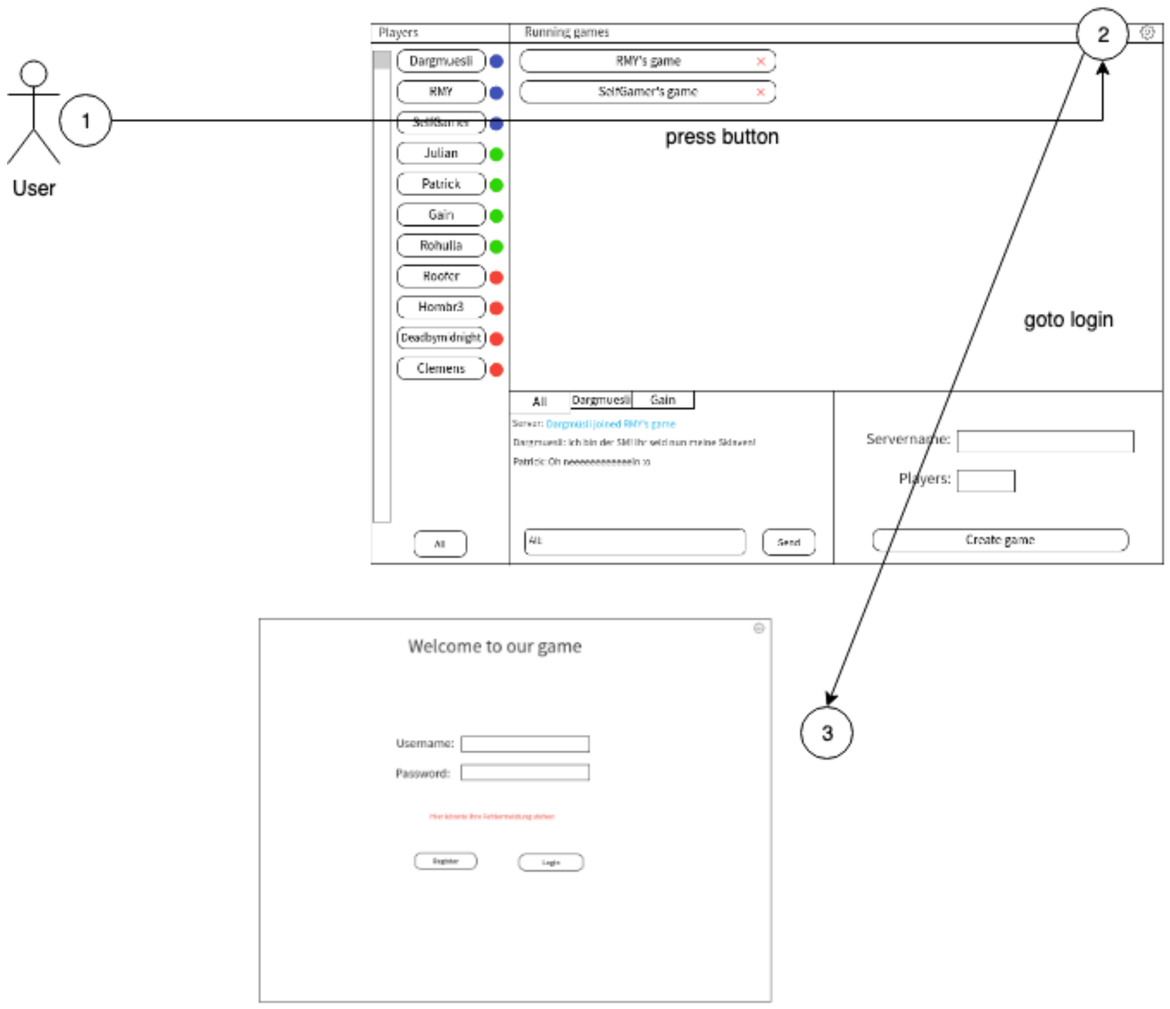


Figure 3.11

(1) In the main screen the user can click on a button in the top right to log out (2). (3) This will direct the user to the login screen.

After the most important domain stories are shown above in figure 3.1 - 3.11, the rest is written in the text below. Divided into four epics.

Design

This epic was about the design choices the team made and how they should have been implemented.

Basic FXML
Screen layout files for the welcome, register and main screen need to exist, so that GUI elements can be added. All scale properly with different window dimensions.

Darkmode
Create application stylesheets that theme a dark background with light text and input elements. A harmonic UX is focused.

Fades
Smooth transitions between the screens are to be created. They also should scale well with different window dimensions.

Error Handling
Error messages returned by the web client need to be visually represented to the user. A class needs to be implemented that can receive error messages as strings and to which, as a screen, can subscribe to display them.

Persistence

Persistence was about saving data for the user and about privacy ideas that the team had.

JSON De-&Serialisation
A class which can be called to load and save data to files needs to be created. Individual keys should be targetable.

En-&Decrypting Values
As a special feature, e.g. a custom server could be utilised to act as a secret provider or processing unit to be able to save and load encrypted (private) messages.

YAML Game Saves
For use in a future release the current state of the game should be saveable using fulibYaml.

Web client

In this epic everything was about connecting and communicating with the server in the right way, so that the user can login, register, chat and create games to his wishes.

HTTP Manager
A class with functions for GET, POST and DELETE requests need to be written. Their parameters should be „URI uri“, „Header[] headers“ and „HttpEntity body“ and they should return a „String responseBody“.

Response Body
A function getResponseBody(HttpResponse httpResponse) needs to be written, which deals with the response itself, its status code and which throws error messages.

REST Endpoint
For every API endpoint a separate request class needs to be written, with each one featuring an inner builder class. They will be created by the main API class and invoked by the http manager. An interface should define the overall layout.

Socket Endpoint
A class that acts as a web socket client needs to be implemented to which message handlers, like the chat class, can be subscribed.

API
An RbsgApi class needs to be written that contains getter functions for each endpoint's builder class object with required parameters as parameters.

Chat
A class that represents every feature of the chat, like history and scopes, needs to be written.

2.3 User Stories

These user stories were used to give the developer just enough information to produce a reasonable estimate of the effort to implement it.

1.1 Go to Register

As a user on the welcome screen I want to be able to click a button named „register“ so that I will be directed to the register screen.

1.2 Login

As a user on the welcome screen I want to be able to fill in my username and password into the according text boxes and click a button named „login“ so that I will login to my account and be directed to the main screen.

1.2.1 Remember Login

As a user on the welcome screen I want to be able to select a „remember login“ checkbox so that my login credentials are remembered across application restarts.

2.1 Register

As a user on the register screen I want to be able to fill in my email address, username, password and repeat password into the according text boxes and click on the button named „register“ to register an account and be directed back to the welcome screen.

2.2 Cancel Register

As a user on the register screen I want to be able to click on the button named „cancel“ to get back to the welcome screen without registration.

3.1.1 Create Game

As a user on the main screen I want to be able to fill in a name into the according text box and a required player number into the according integer inout field and click on a button named „create game“ to a named game on the server.

3.1.2 View Games

As a user on the main screen I want to be able to view a list of existing games so that I can join or delete them.

3.1.3 Join Game

As a user on the main screen I want to be able to click on buttons named as existing games to join the corresponding game.

3.1.4 Delete Games
As a user on the main screen I want to be able to click on a button inside each button for an existing game to delete the corresponding game.

3.2.1 View Lobby Users
As a user on the main screen I want to be able to see logged in users to be able to select them as chat targets.

3.3.1 Select Chat Target
As a user on the main screen I want to be able to click on buttons named as existing players or „all“ to set the target of the chat to the corresponding value.

3.3.2 View Chat Target
As a user on the main screen I want to be able to see the current chat target as a chat tab to decide whether it is correct or not.

3.3.3 Send Chat Messages
As a user on the main screen I want to be able to enter a message into a text box inside the chat section and click on the send button to send the message to the server.

3.3.4 Receive Chat Messages
As a user on the main screen I want to be able to receive messages from the system, other users or the current game to read them in the screens's chat section.

3.4 Logout
As a user in the main screen I want to be able to click on „logout“ accessible via a settings gear in the top right to log out of my account.

3.3.5 Chat Encryption
As a user I want to be able to send encrypted chat messages to other users of this application and read decrypted messages to me by default to ensure my data's privacy.
This is a special feature. An option to show the own and remote public key should exist. A private key must be generated by each client for this purpose. Key negotiation seems only possible via a few first unencrypted chat messages that cannot be interpreted by other game clients.

4.1 Easter Eggs
As a user on any screen I want to be able to access one easter egg specific by each developer to have fun in my life.

5.1 Darkmode

As a user of the application i want to be able to see a dark-mode-themed application to be able to dive into a full fledged gaming experience®.

5.2 Fades

As a user between screens I want to see a smooth transition to stay within my full fledged gaming experience®.

5.3 Error Messages

As a user on any screen I want to be able to see the correct error messages to any problem I will have, so that I am informed about my situation and can adjust properly.

3. Jira Data

The team and mainly the scrum master decided to separate the tasks for the first release into two categories. First of which is backend logic and the second category is frontend logic. These two categories are the sprint headlines and represent each theme of the sprint.

3.1 First Sprint, Backend

Started on may 14th and ended on may 26th, the goal in the first sprint was to implement the backend logic. Mainly server communication and also the core parts, like basic FXML and web sockets so that a template is made ,that can be worked on in the further sprints and releases.

3.1.1 Stories

The first sprint was packed with 14 stories and the stories were estimated to be worth of 117.5 story points.They got separated into 3 epics which are design, web client, persistence. They also got separated into different priority levels which are high, low and medium.
Starting with the design epic, „basic FXML" was the first thing to tackle and it had the highest priority because it laid down the building blocks for the design epic. An estimation was made and „basic FXML" got two story points, which result in two hours of work, but the developers needed 5 hours for this story. That is because they needed time to read through forums to create the best possible result.
Also added was the „dark mode" which sets the theme of the game to be darker and more pleasant for the eyes. The estimation resulted in 10 story points but the story was done in 13 hours, because firstly the developers needed to check if the dark mode option is properly displayable in FXML and does not look too shabby. Secondly they had to read into this topic to make it as good looking as it is right now. With this story being done, the developers also finished implementing the story „go to register", which contained the story to switch to the register screen after pressing the „register" button. This was done fairly quick because it was implement by the developer who also implemented dark mode in the same story.
Next story being implement was „error handling". Error handling was about managing all the possible errors that could happen in any screen and creating messages that would make the user aware of the current error that is occurring. This story was estimated to be 3 story points worth of work and it was finished in 3 and a half, which is a correct rough estimate.
Last story in the design epic was the implementation of „fades". The developers tried to make our client to look like one window, where everything happens, instead of windows popping up and getting resized. The team thought that this would make the best experience for the users and it created a more modern and clean look. The employee needed 13 hours instead of 10. Reason for this were two main things the developers did, so that the fades look as good as they do right now. Firstly they had to read into some code and information that has been written by other sources like

stackoverflow.com and secondly the employees worked out a few solutions so that you can choose which one you like the most.
Following this, the developers worked on the persistence epic, specific, the „YAML game saves“. „YAML game saves“ are for the future of the game, so that the user can save games and start where he left off. This took the team 7 hours and it was estimated for 10, faster again. Reason for this was an experienced developer and his ability to read quick and understand fast.
The second finished story in the persistence epic was the story en-&decrypting values. This story was all about privacy and making sure that users who make use of our client have the best and most secure experience. By encrypting and later decrypting chat message, only users of the client of team B can understand. This story was a lot harder to implement than it was firstly estimated. It took the developers 31 hours of work to finish and polish the story to the fullest satisfaction of the scrum master. The story was estimated to be worth of 20 story points. The reason for the 11 hour delay were disagreements between the scrum master and one of the employees about the private key and how to handle and implement that feature. Those problems were resolved after a lot of discussions and comments in pull requests.
Last persistence story was the JSON „De-&Serialisation“. For this story, the developers needed to implement a class which can be called to load and save files and where individual keys are targetable. This story turned out to be a lot harder than firstly estimated and took the assigned worker 15 hours instead of 10 hours. That was due to the lacking experience of the developer for that story and some code style errors which had to be sorted out and discussed by and with the scrum master.
Finally the last epic for the sprint, being web client. One of the stories of the web client epic was the HTTP manager. The story was about implementing a class with functions that helps to communicate with the game server. 13 hours were estimated and 19 hours got invested into it. There were multiple reasons for this. The team had to learn the process of building a HTTP manager which includes knowledge in JSON, HTTP components and mockito for testing the server.
Next finished story is the response body which deals with the response itself, its status code and which throws error messages. It was guessed to take 5 hours and it took the developers roughly one hour more. Because again our developers need time to read into the topic and understand the code of other employees, so that they could implement that story properly.
Another finished story were the REST endpoints. The challenge for this story was to implement request classes for every API endpoint, each one featuring an inner builder class. An estimation resulted for this story to be worth of 25 story points but the developers finished in 8, because they were already experienced in this subject and concentrated their work hours into this story.
Now getting closer to the end of the sprint, there were two more finished stories. One of which were socket endpoints. That story was about creating a class that acts as a web socket client with a message handler, so that a chat class can be implemented. This story took the developers 11 hours and estimated were 7 and a half hour. The cause of this was that the finished code for the story had to be changed, because they wanted to use GSON instead of simple JSON which are libraries for reading and writing JSON data and that is needed for the server communication.
At last the final story for that sprint was the API story, which was deleted because it turned out that it was not needed, reason for this was that it was already implemented by the developer who finished the REST endpoints story, so we scrapped two and half story points for that sprint.

3.1.2 Unfinished Stories

One story was left and shifted into the second sprint due to time constraints and the work being more difficult and harder for the developers. The left off feature was the chat, which is a class that represents all the feature of the chat. With that said, this story will be finished first in the second sprint. This gave the team 3.7 story points more to work on in the second sprint.

By the end of sprint one, the developers managed to finish one of requirements which was the notification to the server. But that was not the only thing, the developers mainly implemented the basic structures and functions in that sprint. Like the name of sprint said „backend“. The next sprint is all about the functionality and completing the requirements which are almost all about visualisation of the work that's already done in the first sprint.

3.2 Burndownchart

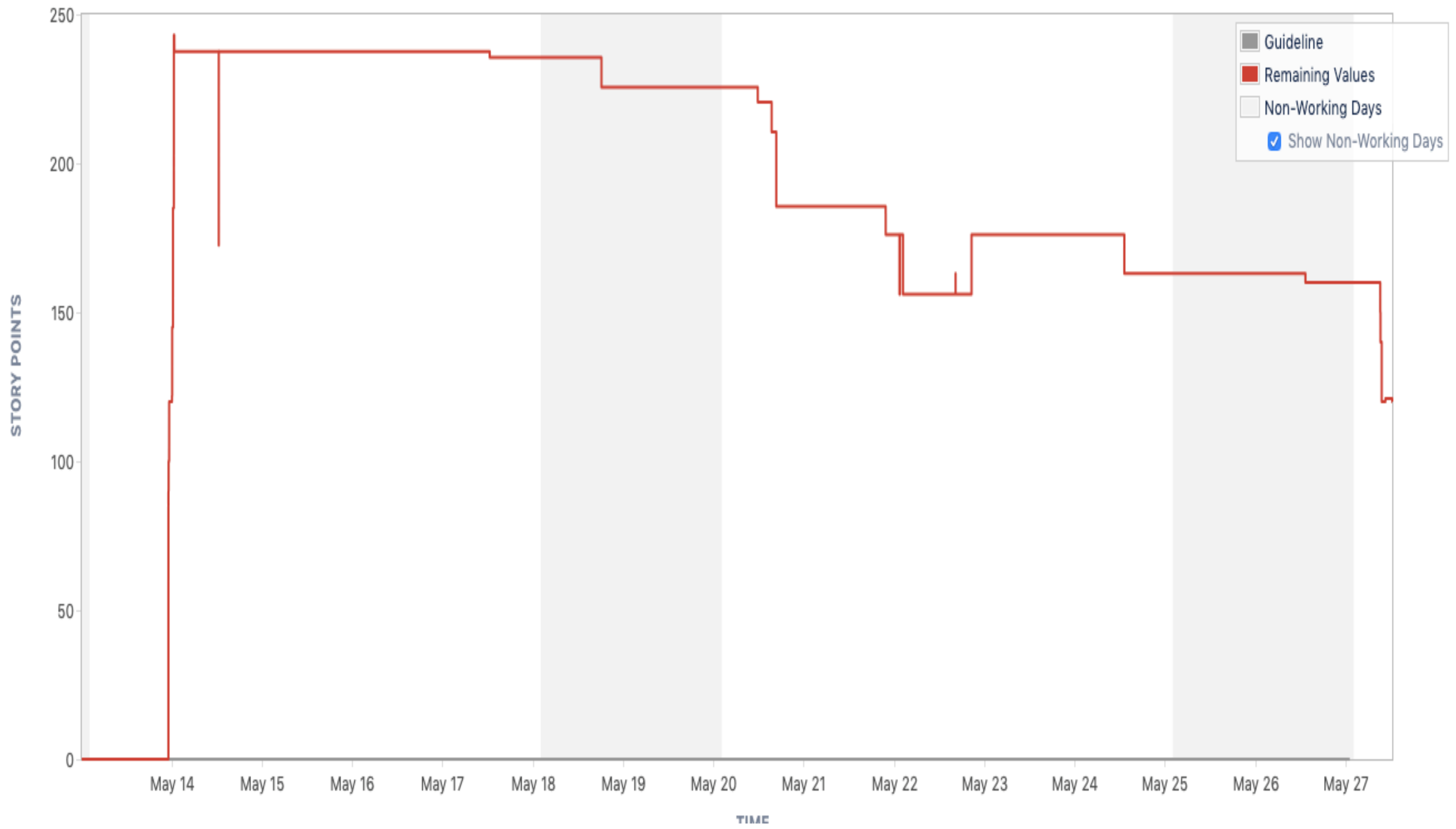


Figure 4.1

The first sprint ended and this is the burndown chart that we got, after the first two weeks of release one. At the beginning, the developers had some troubles and communications errors which resulted in the chart to start on May 14th instead of May 13th. After fixing some of those uncertainties, they managed to gamble out the story points for each story and the first sprint started.
Starting out the scrum master fixed some estimations that were estimated to be more hours worth than they actually were and added a new story which needed to be implemented being the socket endpoints. After creating all necessary stories, the tutors told them that some of those stories were not segmented to their fullest content. So the scrum master reopened 65 story points and segmented them properly, which resulted in that line after one day. Finally the developers got assigned to at least one story and they started working on those. A few days went past and nothing happened until the first story was completed and signed off from the scrum master, that was on May 17th. A lot of other stories were also already done in that period but, the scrum master, who did an excellent job in checking on code and supervising everything, did not let those pull requests past him, before everything was in a good shape and working properly. Those pull requests slowly started to resolve and new stories were assigned to the developers. Resulting in the graph to slowly get lower and lower until the May 20th, where the first week of the first sprint ended. 65.5 story points were

completed and about 55 story points were left to be done in the second week of sprint one. Week two was starting in the same way the first week ended, stories getting assigned to developers and pull requests slowly getting resolved, shown by the burndown chart. On May 22nd one of the developers mistakenly closed three stories which were not finished nor reviewed to that point in time. So our scrum master reopened those stories and reviewed all of them. Resulting in two stories being closed the same day and the third story, which was not properly done, was left open, as you can see on the graph. Following this everything went smoothly its way. Pull requests were resolved, each one reviewed many times so that the user has an unforgettable experience using the client of team B. And the last remaining stories got assigned to the developers.

3.3 Second Sprint, Front End

New sprint, new epic. Heading into the new sprint which was scheduled to start at the 13th May and end at the 9th of June, the scrum master created a new epic called screens. That is of course because the sprint was all about visualisation and representation of everything that has been done in the background.

3.3.1 Tasks, Issues, Bugs

First things first, on the 28th of May the server behaviour changed and one of our tests failed, which needed to be fixed. This was resolved fairly quick and took the developers one hour to fix. Another bug which spawned were broken FXML paths, that needed to be fixed. This was cause by invalid FXML resource links, and lead to the problem that the application did not start anymore, these bug fixes were listed as tasks.
After fixing some of the issues which were created accidentally, by internal or external influences, the team started to work on more of those issues that were listed as tasks. Two of those were in the epic, design, which had only two small tasks in the second sprint. First of which was the loggers and errors task. The developers needed to use the existing error handler to only display error messages readable and understandable for the user. Also a logger was to be implemented to capture all statuses. The second and last task in the design epic was the „fix check styles warnings“ task. A developer needed to check all the written codes for warnings because they implemented a new check style build, which forced the developers to write code in a tidied up way. The developers spent 11 hours on this because it was a lot of code and they tried to structure the imports of java classes alphabetically.
More tasks were in the „web client“ epic. Web socket handler was one of the first things done in the web client epic. It was about implementing all web socket endpoints and creating something like a „On-Chat-Message“-handler. This was worked on pretty fast, but after some hours that were put into this task, the developer decided that it would be better for this task to be finished in release two, because the game web socket endpoint is the only partially usable server side yet. The developer has put in 12 hours and 30 minutes, but also fixing and creating packages and structure the project a lot more. After creating an abstract web socket handler, the developers needed to implement a test for that. So they started to mock again like they did on the REST endpoints. This took the developers 4 hours and 30 minutes. Another task that had to be tested were the REST requests and they also needed to be mocked, so that the team can see early on if it makes actually sense what they are trying to implement.The developer assigned to this task needed 10 hours to mock all the test for the REST requests.

3.3.2 Stories

The team had one Story for the „web client“ epic, that was the chat story. This story was about implementing a class that represented every feature of a chat, like a history and scopes. For this story we estimated 2 and half story points but the developer needed 10 hours, reason for this is the lacking experience of the developer for this story.
Last and biggest epic for this sprint was the „Screen“ epic. This epic was all about implementing visible elements for the user and make them work. Starting from the bottom the first thing was the „Login“ story. This story was about implementing functions that allow the user to enter his name and password and then press the login button to be further directed to the main screen, but only if his username was already registered. Estimated for this story were four and a half story points and the developer needed 4 hours which resembles the estimation. Following this, they implemented the „Remember Login“ story. The employees added a checkbox, which enabled the user to click into the box if he wants to and let the client remember his entered username, so that the user does not get annoyed by entering his name over and over again for every login. The team estimated 7 and half story points for this. And the developers needed exactly 7 and half hour which again makes it a good estimation.
After they implemented the login functionality, the developers started to work on the register screen. First thing was the „Register“ story. As the name of the story says it was about implementing the functions to enable the user to enter his username, password and confirmed password to register and further login afterwards. Also a „register“ button needed to be implemented. The developer needed three hours for this story and estimated were 4 and a half hours. He was faster than expected because he already implemented the login and was experienced with a story like this. The same developer who implemented „Login“ and „Register“ also implemented the „Cancel Register“ story, which was about implementing a button that directed the user, after clicking on it, to the welcome screen. He needed three hours for a story worth of one and a half story points, this is due to him being very disciplined and tidy, because he also changed some code to avoid duplicates and make the code more readable for the other developers. After completing the functionality of the register screen, the main screen was the next thing to tackle. First story was the „Create Game“ story. The developers needed to implement textfields and a button to let the user name their game and decide between two and four players. Clicking on the button „create game“, directs the user into the game lobby and makes the game viewable to other users in the client. The team estimated roughly four and a half story points for this and the developer needed four hours and 20 minutes. Next story was a bigger story with 15 story points, the „View Games“ story. The challenge for this story was that the developers needed to implement something that made all the created and still running games visible to the user and also show the user how many players are in a game and if there is still space to join the game. They needed 16 hours to complete this story with every feature, which was a good estimation for this story. Another big story in this sprint was the same story implementation wise, which was the „View Lobby Users“ story. This was the same as the „View Games“ story, but the developers needed to display the users that are currently logged on in the client. Again this story got 15 story points and the developer needed only 10 hours, because he already worked on the „View

Games“ story and since those two stories were practically identical implementation wise, he was faster than expected. There are only three more stories that were finished in the second sprint. First one being the „Join Game“ story. The implementation of this story enabled the user to click on games that are displayed in the main screen and join them if there is space for one more user.
The developer needed five hours and 15 minutes for this story to be completed and estimated were four and half story points. Second story was the „Delete Game“ story which was implemented by the same developer, because both stories were about interaction with the buttons that are appearing created by the story of „View Games“. This story was about implementing a button that allows the user to delete games that are displayed in the main screen. He needed two hours for this and estimated were four and half story points just like the „Join Game“ story. He was faster than estimated because he already had done something similar to this story and was experience in implementing code like this. Lastly the final story for the sprint was the „Logout“ story. The developers needed to implement a button that allows the user to log out of the main screen and get directed back to the login screen. Roughly nine story points were estimated for this story and it was finished in six and half hour, because it turned out to be less complicated than it was thought to be and due to the developer being experienced in this kind of story.

3.3.3 Unfinished Stories

Three stories were not completed in the second sprint and ultimately in the first release. These three being the „Send Chat Messages“, „Receive Chat Messages“ and „Chat Encryption“. Both of those stories were about interaction of the chat and actually communicating with other users in the client. These three stories get shifted into release two and will be worked on as fast as possible. The team decided to delay the first two stories because, they wanted to give the users an unforgettable chatting experience and they did not want to slack on those stories. So they needed a little more time because they focused on the other stories first with the same mentality of giving their best and trying to give users the best experience of all RBSG’s.
The third story was not done because one of the developer had to leave the team due to medical reasons.

At the end of sprint two we managed to nearly fulfil all the customers needs and wishes plus implementing feature like dark mode, fades, and a „Remember Login“ checkbox. The only thing the team could not finish was the chat functionality as described above. The team tries their best to complete all needs and wishes for the next release but also implement the left of chat functionalities and implement a lot more new ideas that were discussed in meetings and such.

3.4 Burndownchart

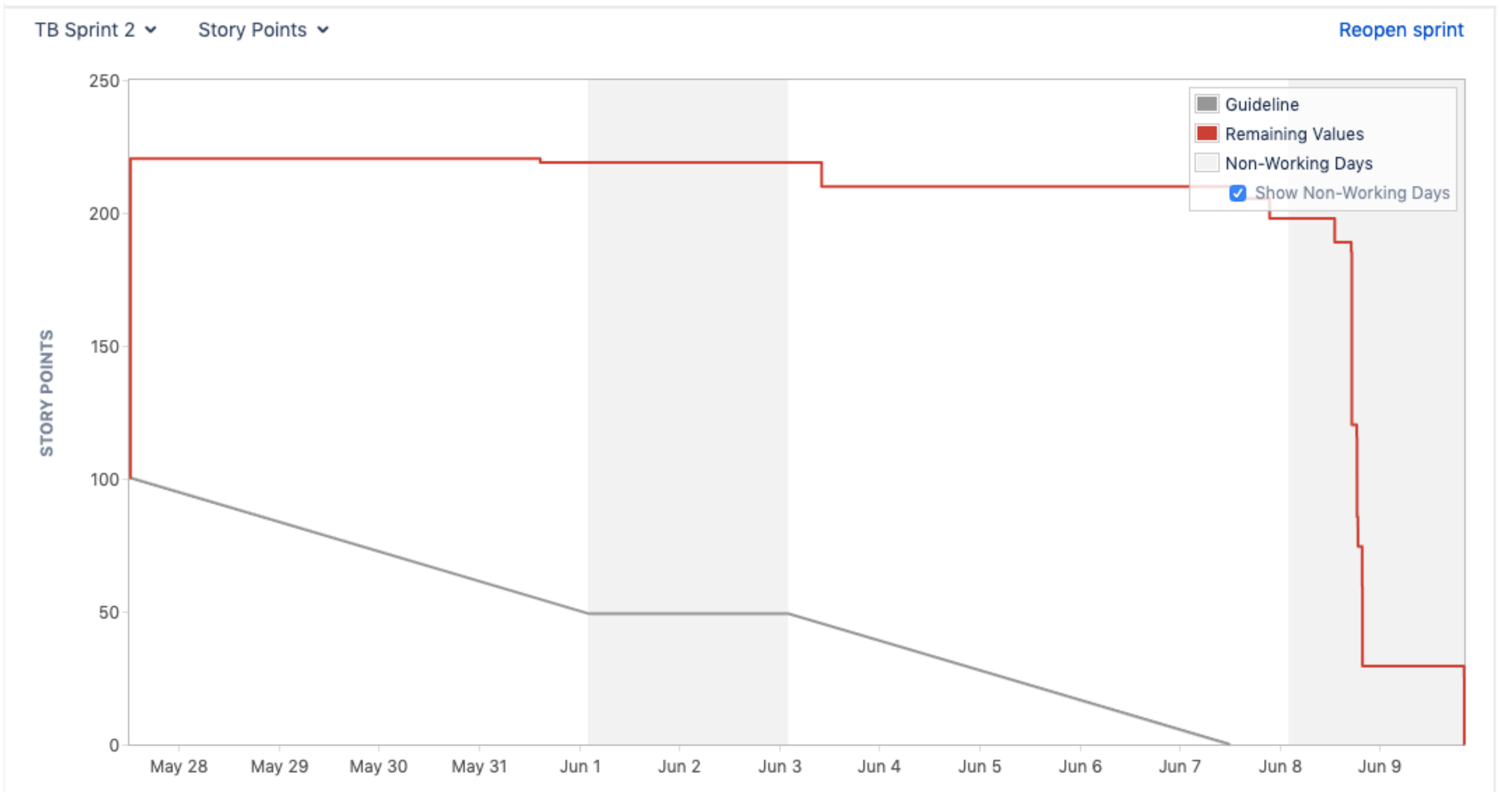


Figure 4.2

This Burndown chart was a little more messy than the first one. Because the team had miscommunications in the first sprint on the first few days. Theses miscommunications were a mix of rushing the implementation of Jira stories and not knowing how to estimate things correctly when we were not even sitting together discussing things. So the scrum master decided to only add tasks instead of stories because they made more sense content wise. But tasks in Jira do not effect the Burndown chart, which was a part of our job. This lead to implementing more stories that were tasks at the beginning. Also the epic's that the team implemented double the story points which was not intentional.

So basically the whole chart begins to be interesting at the 8th of June. But first let us take a look at the beginning. The start goes up again because as mentioned the team had to convert some tasks to stories which resulted in more story points. After that the chart goes down slowly, that is because the team also had tasks that were not converted to stories instead they stayed tasks because they were bug fixes for example and no real stories. So the team worked on something but it is not displayed on the Burndown chart. The first thing being done as a story was at the 31st May and it was a minor one. Then again only bug fixes and stories which had been finished. Following this on June the 3rd another story has been completed. But then again nothing exciting happens until the 8th of June. Where everything slowly comes to an end. This delay of finished stories is due to the carefulness of the scrum master, reviewing and commenting every pull request, until everything was to his content. The chart shows that stories are slowly finishing and then some big leaps appear. This is because the team finished epics which had a lot of story points and a lot stories in them. And also because one of our employees had a story, which blocked off several other stories. So the developers needed to wait until that blocking story was done, seen in figure 4.2, 8th of June.The last few steps were one last story that has been finished and 25 story points that has been shifted over to the second release due to the reasons above in the end of the second sprint part.

4. Results

As already mentioned the team managed to nearly complete everything and also added some really nice feature. The team has worked pretty hard on accomplishing this and they tried their best to make it as good looking as it is right now.

4.1 Final Screens

In the following three pages, the reader can see and understand the teams design choices and see the end result of release one.

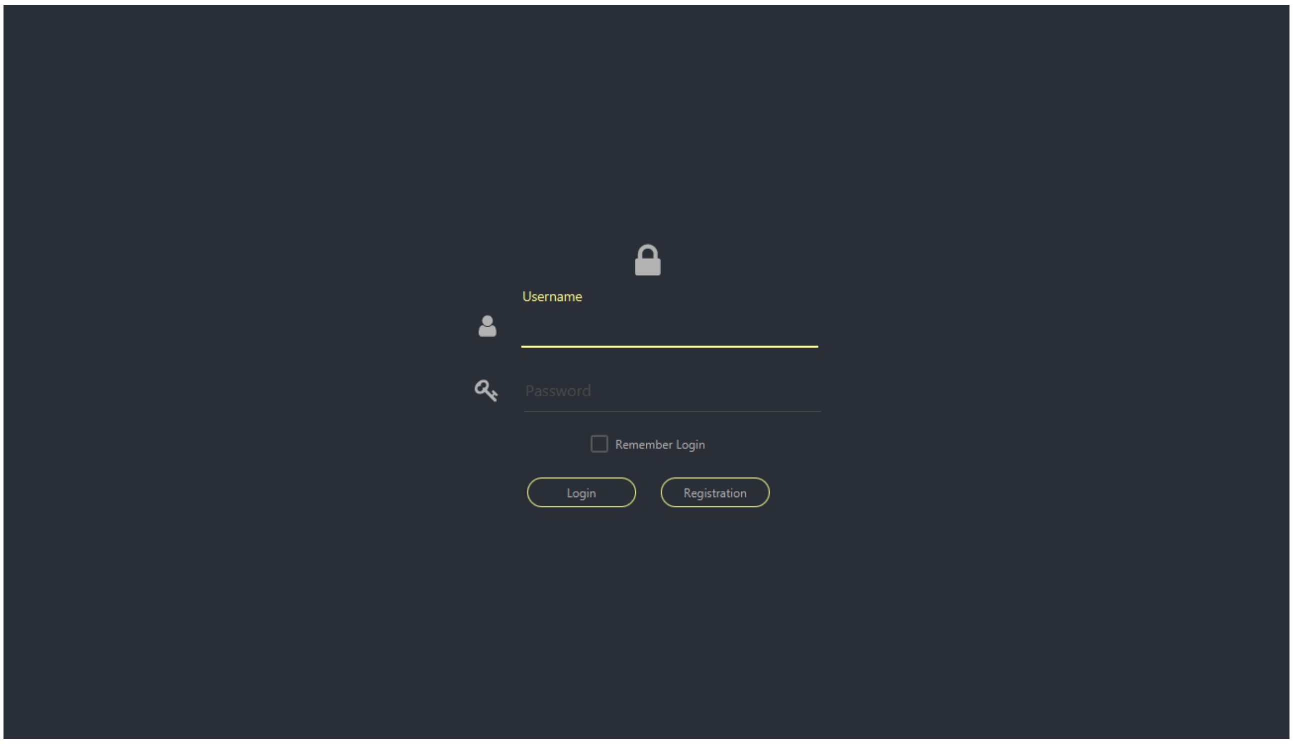


Figure 5.1

4.1.1 Welcome Screen

This is the result the developers got at the end of release one. The team focused on simplicity and smoothness making the look of the welcome screen very simple without any extra fuzz but with a lot of love to detail. The team added dark mode to every screen since they thought this should not be an option it should rather be the default look of the client.

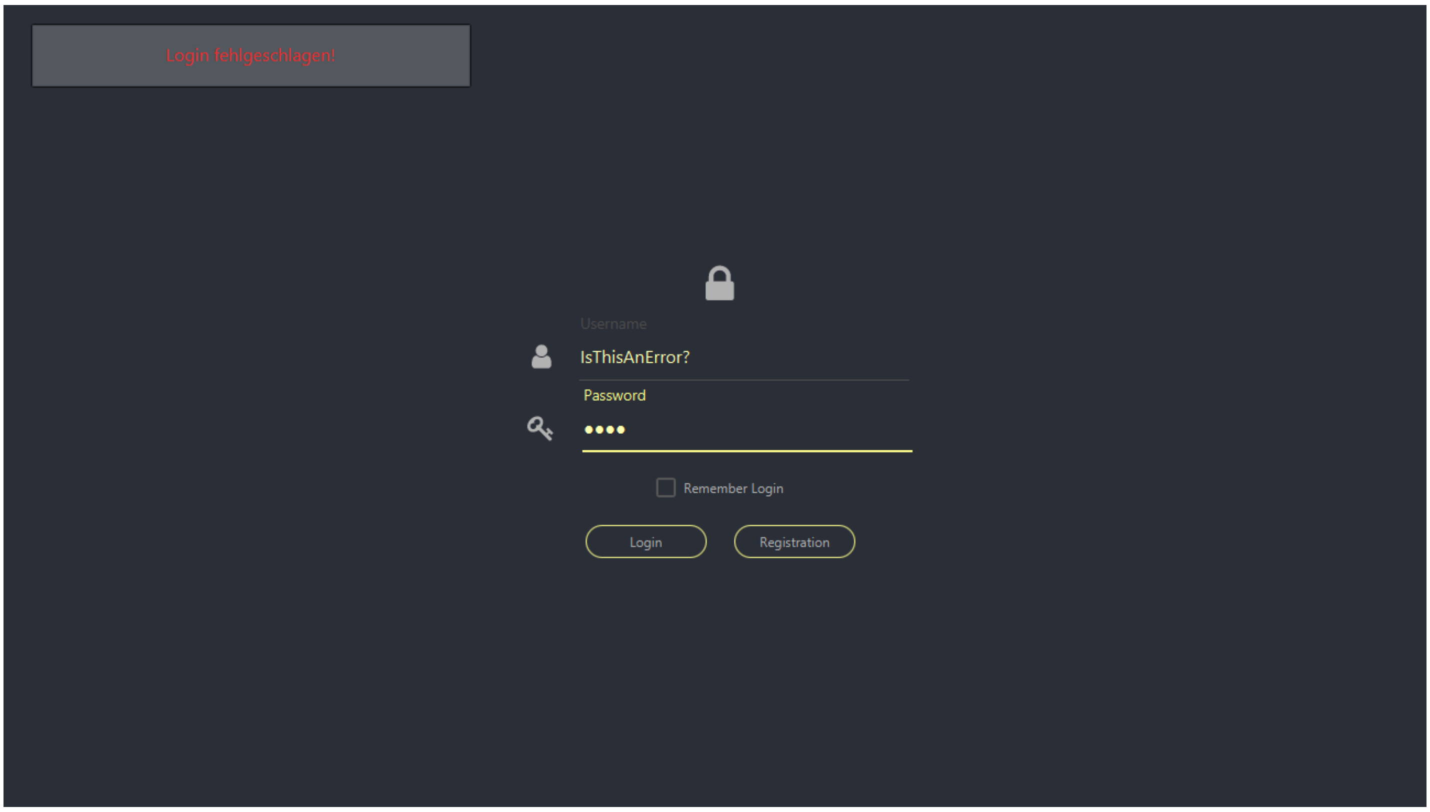


Figure 5.2

4.1.2 Error Screen

As described in the first few pages the team tried to let error messages fade in instead of letting pop up windows taking control of the screen, as shown in figure 5.2. Of course this is not the finished product and will have tweaks and some minor changes to the design.
A feature that the team thought of, was the „Remember Login" checkbox to give the user the option to login over and over again without getting annoyed with entering the username and password all the time.

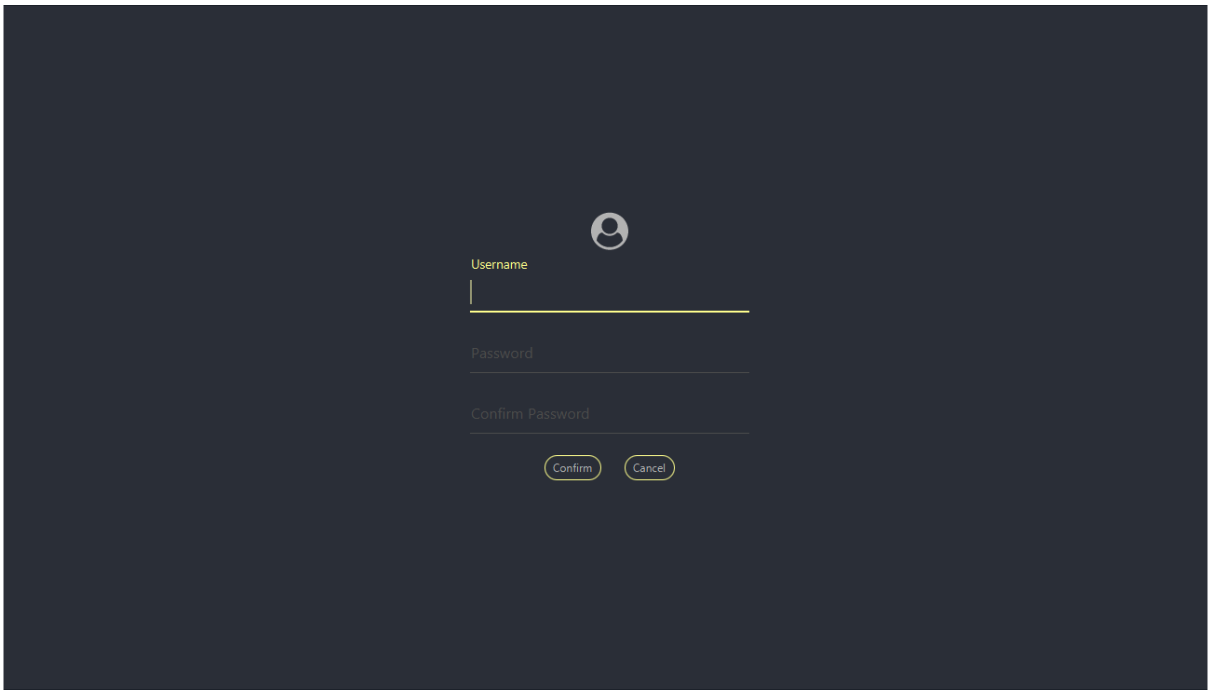


Figure 5.3

4.1.3 Register Screen

Now looking at the register screen at the end of release one. The team tried to make both welcome and register to look pretty similar to give the user the illusion that it is one whole instead of two or three different windows. To also boost this idea of a „one whole“ they added fades for every transition happening between the screens to make it look smooth and give the user a very appealing look. Error messages are handled the same way as in figure 5.2 and will be displayed as fade in, in the top left corner, each saying something unique for each problem that is occurring, so that the user knows how to react. Both buttons „Confirm“ and „Cancel“ work as intended. The „Register“ button was changed to „Confirm“ because the team thought that this would align better with the „Cancel“ button and it is more fitting than „Register“ in the opinion of the team.

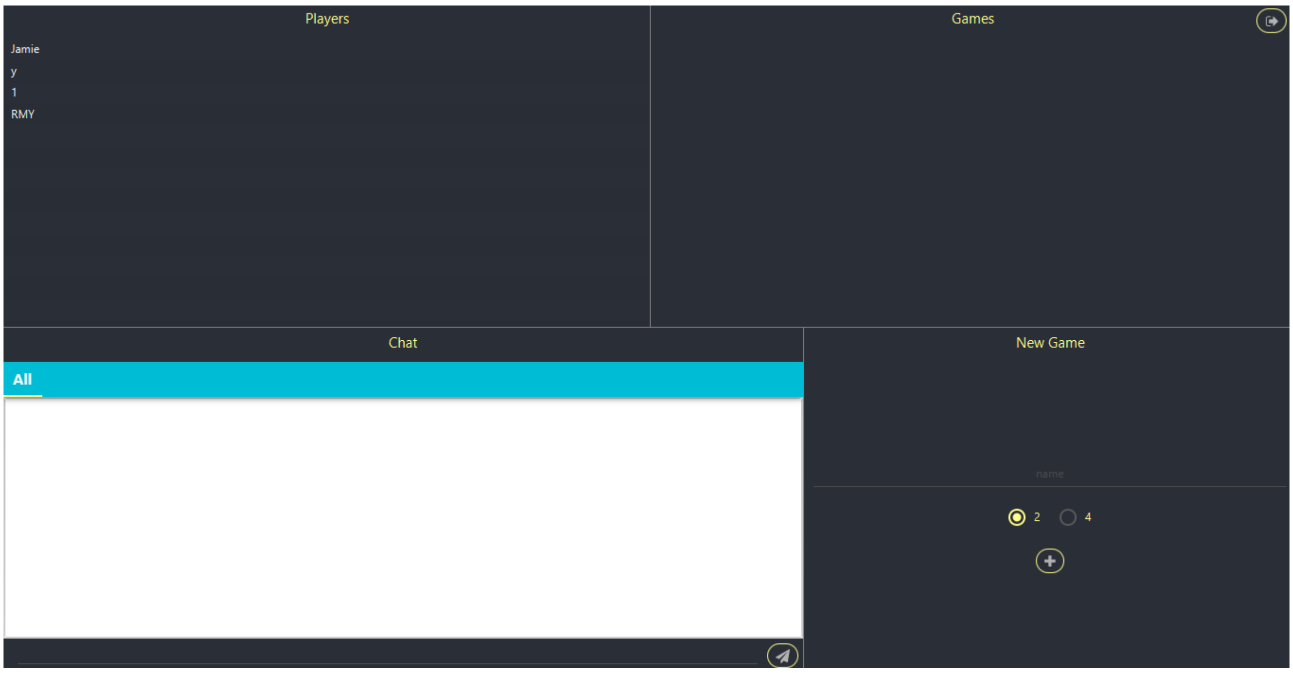


Figure 5.4

4.1.4 Main Screen

This is the final look of the main screen. The team tried to cling on the mockup and made choices about chat size and the size of players and games. They thought it would be more appealing if the players field gets bigger and the chat, which puts more focus on both of these things. And also giving the developers later more room to implement interaction with other users. Another choice we made was the limitation of players to two and four, since the game only starts with four or two players. And the last change the developers did was the cutting of the settings button on the top right, seen in figure 1.3, this is due to time constraints and also lacking features that are not yet implemented. So there was no need for something like this, but will be changed in future releases. The rest of the design got a new darker and cleaner look making it more appealing to the eyes.

5. Concluding words

The first release went fine, the team had some troubles but solved them rather quickly because everyone pulled together through every step back and now we're here at the end with a pleasing product that fulfils the expectations that were given in the first week. Of course two features did not make the cut in this release but this was due to a medical situation to a developer who left the team in the last week. Those two feature are going to be implemented immediately and will be done in the first sprint of release two. The team takes all the experience they gained from these four weeks and will work harder and better in future releases.