SCREW YOUR NEIGHBOR

Report for Milestone 2 of Sopra FS22, Group 36

Figure 1: cover (source B. Furrer)

Members

Carmen Kirchdorfer (20-720-132)
Salome Wildermuth (10-289-544)
Beat Furrer (07-542-392)
Lucius Bachmann (11-060-274)
Moris Camporesi (19-764-349)

Diagrams

Component Diagram

The Rapid Api Development Architecture provided by spring-data-rest will be used to develop the api. The PagingAndSortingRepositories provide request handling, deserialization, crud on the database, serialization and rendering of errors. Validation will be implemented with Bean validations and if necessary with Spring event handlers. Side effects will also be done with Spring event handlers.

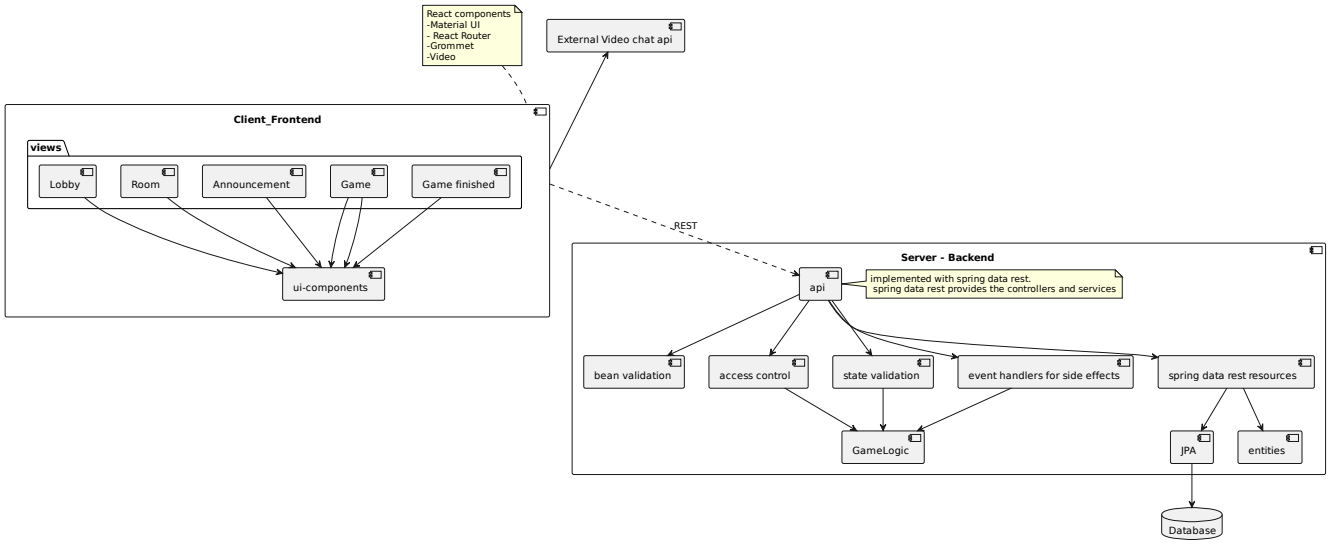

Figure 2: Component diagram

Class Diagram

The class diagram only depicts the domain classes. The classes for validation, the repositories etc. are left out that the diagram stays simple.

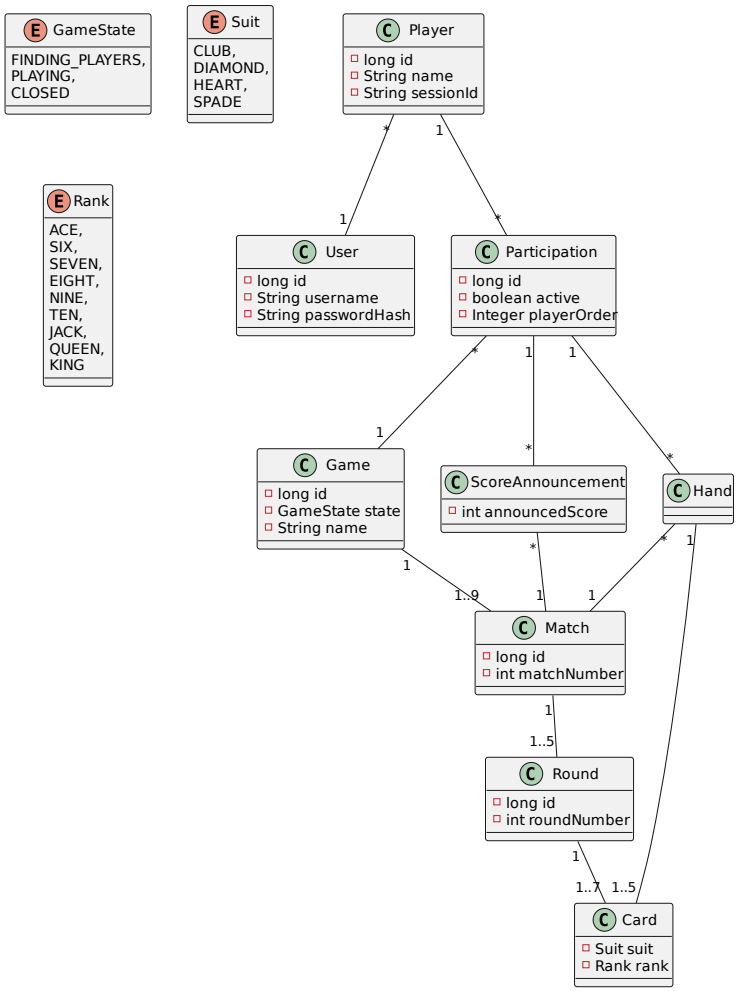

Figure 3: Class diagram

Activity Diagram

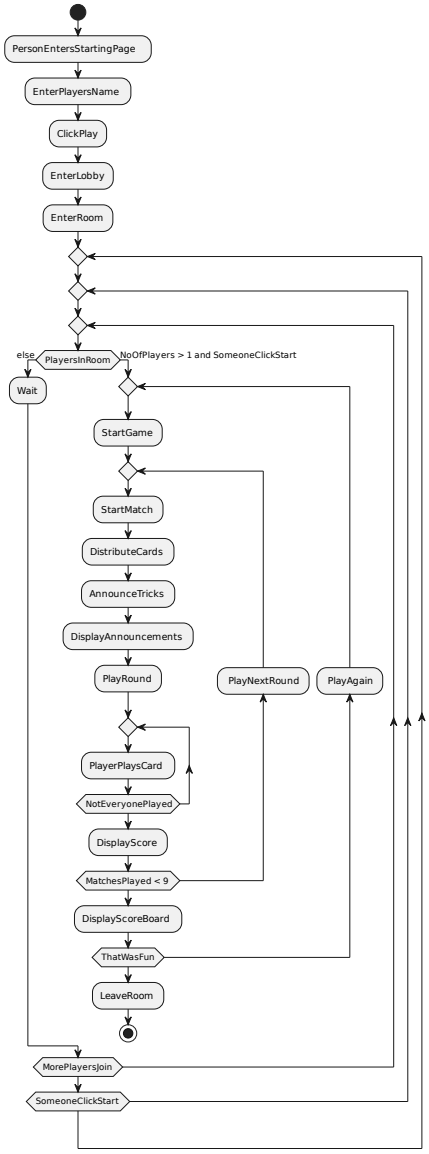

Figure 4: Activity diagram

UI Mockups

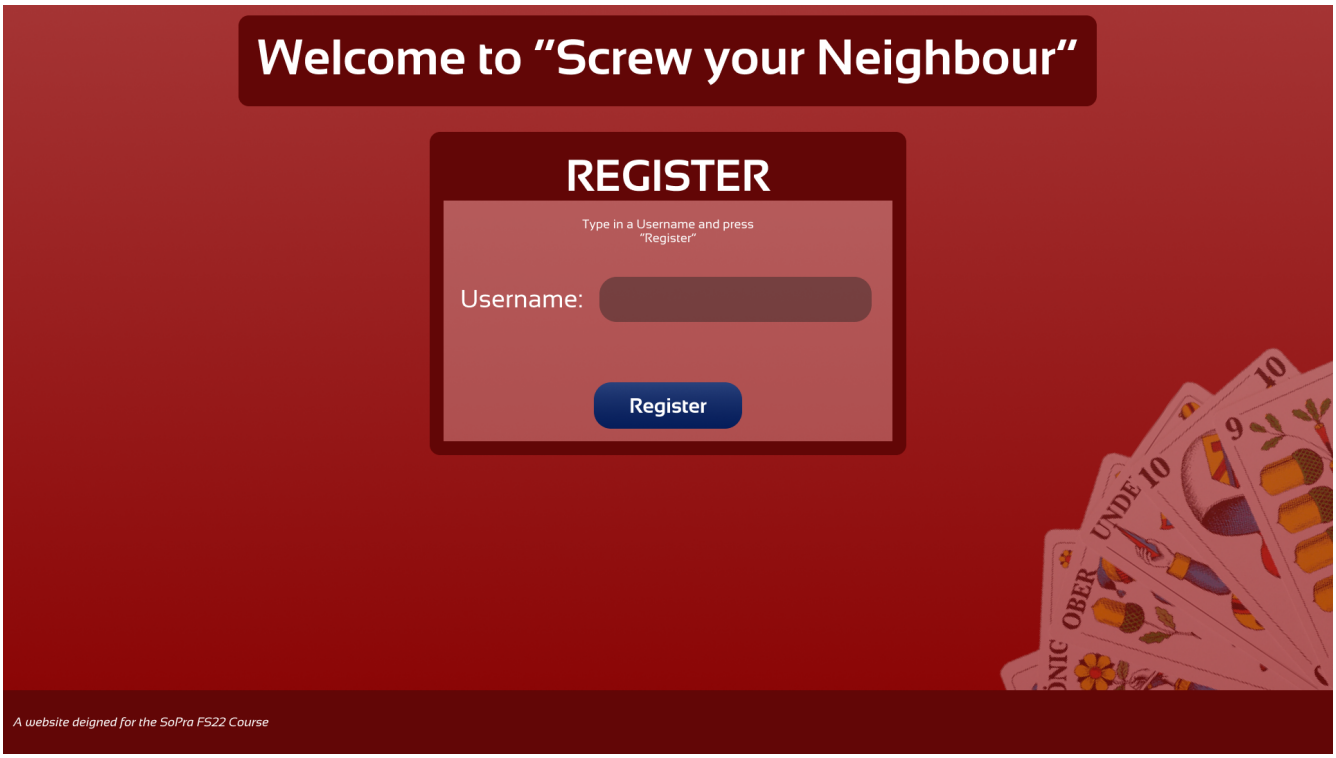

Figure 5: Landing Page

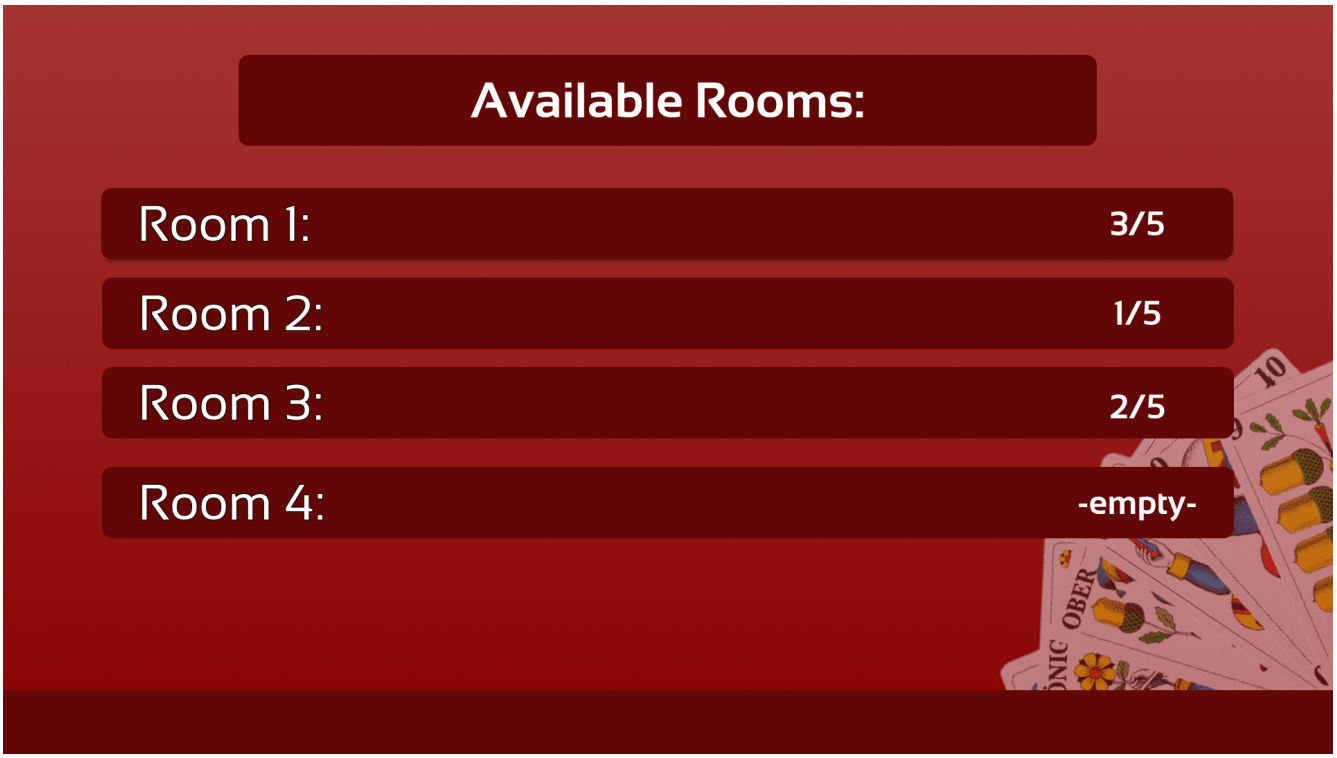

Figure 6: Lobby

Figure 7: Room

Figure 8: Trick announcement

Figure 9: Game

Figure 10: Game when it's your turn

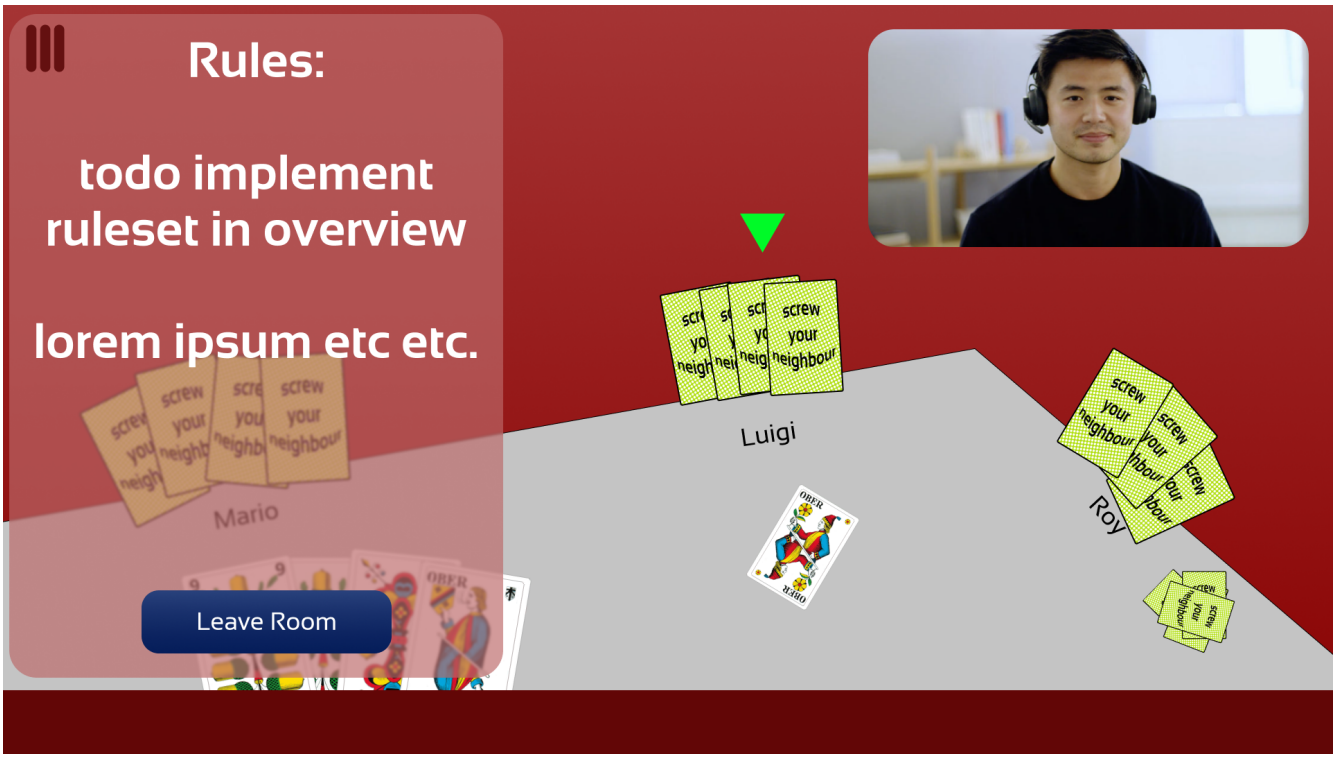

Figure 11: Display rules

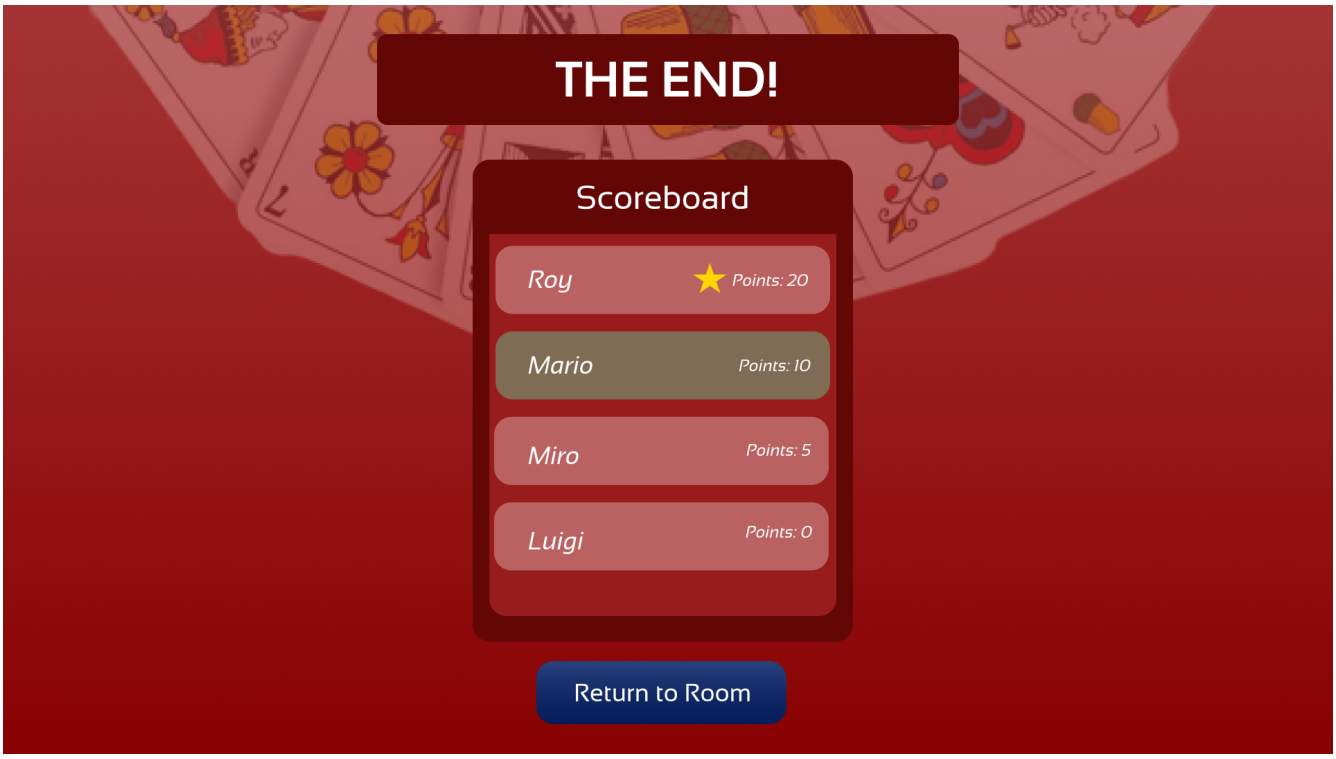

Figure 12: Game End

Rest Specification

Description of the notation

How the endpoints specification is structured:

Endpoint /endpoints

Model:

Modelname

- PropType1 propName1
- PropType2 propName2

Method	Description
POST	Post Request to /endpoints
GET	Get Request to /endpoints (GET collection)
GET /{id}	Get Request to /endpoints/{id} (GET item)
PATCH /{id}	PATCH Request to /endpoints/{id}
DELETE /{id}	DELETE Request to /endpoints/{id}

We don't use the PUT request, because all functionality we need of a PUT request can be done with a PATCH request, and the PATCH request is easier.

All endpoints support at least:

- Accept: application/hal+json
- Content-Type: application/hal+json (POST, PATCH)

The parameter types are as follows:

- {id}: Patch parameter
- Model: Body parameter (as application/hal+json)
- ?sort= and ?propName1= filters : Query Parameter

Endpoints

Endpoint /players

Model:

Player

- int id
- String name
- String sessionID

Method	Response Codes	Description
POST	201 400 422 (Player for this session already exists)	Creates a session and a player for this session
GET	200	
GET /{id}	200 403 (Not allowed to see player) 404	
PATCH /{id}	200 400 404 422 403 (Not allowed to patch other player)	Set player name

Side effect: creates a player session so that all the player's further requests can be identified.

Endpoint /games

Model:

Game

- int id
- String name
- Enum state
- Collection players (embedded)
- Collection matches (embedded)

Method	Response Codes	Description
POST	201 400 422 (Cannot create a room when already in a room)	Creates a new game room
GET	200	
GET /{id}	200 404 403 (Not allowed to see room)	
PATCH /{id}	200 400 404 422 403 (Not allowed to patch game you are not in)	Update game name
DELETE /{id}	204 403 (Not allowed to delete game you are not in) 404 422 (not allowed to delete game with players)	Delete the game

Side effects: If state is patched to playing, then a new match is created and a hand for each player is created.

Endpoint /participations

Model:

Participation

- int id
- boolean active
- int playerOrder
- Player player (embedded)
- Game game (embedded)

Method	Response Codes	Description
POST	201 400 404 Game not Found 422 (Game closed)	Enter game as player
PATCH /{id}	200 400 403 (Not allowed to patch the participation of another player) 404 422 (update only allowed to inactive)	Mark as inactive

Endpoint /matches

Model:

Match

- int id
- int matchNumber
- int numberOfPlayedCards (calculated property)
- Collection scoreAnnouncements (embedded)
- Collection rounds (embedded)
- Collection hands (embedded)

Hand

- Participation participation
- Collection card (embedded)

Card

- int id
- Suit? suit (may be null if the card is hidden)
- Rank? rank (may be null if the card is hidden)

Method	Response Codes
GET /{id}	200 403 (Not allowed to see match) 404 (Match with id not found)

Endpoint /scoreAnnouncements

Model:

ScoreAnnouncement

- int id
- int announcedScore
- Participation participation
- Match match

Method	Response Codes	Description
POST	200 403 (Not your turn, player not in match) 422 (invalid score)	Announce score for match

Side Effects: if all ScoreAnnouncement are made, a new round is created.

Endpoint /rounds

Model:

Round

- int id
- int roundNumber
- Player winner
- Collection turns

Method	Response Codes
GET	200 403
GET /{id}	200 403 (Not allowed to see round) 404 (Round with id not found)

Endpoint /turns

Model:

Turn

- int id
- Round round
- Card card

Method	Response Codes	Description
POST	200 403 (Not your turn, player not in match) 404 422 (invalid card)	Announce score for match

If the last turn of a round is made, then the round is ended. If it was the last round of the last match, the game is finished.

The Endpoints to create a user, to log in and log out are left out. We don't know yet if we will implement them.