BLOKS3 – Tetris as a board game

Team 3

Ideated and designed by:

Linus Baudenbacher

Tore Buscher

Tilman Otto

Tobias Geilen

Tim Gutberlet

Contents

1. Project Plan and Overview

@author tgutberl

1.1. Project Goal

The aim of the project is to develop a multiplayer network version of the game Blokus. The artefact to be developed should fulfil all known rules, as well as all requirements specified in the specifications. Consequently, the project will be considered a success if the finished game is fully tested and delivered on time together with the required design artefacts. It should be fully playable and bug free.

1.2. Approach

We approached the project considering the time plan we created as a Gant Chart for the different phases of creation. However, the main idea was to follow the agile project management approach SCRUM. Therefore, we firstly planned to create a prototype that already includes first versions all different aspects of tasks we wanted to fulfil and then create the fully working game based on that prototype. For that we had a task backlog and assigned tasks in weekly sprint calls. In the different phases of our project, as seen in the Gant Chart we focused more on special areas of tasks. However, to really follow an agile approach, we constantly reiterated and reworked all different aspects of our program in the Sprint cycles as we got on with the project.

1.3. Team

Name	ILIAS Username
Tobias Geilen	**tgeilen**
Tore Frederik Buscher	**tbuscher**
Tilman Morten Otto	**tiotto**
Linus Fabian Baudenbacher	**lbaudenb**
Tim Gutberlet	**tgutberl**

1.4. Documents

Assignee	Document Name
tgutberl	Project plan
lbaudenb / tbuscher	Description of third-party libraries
tgeilen	Architecture diagram
tgutberl	Domain Model
lbaudenb	Operation contracts
tbuscher / lbaudenb	Use cases (+user stories)
tiotto	UML Class Diagram
tbuscher	Sequence Diagrams

1.5. Tasks:

1. Coding

1.1. Game Logic

Purpose: The Game Logic controls the whole In-Game Process, handles user and AI moves, calculates if moves are possible and when the game is over. It keeps track of all tiles left and of the whole board.
Assigned: totto, tgutberl, tgeilen

1.2. Menus

Purpose: This Taks includes the different Menu and Setting views and their logic. The views will be written in FXML, and the functionality will be implemented in the different controllers.
Assigned: tgutberl, lbaudenb

1.3. In-Game Frontend

Purpose: The In-Game Frontend displays the whole process of the game and lets the user interact with the game and recognizes the inputs. It handles all inputs of the user and gives them over the the game logic.
Assigned: lbaudenb, tgutberl

1.4. AI

Purpose: The AI consists of an easy, middle, hard and random mode. All four modes differ in their difficulty, and they can play against real players and take part in a normal game and play against each other or in the tutorial.
Assigned: totto

1.5. Network

Purpose: The Network handles the logic and communication between client and server in different packages. It will manage the network interaction in multiplayer games and will therefore interact with the game logic.
Assigned: tbuscher, tgeilen

1.6. Server

Purpose: If a player wants to host a game, he/she can start a server on his local computer. Other clients then can join this server if they are in the same local network as the host and they can then play multiplayer games. The communication of packets between the different clients and the game logic will then be handled on the server.
Assigned: tbuscher

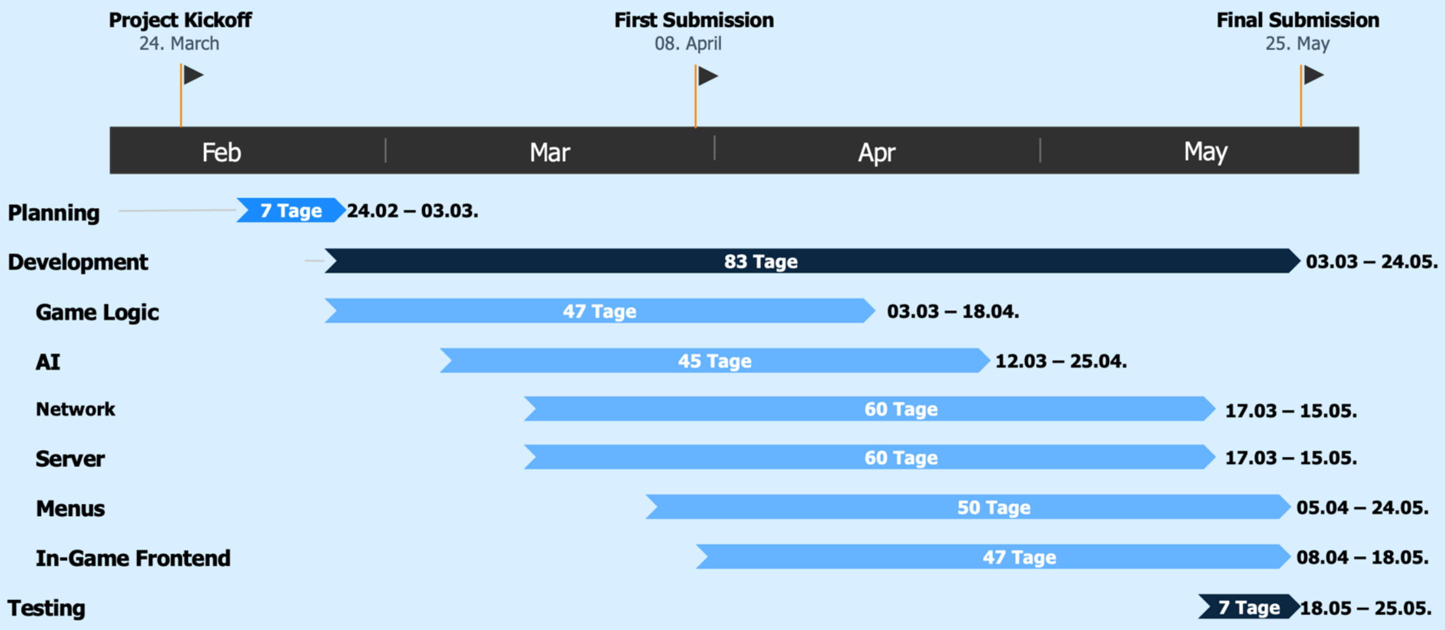
1.6. Final Report

All in all, we have sticked to our time plan. However, we had to implement some major features and fix bugs, while we were also testing.

All requirements should be fully satisfied and implemented. We build 4 different working JAR versions, for different Operating systems. There may be some minor Bugs in the UI and some Server communication issues, that we could not have noticed, as there are many edges case we maybe not thought of. All software has been fully commented and no style issues should be noticed.

The Project is considered a success and the team had no difficulties in working together and staying focused until the end.

1.7. Time plan:



Project Kickoff
24. March
First Submission
08. April
Final Submission
25. May
Feb
Mar
Apr
May
Planning
7 Tage
24.02 – 03.03.
Development
83 Tage
03.03 – 24.05.
Game Logic
47 Tage
03.03 – 18.04.
AI
45 Tage
12.03 – 25.04.
Network
60 Tage
17.03 – 15.05.
Server
60 Tage
17.03 – 15.05.
Menus
50 Tage
05.04 – 24.05.
In-Game Frontend
47 Tage
08.04 – 18.05.
Testing
7 Tage
18.05 – 25.05.

2. Description of third-party libraries

@author tbuscher / lbaudenb

The Project uses Java 17, more specifically the Oracle JDK 17. As an IDE IntelliJ is used for development, although other IDE should work without issue.

Maven is used to handle dependencies and the building of .jar files. Maven Plugins used include:

- maven-clean-plugin

  To remove any unwanted files that were created
- maven-compiler-plugin

  To compile java code
- maven-install-plugin

  To ensure everything present in the pom.xml is present during runtime
- maven-resources-plugin

  Make sure resources are available within the jar
- maven-jar-plugin

  Build a single jar from the entire project
- maven-surefire-plugin

  To run tests present in /java/test

Notable dependencies that are needed for the application to run properly and what they are used for here are listed below.

- JavaFx by openjfx:

  Used to implement the project's graphical user interface and play music.
- Sqlite-jdbc by xerial:

  Persistently saving scores and accounts to a database within the Project.
- Commons Codec by Apache:

  Hash passwords using sha256 before sending them over the network.
- Java-jwt by auth0:

  Used to generate tokens for token-bearer authentication.
- Eclipse Jetty:

  Used for the WebSocket-communication, both server and client side features are used.
- Eclipse Jersey:

  Used for RESTful communication: Running a server that handles requests.
- JAX-RS:

RESTful communication: Providing Resources to the server and establishing a connection as a client.

- SLF4J:

  Used to provide Logging.

- Jackson by Faster-XML:

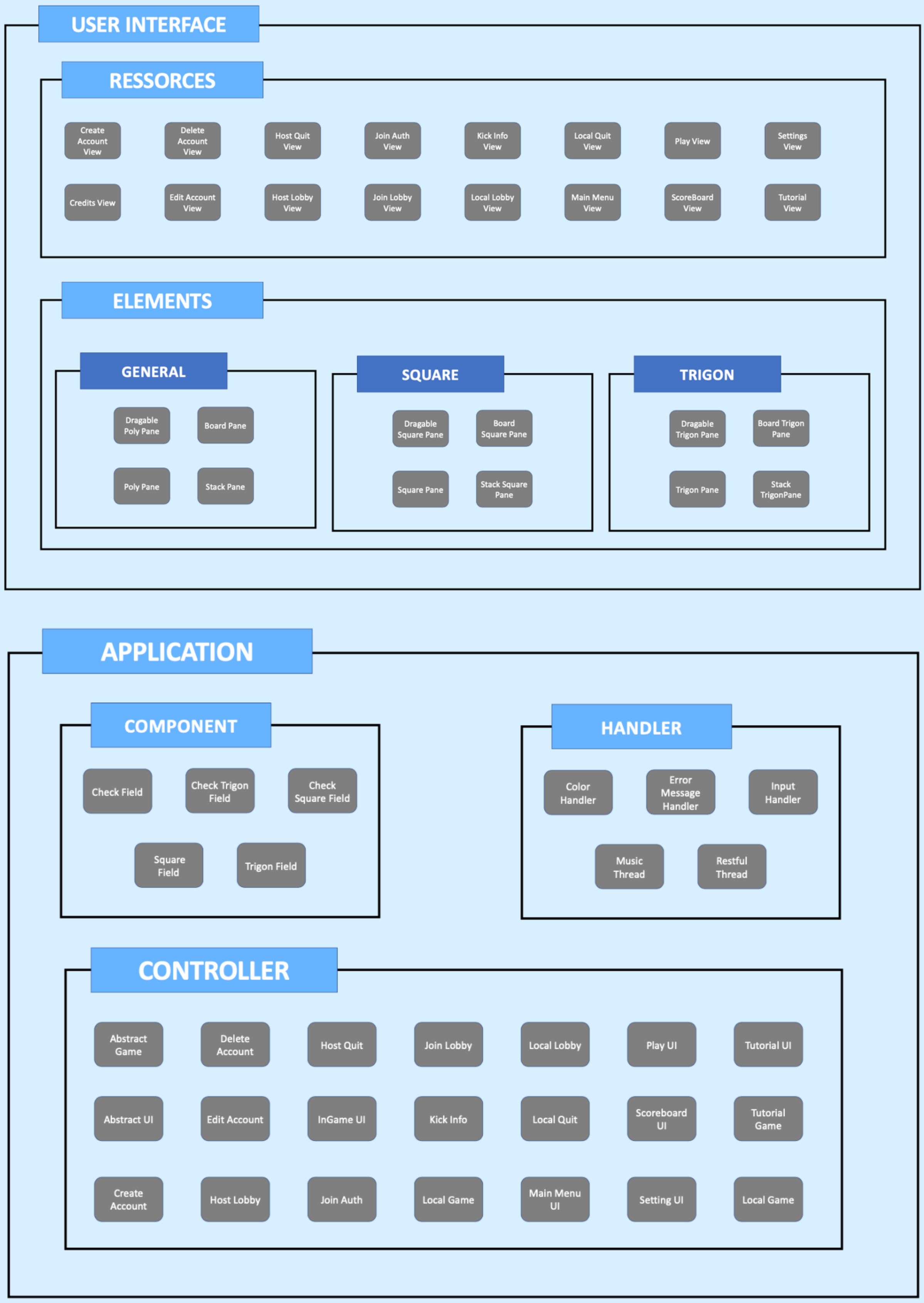
  De- and encode any packets sent by server & client as JSON.

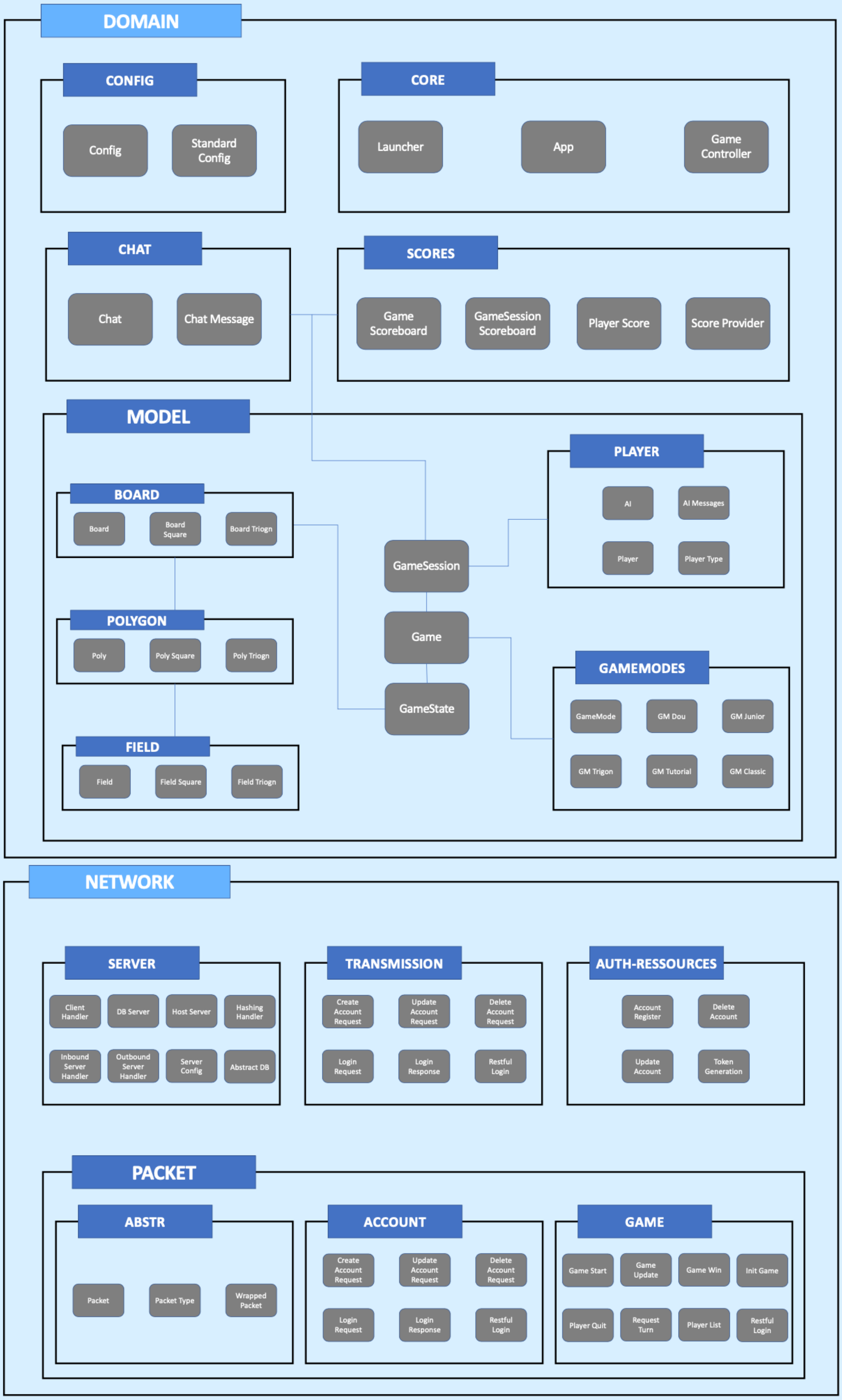
- JUnit:

  Enables the usage of meaningful test classes.

3. Architecture Diagram

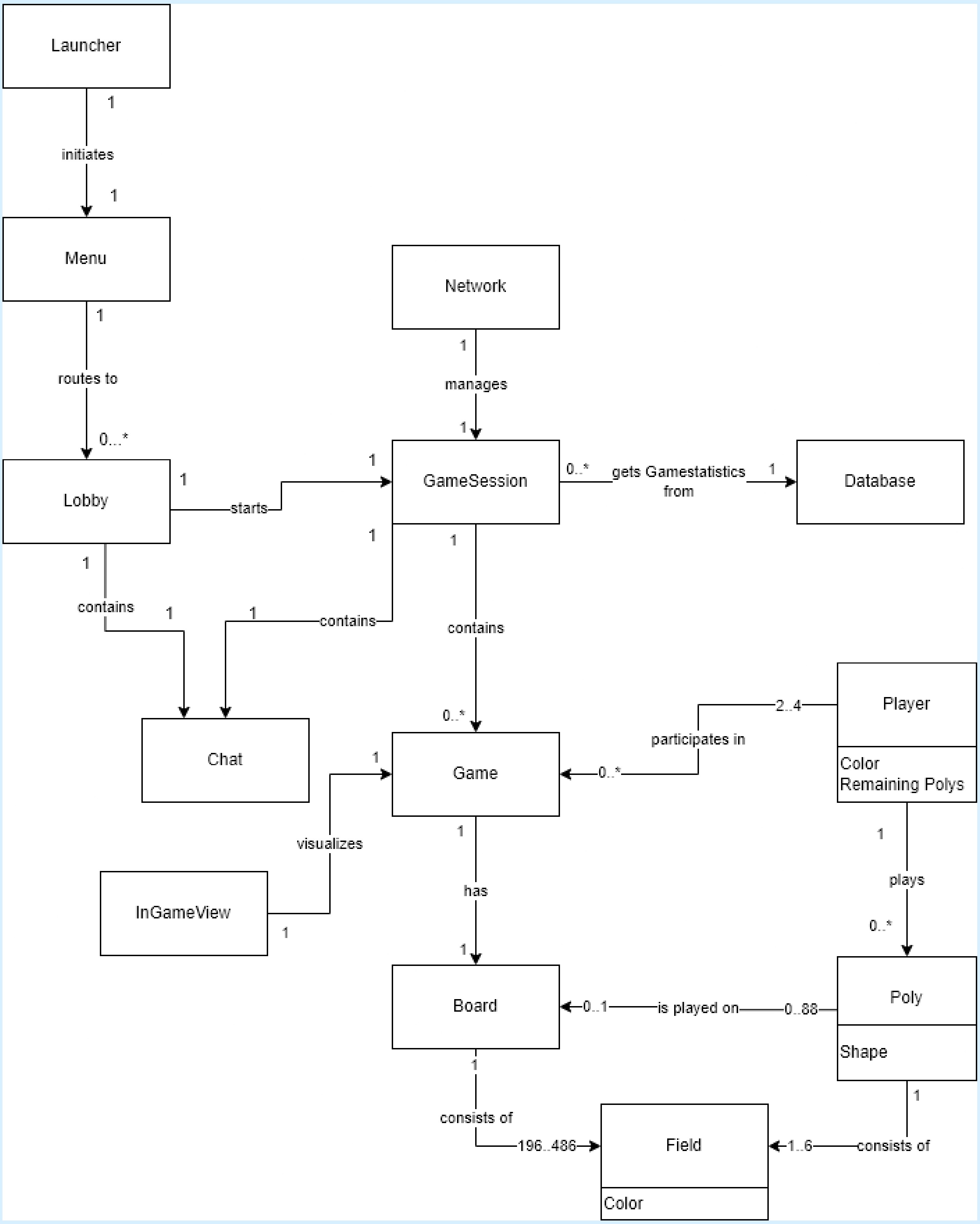
@author tgeilen




DOMAIN
CONFIG
Config
Standard Config
CORE
Launcher
App
Game Controller
CHAT
Chat
Chat Message
SCORES
Game Scoreboard
GameSession Scoreboard
Player Score
Score Provider
MODEL
BOARD
Board
Board Square
Board Triogn
POLYGON
Poly
Poly Square
Poly Triogn
FIELD
Field
Field Square
Field Triogn
GameSession
Game
GameState
PLAYER
AI
AI Messages
Player
Player Type
GAMEMODES
GameMode
GM Dou
GM Junior
GM Trigon
GM Tutorial
GM Classic
NETWORK
SERVER
Client Handler
DB Server
Host Server
Hashing Handler
Inbound Server Handler
Outbound Server Handler
Server Config
Abstract DB
TRANSMISSION
Create Account Request
Update Account Request
Delete Account Request
Login Request
Login Response
Restful Login
AUTH-RESSOURCES
Account Register
Delete Account
Update Account
Token Generation
PACKET
ABSTR
Packet
Packet Type
Wrapped Packet
ACCOUNT
Create Account Request
Update Account Request
Delete Account Request
Login Request
Login Response
Restful Login
GAME
Game Start
Game Update
Game Win
Init Game
Player Quit
Request Turn
Player List
Restful Login

4. Domain Model

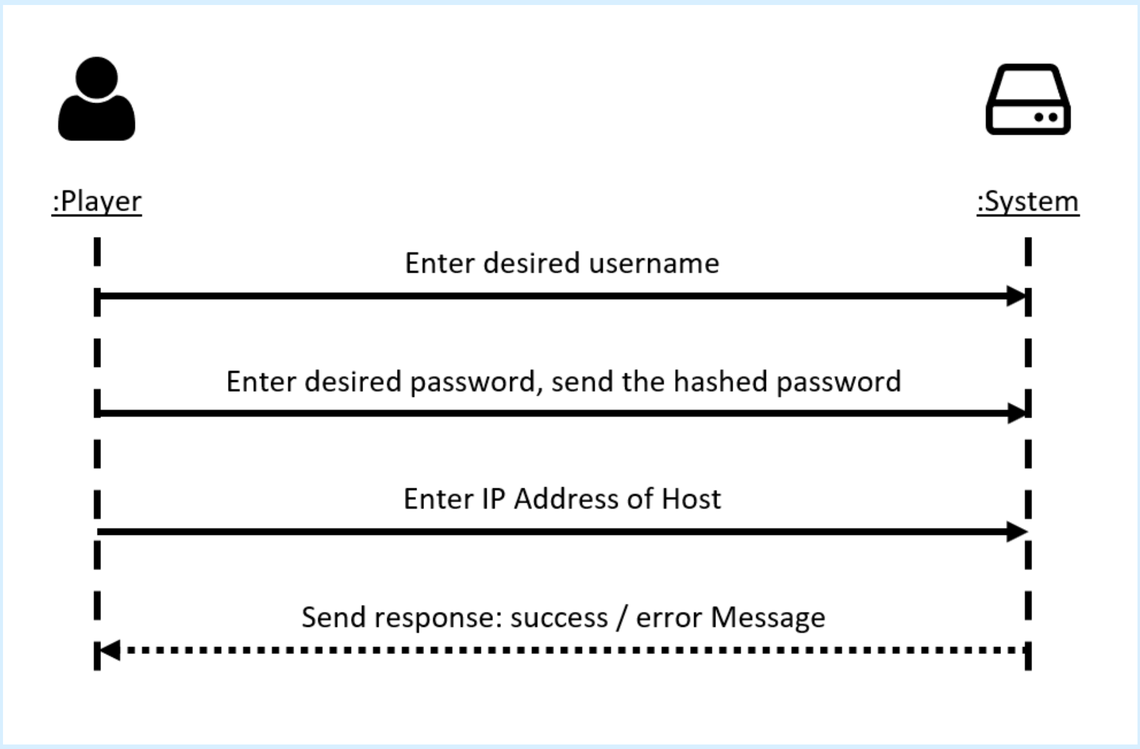
@author tgutberl



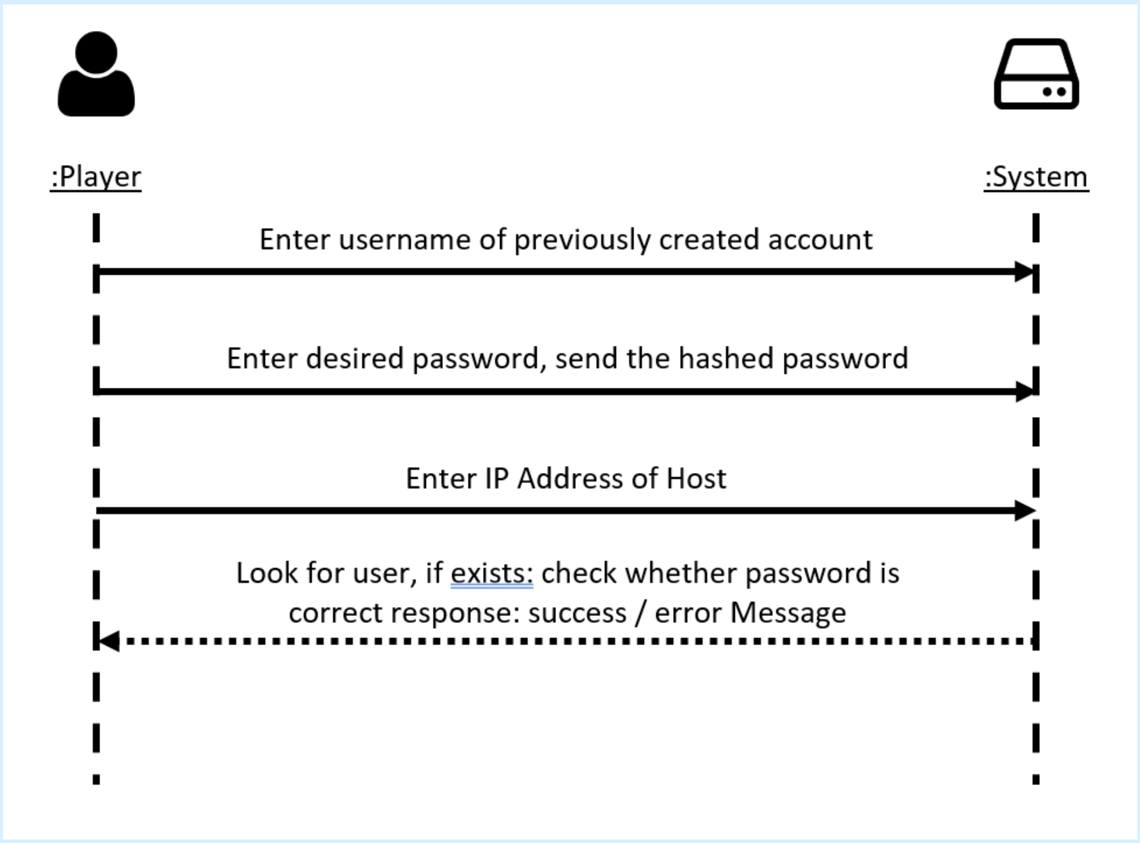
5. System Sequence Diagram

@author tbuscher

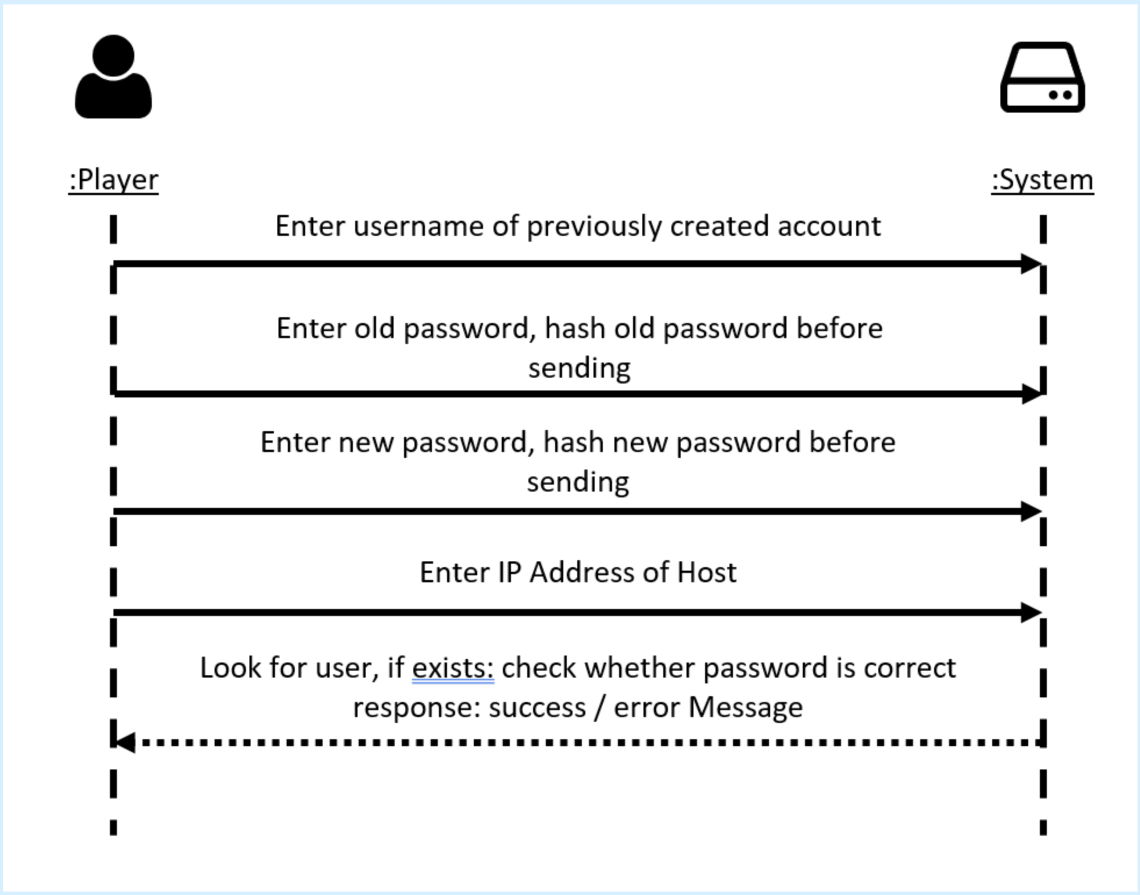
5.1. Usage of remote accounts: Creation


:Player
:System
Enter desired username
Enter desired password, send the hashed password
Enter IP Address of Host
Send response: success / error Message

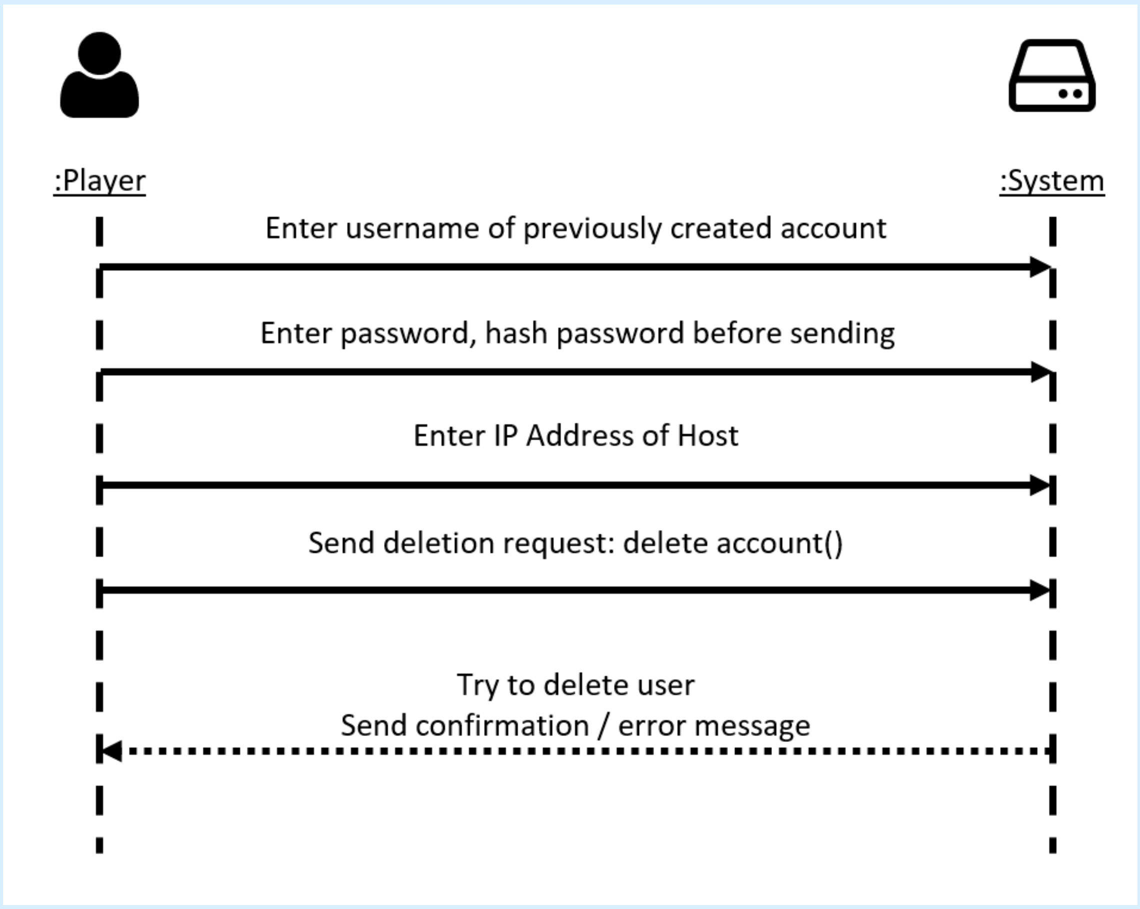
5.2. Usage of remote accounts: Usage


:Player
:System
Enter username of previously created account
Enter desired password, send the hashed password
Enter IP Address of Host
Look for user, if exists: check whether password is correct response: success / error Message

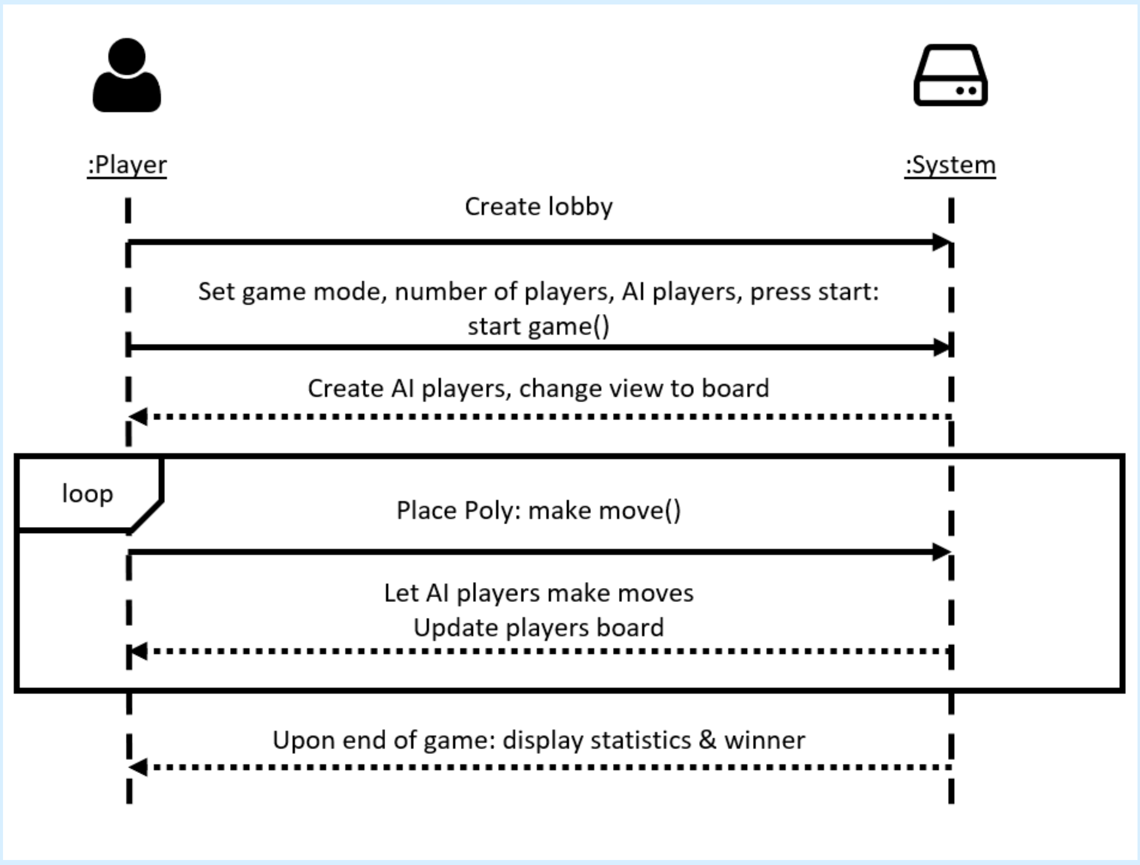
5.3. **Usage of remote accounts: Changing**


:Player
:System
Enter username of previously created account
Enter old password, hash old password before sending
Enter new password, hash new password before sending
Enter IP Address of Host
Look for user, if exists: check whether password is correct
response: success / error Message

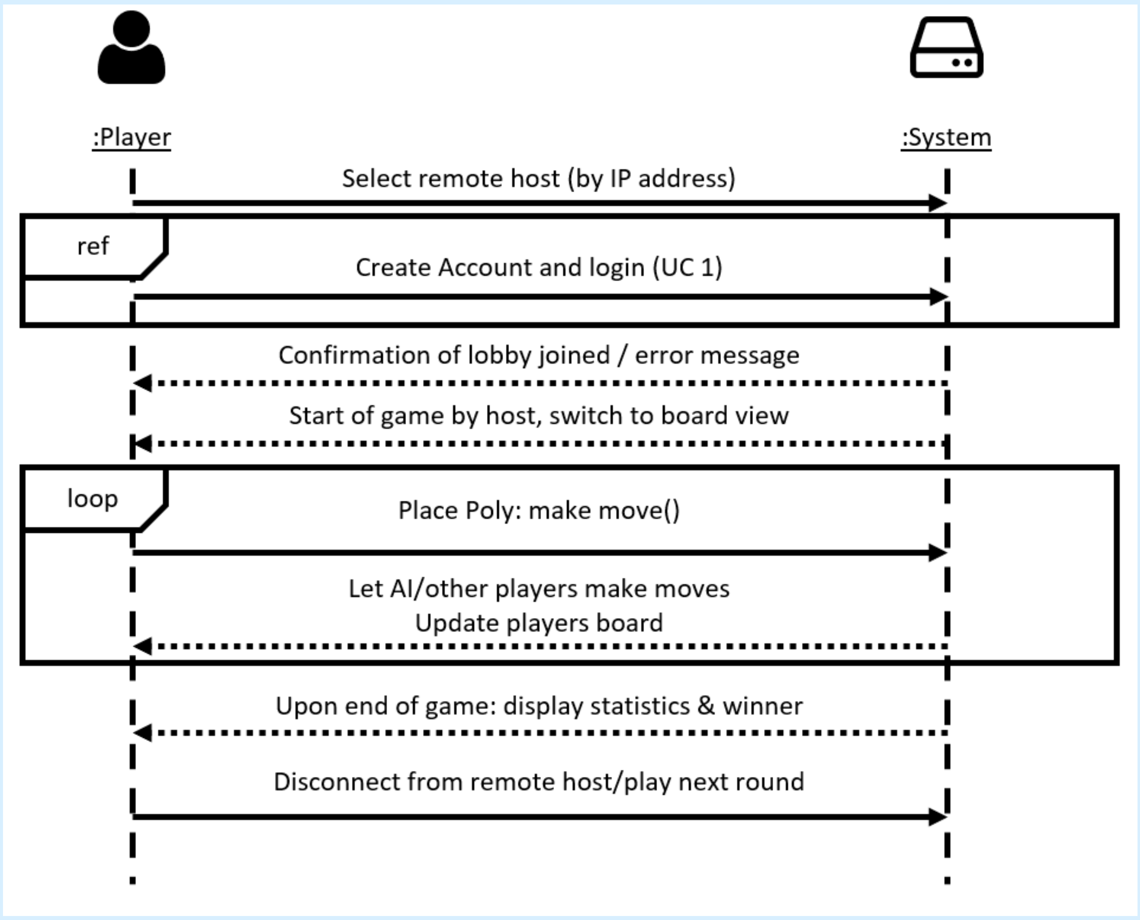
5.4. **Usage of remote accounts: Deletion**


:Player
:System
Enter username of previously created account
Enter password, hash password before sending
Enter IP Address of Host
Send deletion request: delete account()
Try to delete user
Send confirmation / error message

5.5. **Play local game**

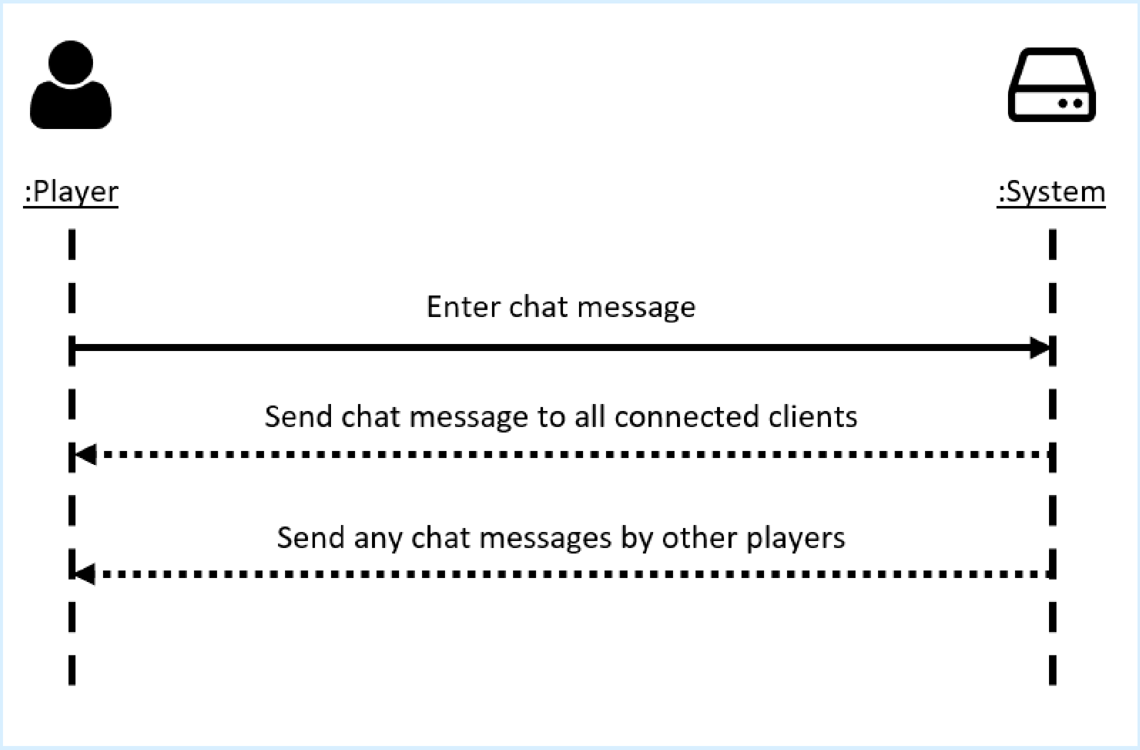

:Player
:System
Create lobby
Set game mode, number of players, AI players, press start:
start game()
Create AI players, change view to board
loop
Place Poly: make move()
Let AI players make moves
Update players board
Upon end of game: display statistics & winner

5.6. Play remote game



:Player
:System
Select remote host (by IP address)
ref
Create Account and login (UC 1)
Confirmation of lobby joined / error message
Start of game by host, switch to board view
loop
Place Poly: make move()
Let AI/other players make moves
Update players board
Upon end of game: display statistics & winner
Disconnect from remote host/play next round

5.7. Play remote game: Chat



:Player
:System
Enter chat message
Send chat message to all connected clients
Send any chat messages by other players

6. Operational Contracts

@author lbaudenb

Operation	deleteAccount()
Cross Reference	UC1: Manage Player Account
Precondition	Account exists.
Postcondition	Account was deleted. Confirmation Message was sent.

Operation	createAccount(username: String, password: String)
Cross Reference	UC1: Manage Player Account
Precondition	Account register resource is registered on host server. Server.DB instance was created.
Postcondition	Server.DB is updated and now contains username and password hash. Confirmation message was sent.

Operation	startGame ()
Cross Reference	UC2: Play
Precondition	A list of players exists A list of game variants exists All game variants allow for the number of players
Postcondition	A Game instance game was created game.running became true A GameVariant instance gameVariant was created gameVariant was associated with game Each player was associated with game

Operation	makeMove (poly : Poly , position : int[])
Cross Reference	UC2: Play
Precondition	A game instance is created The game is running
Postcondition	The board was updated The remaining polys of the current player were updated player.score was updated The next player was determined

7. Use Cases

@author tbuscher / lbaudenb

7.1. UC 1 - Use Cases – Manage Player Account

Primary Actor	Remote Human player
Stakeholders and interests	Remote Human Player wants to either create, edit or delete an account. Host Human Player expects remote players to be able to join him in his hosted Lobbies.
Preconditions	The Remote Human Player and the Host must be within one network that works reliably.
Postconditions	The account has been either created, changed or removed from the Host Human Players game.
Main Success Scenario	1. The remote Human Player launches the application, klicks "Play" and then "Join Lobby". 2. The Host Human Player also launches the Application and navigates to Host Lobby is shown the computers IP-Address. This information is relayed to the Remote Human Player. The Host Human player leaves the application open until the remote Human Player has completed his actions. 3. The Remote Human Player chooses one of the options from "Create account", "Edit account", "Delete account" and is presented with inputs. 4. The Remote Human Player enters all the necessary information, that always includes the IP Address of the Host Human Players computer. 5. Once the Remote Human Player clicks the button to go either create, edit, or delete the account the screen to join a lobby is shown. In case of the creation and change of an account, the IP-Address the account was registered on, and the username are shown in the appropriate inputs.
Extensions	5a The Remote Human Player had entered invalid credentials. An error message is shown for the input that contained. --> 3 5b The Remote Human Player had entered an existing username while trying to create an account. An error message is shown for the input username. --> 3

Special Requirements:

- Both human players need to be and remain connected to one common network.
- The Host has to open the required ports on his machine.

7.2. UC 1.1 - Select Theme

Primary Actor	Human player
Stakeholders and interests	Human player wants to select theme.
Preconditions	Human player has started the application.
Main Success Scenario	1. Human player is in main menu. 2. Human player clicks "Settings" 3. Human player selects theme. 4. Theme is updated.
Postconditions	Theme is updated.

Extensions	*a Application breaks down: 1. Application shows error message. 2. Application closes.

7.3. UC 2 – Play

Primary Actor	Human Player
Stakeholders and interests	Player wants to play the game of Blokus with a correct implementation of all rules and processes.
Preconditions	Human player has started the application and created a lobby.
Main Success Scenario	1. System selects human player. 2. Human Player chooses Poly out of his/her remaining Polys. 3. Human Player chooses rotation and if the Poly is mirrored. 4. Human Player chooses position on the board. 5. System checks if the selected poly is possible at the given position. 6. Poly is placed on the board at the chosen position. 7. System calculates the points scored in this move and adds them to the player's score. 8. System updates the remaining polys of the human player. 9. System selects next player. 10. Next player places poly on the board. Repeat steps 1-10 while the game is running. 11. System determines winner by choosing the player with the highest score. 12. Human player is back in the main menu or in the next round if multiple rounds have been selected.
Postconditions	Player played a complete game of Blokus. Player statistics were updated. Human player is back in the main menu.
Extensions	*a At any time, Application fails: 1. Game is not saved and cannot be restored. 2. Application shows error message. 3. Application closes. 3a Human Player added too many or too few AI Players for any game mode in the list: 1. Game shows message: Human Player can change game mode or add / remove AI player. 2. Human player changes game mode or add / remove AI player. 6a Human player has no edges left to place poly at: 1. The system informs the human player why he cannot place a poly. 2. Go to 12 .in the main success scenario.

	6b Human player skips turn: 1. System adds nothing to the player's score. 2. System selects next player 3. Go to 13. In the main success scenario. 10a System accepts poly at given position: 1. System informs human player that the poly can be placed. 2. Go to 9. in the main success scenario. 10b System rejects poly at given position: 1. System informs human player that the poly cannot be placed. 2. Human player is asked change his last turn. 3. Player changes his last turn. 4. Go to 8. in the main success scenario.
Special Requirements	Human player is informed why certain actions cannot be performed. Frames per Second should stay above 30.

User Stories:

- As a player I want to have a complete rulebook that prevents other players from cheating.
- As a player I want to skip a turn, if I do not find possible turns.
- As a player I want the UI to mark possible turns so that the gameplay is less frustrating.
- As a player I want to be informed why certain actions cannot be performed.
- As a player I want to see my statistics to track my improvements.
- As a player I want to change themes, so that the UI fits my preferences.

7.4. UC 2.1 : Create Game Session

Primary Actor	Host player
Stakeholders and interests	Host player wants to create a new game session.
Preconditions	Host player has started the application. For multiplayer host player must have a working internet connection.
Main Success Scenario	1. Host player is in the main menu. 2. Host player clicks "Play" 3. Host player hosts a local/multiplayer lobby. 4. Rounds with associated game variants are added by host player. 5. Remote AI players are added to reach maximum player count of game variant.
Postconditions	A game lobby was created. A game can be instantiated.
Extensions	*a Host closes game session: 1. Remote players are disconnected. 2. Remote players receive disconnect message. 3. Host player receives disconnect message. 4. Game session is deleted. 3a Host player hosts multiplayer lobby: 1. Remote human players join lobby. 2. Continue at 4. in main sucess scenario. 3b Host player hosts local lobby: 1. Host player sets AI difficulty 2. Host player adds AI remote player. 3. Continue at 4. in main success scenario.

Special Requirements	Host player must confirm disconnect.

7.5. UC 2.2 - Join Game

Primary Actor	Joining player
Stakeholders and interests	Joining player wants to join a game lobby to play blokus with other players in that lobby. Hosting player wants to host a game lobby and accept joining players.
Preconditions	Joining player has a working internet connection.
Main Success Scenario	5. Joining player has started the application. 6. Joining player is in the main menu. 7. Joining player selects "Play". 8. System switches to play menu. 9. Joining player selects "Join game". 10. System asks joining player to enter IP address. 11. Joining player enters IP address. 12. System asks joining player to enter username. 13. Joining player enters username. 14. System asks joining player to enter password. 15. Joining player enters password. 16. Joining player is connected to the game lobby.
Postconditions	Joining player has joined desired lobby and a game can be instantiated.
Extensions	*a Host closes the game or his/her computer shuts down: 1. Game is not saved and cannot be restored. 2. Joining player *a Remote Human Player closes the game or his/her computer shuts down: There is a certain amount of time for the player to reconnect. If the player does not re-establish a connection in time, his moves will be decided upon by an easy AI. 12a Lobby is full: 1. The system informs the joining player that the lobby he/her tries to join is full. 2. Go to 4. in the main success scenario.
Special Requirements	All human players need to be and remain connected to one common network. The Host must open the required ports on his machine.

7.6. UC 2.3 : Chat

Primary Actor	Chatting player
Stakeholders and interests	Chatting player wants to send text messages, so that the other players in the same lobby/game can see them. Chatting player wants to receive text messages from other players in the same lobby/game as him/her.
Preconditions	Chatting player has joined/hosted a lobby.

Main Success Scenario	1. Chatting player clicks on a text field embedded in the chat window. 2. Chatting player writes a message and presses Enter. 3. The text message is send to the server. 4. Server sends text message to all players. 5. Text message together with the time stance and username is visible to all players who received a message package from the server.
Postconditions	Text message together with time stance and username from chatting player is visible to all players.
Failed End Condition	Human player is back in Play menu.
Extensions	*a Host closes the game or his/her computer shuts down: Chat is not saved and can't be restored. *a Remote player loses connection to the server: All text messages sent from the remote player are still visible to the other players. If he/she reconnects in time he can participate in the chat again. 1. Remote player is disconnected. 2. Remote player is informed about lost connection. 1a Chat is not visible: 1. Remote player can set chat visibility on/off in game.
Special Requirements	Chat visibility can be turned off.

7.7. UC 2.4 - Leave Game

Primary Actor	Leaving player
Stakeholders and interests	Leaving player wants to leave the lobby he/she is currently part of. Other players want to be informed about any players leaving.
Preconditions	Leaving player has joined a lobby.
Main Success Scenario	1. Leaving player clicks "Leave Game". 2. Leaving player is disconnected. 3. Other players are informed about leaving player. 4. Leaving player is in the main menu.
Postconditions	Leaving player was disconnected. Leaving player is in the main menu.
Extensions	*a Host closes the game or his/her computer shuts down: Since the host closed the game, leaving player is disconnected anyway. 2a Error while leaving player disconnects: 1. Leaving player is disconnected anyway
Special Requirements	Leaving player must confirm his leave

8. UML Class Diagram

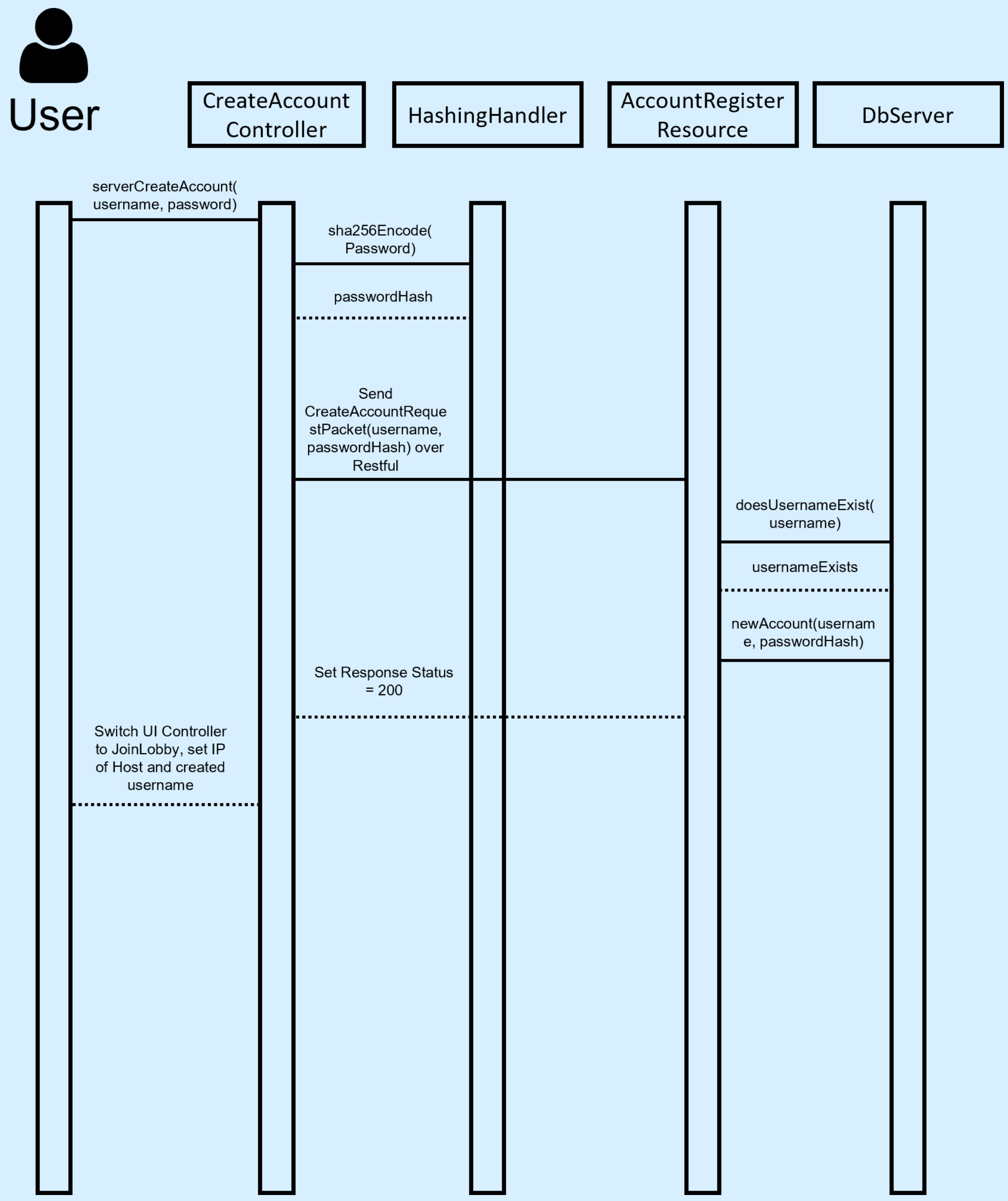
@author tiotto

Due to formatting the UML class diagram is stored in an extra file. Please look at the attached png.

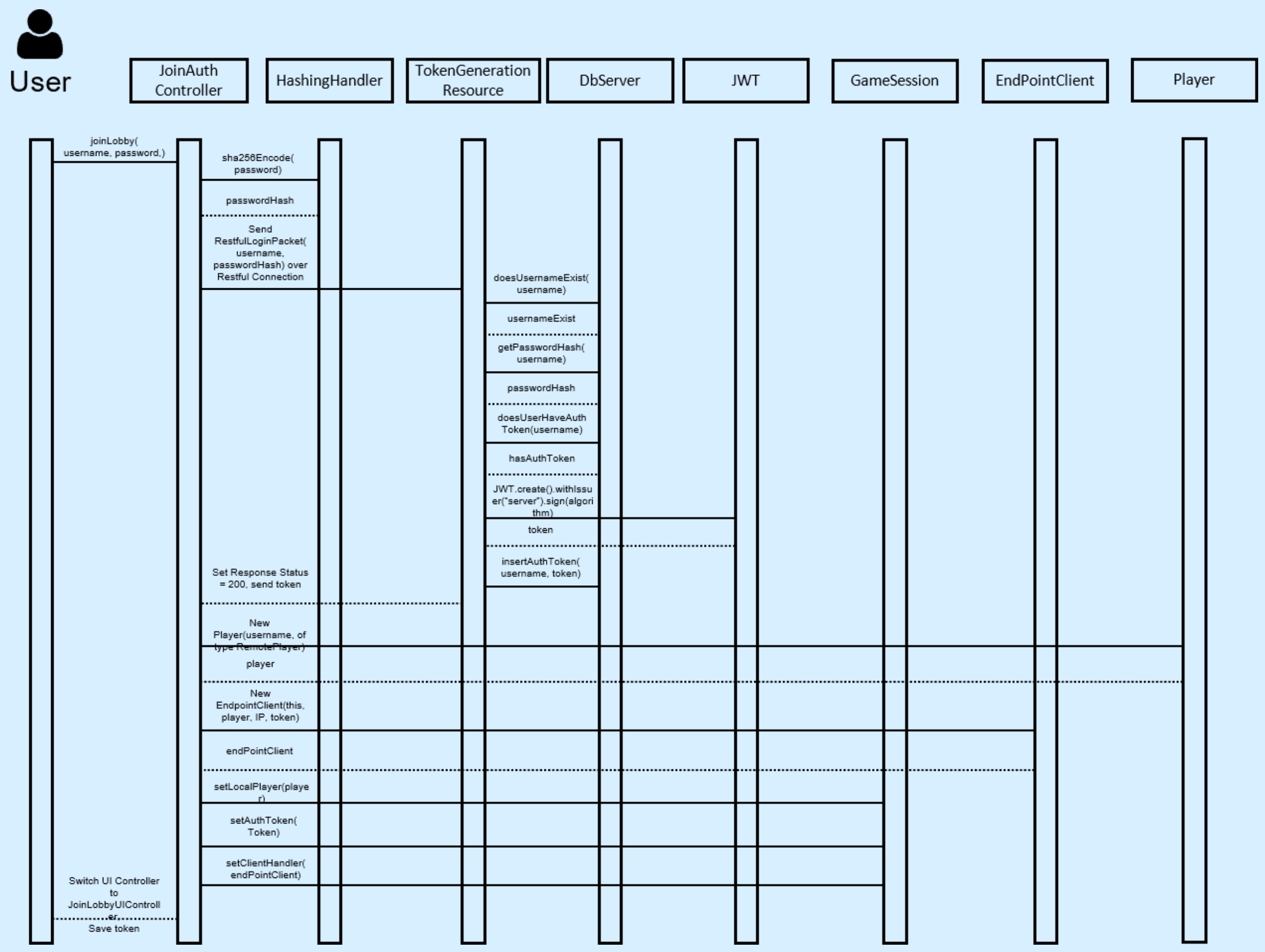
9. Sequence Diagram

@author tbuscher

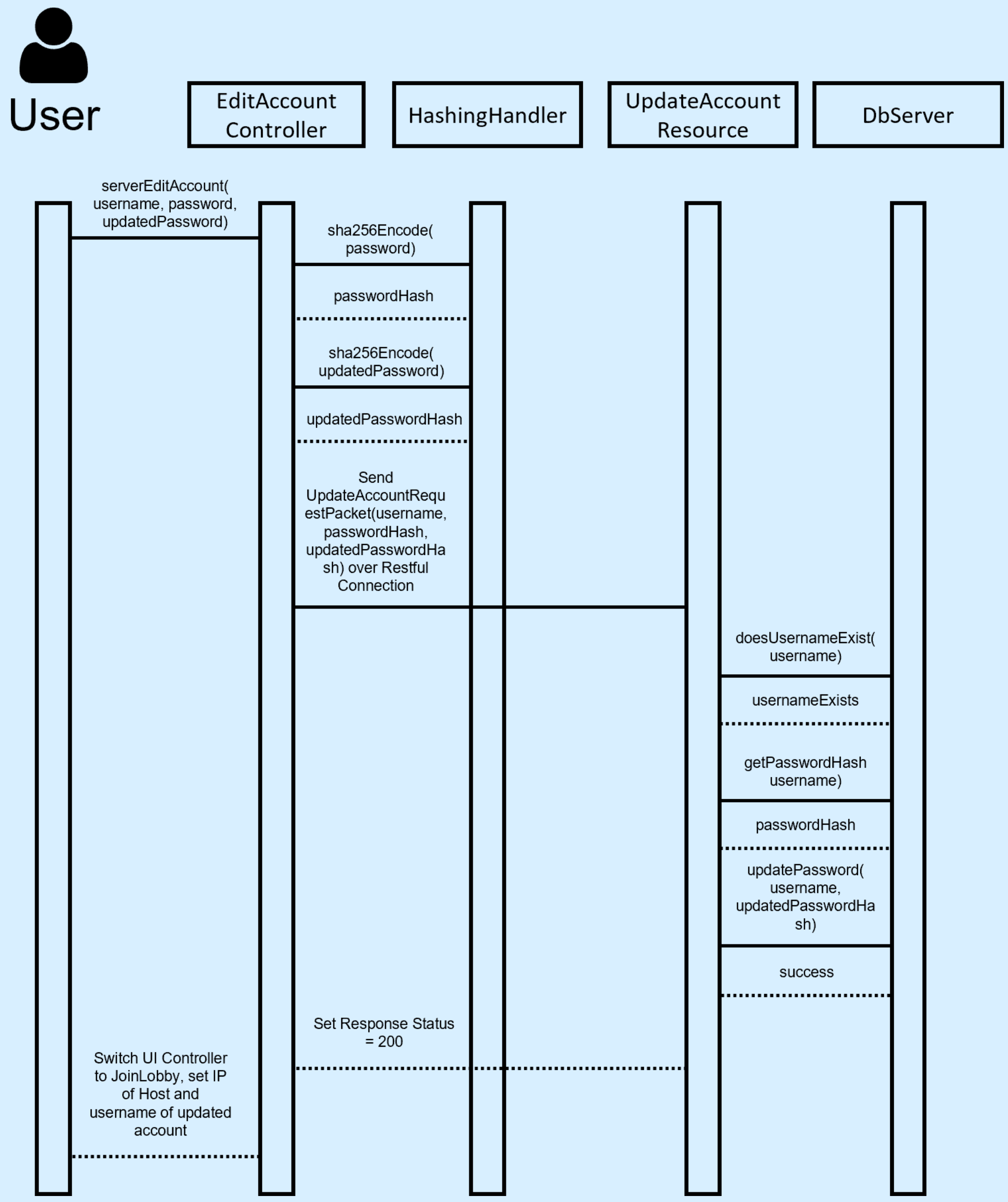
9.1. Create account


User
CreateAccount Controller
HashingHandler
AccountRegister Resource
DbServer
serverCreateAccount(username, password)
sha256Encode(Password)
passwordHash
Send CreateAccountRequestPacket(username, passwordHash) over Restful
doesUsernameExist(username)
usernameExists
newAccount(username, passwordHash)
Set Response Status = 200
Switch UI Controller to JoinLobby, set IP of Host and created username

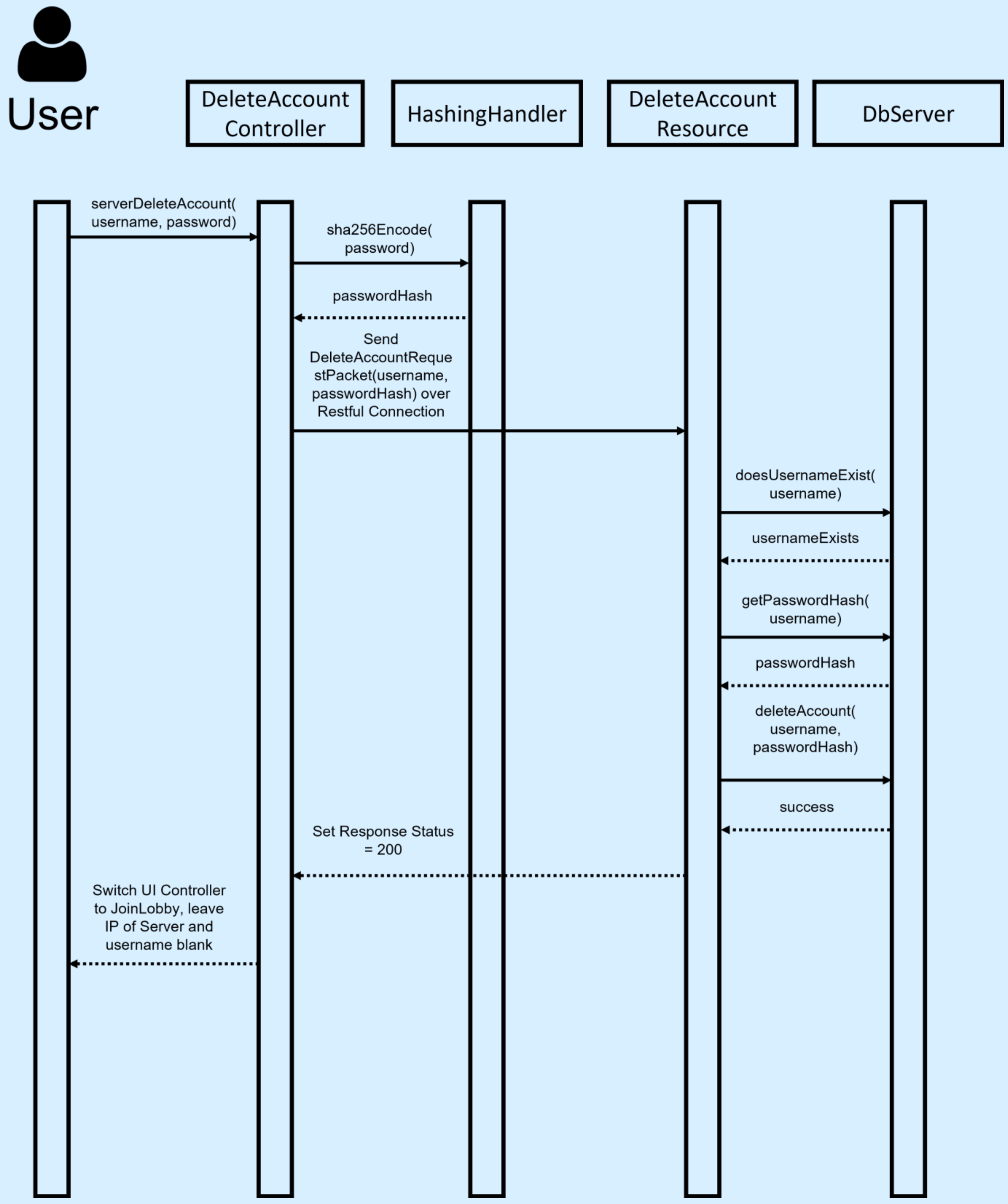
9.2. Login with account and prepare GameSession


User
JoinAuth Controller
HashingHandler
TokenGeneration Resource
DbServer
JWT
GameSession
EndPointClient
Player
joinLobby(username, password,)
sha256Encode(password)
passwordHash
Send RestfulLoginPacket(username, passwordHash) over Restful Connection
doesUsernameExist(username)
usernameExist
getPasswordHash(username)
passwordHash
doesUserHaveAuth Token(username)
hasAuthToken
JWT.create().withIssu er("server").sign(algori thm)
token
insertAuthToken(username, token)
Set Response Status = 200, send token
New Player(username, of type RemotePlayer)
player
New EndpointClient(this, player, IP, token)
endPointClient
setLocalPlayer(playe r)
setAuthToken(Token)
setClientHandler(endPointClient)
Switch UI Controller to JoinLobbyUIControll er
Save token

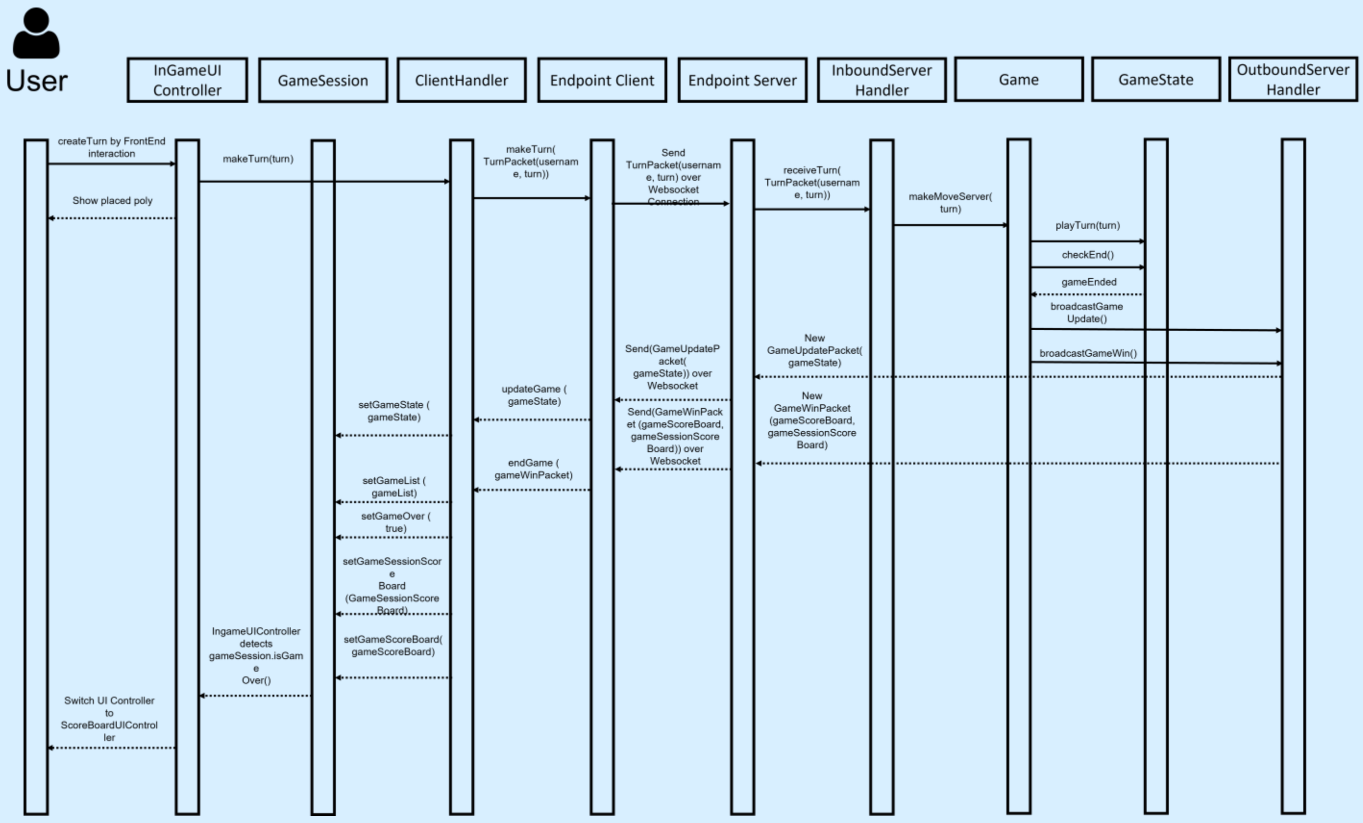
9.3. Update account


User
EditAccount Controller
HashingHandler
UpdateAccount Resource
DbServer
serverEditAccount(username, password, updatedPassword)
sha256Encode(password)
passwordHash
sha256Encode(updatedPassword)
updatedPasswordHash
Send UpdateAccountRequestPacket(username, passwordHash, updatedPasswordHash) over Restful Connection
doesUsernameExist(username)
usernameExists
getPasswordHash username)
passwordHash
updatePassword(username, updatedPasswordHash)
success
Set Response Status = 200
Switch UI Controller to JoinLobby, set IP of Host and username of updated account

9.4. **Delete account**


User
DeleteAccount Controller
HashingHandler
DeleteAccount Resource
DbServer
serverDeleteAccount(username, password)
sha256Encode(password)
passwordHash
Send DeleteAccountRequestPacket(username, passwordHash) over Restful Connection
doesUsernameExist(username)
usernameExists
getPasswordHash(username)
passwordHash
deleteAccount(username, passwordHash)
success
Set Response Status = 200
Switch UI Controller to JoinLobby, leave IP of Server and username blank

9.5. Game end


User
InGameUI Controller
GameSession
ClientHandler
Endpoint Client
Endpoint Server
InboundServer Handler
Game
GameState
OutboundServer Handler
createTurn by FrontEnd interaction
makeTurn(turn)
Show placed poly
makeTurn(TurnPacket(usernam e, turn))
Send TurnPacket(usernam e, turn) over Websocket Connection
receiveTurn(TurnPacket(usernam e, turn))
makeMoveServer(turn)
playTurn(turn)
checkEnd()
gameEnded
broadcastGame Update()
broadcastGameWin()
New GameUpdatePacket(gameState)
Send(GameUpdateP acket(gameState)) over Websocket
updateGame (gameState)
setGameState (gameState)
New GameWinPacket (gameScoreBoard, gameSessionScore Board)
Send(GameWinPack et (gameScoreBoard, gameSessionScore Board)) over Websocket
endGame (gameWinPacket)
setGameList (gameList)
setGameOver (true)
setGameSessionScor e Board (GameSessionScore Board)
setGameScoreBoard(gameScoreBoard)
IngameUIController detects gameSession.isGam e Over()
Switch UI Controller to ScoreBoardUIControl ler

10. Developer Manual

@author tiotto

Our program aims to realize the board game Blokus as a video game. We divided our code into three parts, which work with each other.

First the **engine package** contains mainly the controller of game and the user interface (UI) as well as different handler to manage parts of the program like input, error messages or other threads.

Second the **game package** contains the implementation of Blokus itself. It is divided itself according to the following sub-packages:

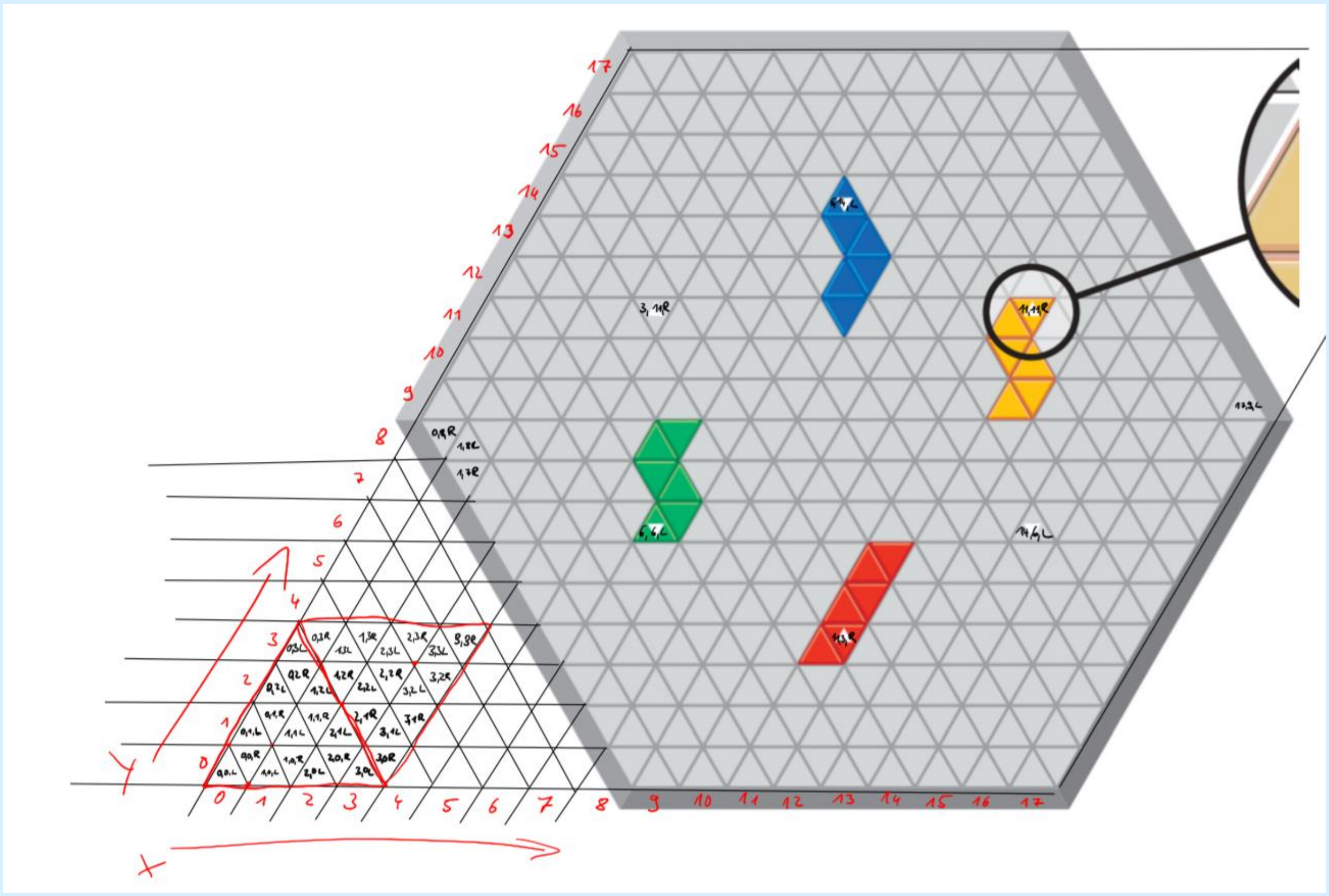
- The config sub-package contains the configuration data of the game and is stored beyond the game.
- The controller sub-package provides all controller that are shown as part of the user interface. Especially important is the InGameUiController, which manages the in-game gameplay of our program. There all objects of the package game.view are handled.
- The core sub-package provides the main method of our Bloks3 game and initializes the game by starting the game controller.
- The model sub-package is the actual implementation of the domain layer. Included are all classes for the game logic and the AI like the "Board", one single "Field" of the board or one polygon "Poly". With those the basic game is implemented. A field is the atomar piece of the game out of which the polys as well as the boards are built. Each of those classes are abstract and have a square as well as a trigon implementation. One "Game" is representing one single game, where one game can be initialized, started and moves can be played. In every game the "GameState" implements the current state of the game containing attributes like the board or the remaining polys for every player. Every game is included in a "GameSession" which contains all objects and methods needed for a whole game lobby. This includes among the game itself the chat or the joined players.
- The scores sub-package contains all score boards as well as methods to calculate the different statistics.
- The view sub-package implements all visible objects needed to show the in-game user interface with classes like panes for boards, polys and stacks as well as draggable panes for placing the polys by the human player.

The **net package** implements the network of Bloks3. It also provides functions only used in conjunction with the network, such as hashing and token generation.

- The auth sub-package provides all resources users access when creating, updating, deleting and using their accounts.
- The packet sub-package contains all packets send over the network. It is further split into:
  - abstr: here are the classes needed in the transmission of packets
  - account: any packets related to the handling of remote accounts
  - chat: the packet containing a chat-message
  - game: packets that contain information about certain events in a game
- The server sub-package contains the classes enabling network connections as well as the Database Server and classes that contain functions called by the server.

- The transmission sub-package contains the endpoints that are used when sending packets as well as an Encoder & Decoder for the Class packet that is send over the network.

To expand this general overview two aspects will be presented more detailed.
While the square board is implemented easily by a grid with rows and columns the **trigon board logic** is much more complicated. To get a suitable model we put the hexagon in a parallelogram so that a grid is possible again to refer to small parallelograms. Each of these small parallelograms consist out of two triangles we acutely want to address, so we introduce the variable isRight as an integer. For the left triangle the integer is zero, while for the right triangle it is one. The trigon polys are all represented by fields within the parallelogram out of fields from x and y from 0 to 3. Rotation and mirroring needed to be implemented manually by picturing each field on the corresponding one.



The **artificial intelligence** has four different difficulty modes. Behind that there are five basic algorithms. The first generates just a random possible move. The next three are implemented through sorting all the possible moves by different parameters. The probably most common one is size, so that the biggest polys will be played first. The second parameter is the number of fields that are blocked by the turn, where opponents could put polys in future turns. With the third parameter the possible turns are sorted by the number of rows and columns that are newly occupied and aims therefor at the beginning to place polys towards the middle as fast as possible. The five algorithm generates a new turn using the monte carlo tree search (MCTS). That means random possible turn sequences will be generated and measured, what turns are more likely to get a higher score than others.
With these algorithms the different difficulties are built. The easy AI chooses 80 % of the time a random move while in the other 20 % it chooses one of the biggest polys left. The middle AI chooses all the time after the size of the polys, while the hard AI chooses always the turn, which blocks the

most fields for opponents. The godlike AI varies its strategy within the game. First, depending on the game mode how many turns, the AI will try to get to the middle as fast as possible and spread over the board. In the second stage the godlike AI plays similar to the hard AI trying to block the opponents while in the last few turns the most promising moves will be chosen regarding the score due to using the MCTS.

11. User Manual

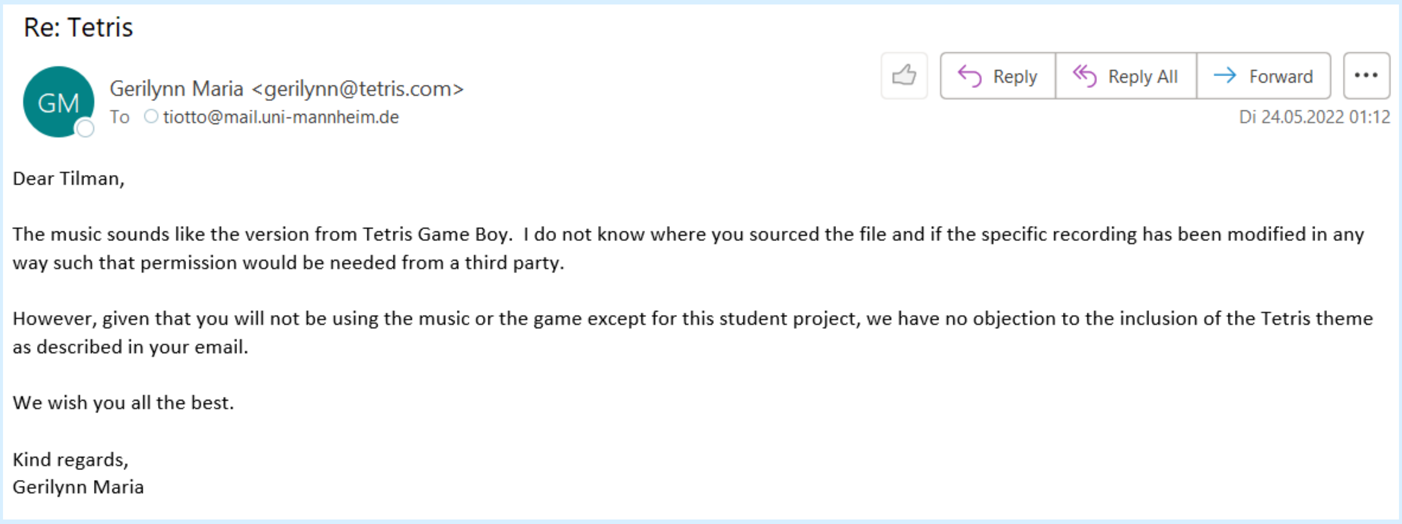
@author tiotto

See here or in the extra document in the folder.

12. Legal Licensing

@author tiotto

We use the Tetris Gameboy music as background music. Therefor we got the permission from Tetris although we downloaded the actual file not from Tetris itself:



Re: Tetris

Gerilynn Maria <gerilynn@tetris.com>
To tiotto@mail.uni-mannheim.de

Di 24.05.2022 01:12

Dear Tilman,

The music sounds like the version from Tetris Game Boy. I do not know where you sourced the file and if the specific recording has been modified in any way such that permission would be needed from a third party.

However, given that you will not be using the music or the game except for this student project, we have no objection to the inclusion of the Tetris theme as described in your email.

We wish you all the best.

Kind regards,
Gerilynn Maria