Softwarekonzept des Kartenspiels *Mau-Mau*

## Modul: Komponentenbasierte Entwicklung

## Gruppe 6: Jasmin Quast, Maria Hallmann, Philipp Schalau, Richard Frost

Inhalt

[1 Komponentenschnitt 3](#_Toc110079651)

[2 Schnittstellenbeschreibung 4](#_Toc110079652)

[2.1 Interface AppController 4](#_Toc110079653)

[2.2 Interface ViewService 5](#_Toc110079656)

[2.3 Interface CardService 9](#_Toc110079677)

[2.4 Interface DeckService 10](#_Toc110079682)

[2.5 Interface GameService 11](#_Toc110079687)

[2.6 Interface PlayerService 15](#_Toc110079703)

[2.7 Interface RuleService 16](#_Toc110079709)

[2.8 Interface AIService 19](#_Toc110079719)

[3 Konzeptionelles Datenmodell 21](#_Toc110079726)

[4 Präsentationsschicht 22](#_Toc110079727)

[4.1 Spielbeginn 22](#_Toc110079728)

[4.2 Spielablauf 23](#_Toc110079729)

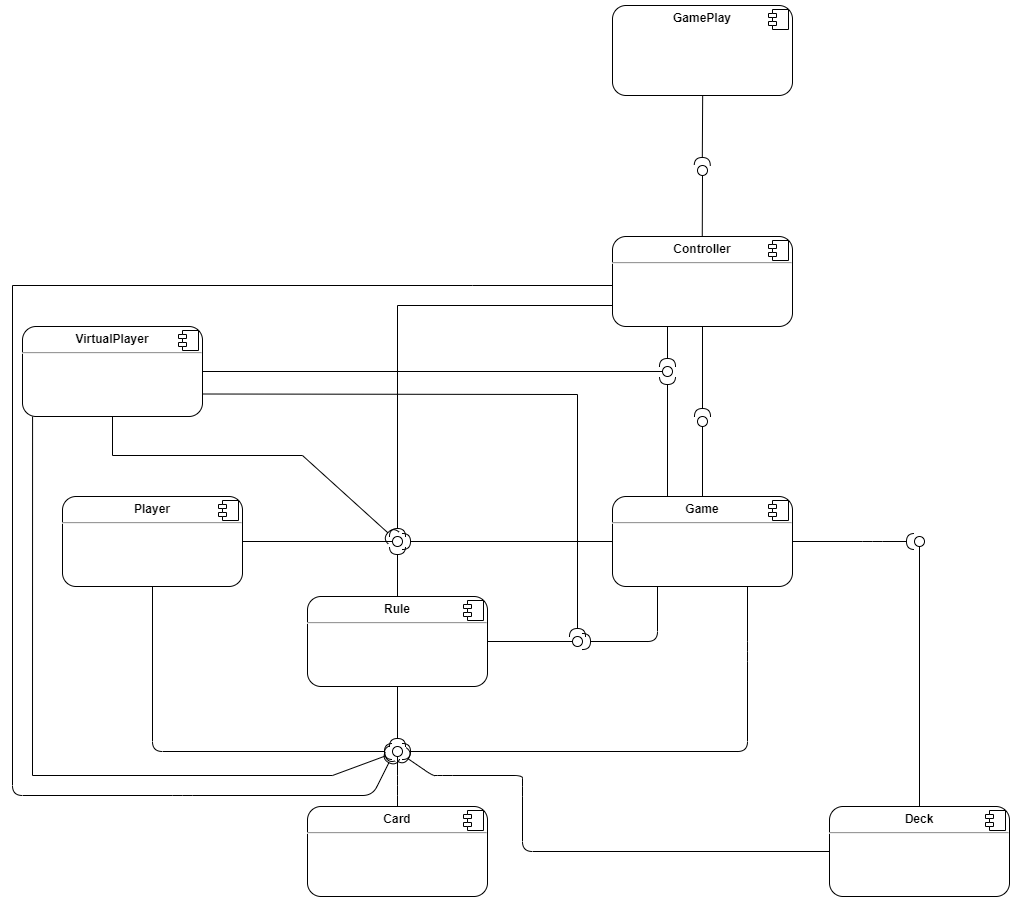
[4.3 Spielende 24](#_Toc110079730)

[4.4 Fehlerausgaben 25](#_Toc110079731)

[5 Frameworks 26](#_Toc110079732)

[6 Ablaufumgebung 30](#_Toc110079733)

# Komponentenschnitt



Unsere Software ist ein Multimodulprojekt. Sie besteht aus insgesamt 8 Modulen:

***Player***

Die Komponente *Player* erstellt neue Spieler und verwaltet ihre Handkarten. Sie stellt eine Schnittstelle zur Verfügung, die von den *Komponenten* Rule, *Game*, *VirtualPlayer* und *Controller* benutzt wird.

***Virtual Player***

Die Komponente *VirtualPlayer* legt das Verhalten von KI Spielern fest. Sie stellt eine Schnittstelle zur Verfügung, die von den Komponenten *Game* und *Controller* benutzt wird.

***Card***

Die Komponente *Card* erstellt einzelne Karten (bestehend aus zwei Enums) und Kartenlisten. Ihre Schnittstelle wird von den Komponenten *Deck, Game, Player, VirtualPlayer, Rule* und *Controller* genutzt.

***Deck***

Die Komponente *Deck* verwaltet die Kartenliste und stellt somit das Kartendeck des Spiels zur Verfügung, die sie von der Komponente *Card* erhält. Es werden Karten herausgegeben und hinzugefügt. Die Komponente *Game* nutzt die Schnittstelle von Deck.

***Rule***

Die Komponente *Rule* prüft die Anwendung von Spielregeln und ob gespielte Karten zulässig sind. Ihre Schnittstelle wird ebenfalls von der Komponente *Game* und *VirtualPlayer* benutzt.

***Game***

Die Komponente *Game* vereint Spieler und das Kartendeck zu einem neuem Gameobjekt. Zusätzlich werden gespielte oder zu ziehende Karten weitergereicht an Spieler und Deck. Außerdem behandelt sie die Ergebnisse aus der Komponente Rule und kümmert sich um die Konsequenzen für anzuwendende Spielregeln für den Spieler oder das Spiel generell. Zusätzlich befindet sich innerhalb dieser Komponente das *Data Access Object (DAO)*. Die Komponente *Controller* greift auf diese Schnittstelle zu.

***Controller***

Der *Controller* besteht aus dem Appcontroller und der *View*. Die View übernimmt alle sichtbaren Ausgaben des Spiels an den Spieler und nimmt ebenfalls die Eingaben des Spielers entgegen. Der Appcontroller übernimmt den gesamten Spielablauf und geht mit der Auswahl, die der Spieler der View übermittelt hat, entsprechend um. Rundenbasiert wird gespielt, bis ein Spieler keine Karten mehr auf der Hand besitzt.

Hinzukommend können Spiele gespeichert und geladen werden. Die Schnittstelle der *Controller* Komponente wird von *GamePlay* verwendet.

***GamePlay***

Diese Komponente beinhalt die Mainmethode und startet das gesamte Spiel.

# Schnittstellenbeschreibung

Die unten aufgeführten Tabellen zeigen alle Methoden, die zur jeweiligen Schnittstelle gehören und was sie leisten. Das JavaDoc befindet sich in unserem Github Respository (<https://github.com/RiFrost/mau-mau>) unter: docs/JavaDoc

## Interface AppController

Die Klasse *GamePlay* importiert das Interface *AppController*.

|  |
| --- |
| ***Method Details***  **play**  void play()  runs the entire game |

## Interface ViewService

Der *ViewService* wird von der Klasse *AppControlerImpl* benutzt.

|  |
| --- |
| ***Method Details*****showWelcomeMessage** void showWelcomeMessage()  shows welcome message **getNumberOfPlayer** int getNumberOfPlayer()  shows gaming instructions asks the player for the desired number of players  **Returns:**  number of players **getNumberOfAI** int getNumberOfAI(int totalNumPlayers) **getPlayerNames** [List](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/List.html)<[String](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/String.html)> getPlayerNames(int numberOfPlayer)  asks for player names depending on the number of players  **Parameters:**  numberOfPlayer - number of players in the game  **Returns:**  list of player names **showStartGameMessage** void showStartGameMessage(long id)  lets the player know that the game has started  **Parameters:**  id - game id |

|  |
| --- |
| **showTopCard** void showTopCard([Card](file:///C:\\Users\\card\\export\\Card.html" \o "class in htw.kbe.maumau.card.export) topCard)  lets the player know which card is on top of the pile  **Parameters:**  topCard - card that is in top of the discard pile **showHandCards** void showHandCards([Player](file:///C:\Users\player\export\Player.html) player, [Suit](file:///C:\Users\card\export\Suit.html) suit)  shows the player his hand cards and if there is a suit wish, it is also shown  **Parameters:**  player - player who is in turn  suit - suit that was requested (optional) **getPlayedCard** [Card](file:///C:\Users\card\export\Card.html) getPlayedCard([Player](file:///C:\\Users\\player\\export\\Player.html" \o "class in htw.kbe.maumau.player.export) player)  asks the player which card he wants to discard or if he wants to draw card(s)  **Parameters:**  player - player who is in turn  **Returns:**  card chosen card that the player wants to play **getChosenSuit** [Suit](file:///C:\Users\card\export\Suit.html) getChosenSuit([Player](file:///C:\Users\player\export\Player.html) player, [List](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/List.html)<[Suit](file:///C:\Users\card\export\Suit.html)> suits)  asks the player for the suit wish  **Parameters:**  player - player who is in turn  suits - list of suits to choose from  **Returns:**  desired suit wish **saidMau** boolean saidMau([Player](file:///C:\\Users\\player\\export\\Player.html" \o "class in htw.kbe.maumau.player.export) player)  asks the player if he wants say 'mau'  **Parameters:**  player - player who is in turn  **Returns:**  true if player said 'mau', false if not **showPlayedCard** void showPlayedCard([Player](../../player/export/Player.html) player, [Card](../../card/export/Card.html) card)  shows played card from player  **Parameters:**  player - player who is in turn  card - card that was played **showDirectionOfRotation** void showDirectionOfRotation(boolean isClockwise)  shows the game direction when changed  **Parameters:**  isClockwise - true if direction is clockwise, false if counterclockwise **showDrawnCardMessage** void showDrawnCardMessage([Player](file:///C:\\Users\\player\\export\\Player.html" \o "class in htw.kbe.maumau.player.export) player, int numberOfDrawnCards)  lets the player know how many cards he has drawn  **Parameters:**  player - player who is in turn  numberOfDrawnCards - number of cards to be drawn **showAiPlayedCardMessage** void showAiPlayedCardMessage([Player](file:///C:\Users\player\export\Player.html) player, [Card](file:///C:\Users\card\export\Card.html) card)  lets the player know which card was played by AI  **Parameters:**  player - player(AI) who is in turn  card - card which was played **showPlayersMau** void showPlayersMau([Player](file:///C:\\Users\\player\\export\\Player.html" \o "class in htw.kbe.maumau.player.export) player)  shows the player that AI has one card left and therefore said mau  **Parameters:**  player - player(AI) who said mau **showErrorMessage** void showErrorMessage([String](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/String.html) exceptionMessage)  lets the player know want went wrong in the game  **Parameters:**  exceptionMessage - message that was thrown by exception **showWinnerMessage** void showWinnerMessage([Player](file:///C:\\Users\\player\\export\\Player.html" \o "class in htw.kbe.maumau.player.export) player)  lets players know who has won  **Parameters:**  player - player who won **hasNextRound** boolean hasNextRound()  asks for next round to play  **Returns:**  true if player wants to play again, false if game should quit **playerWantsToLoadGame** boolean playerWantsToLoadGame()  asks the player if he wants to load a game  **Returns:**  true, if player wants to load a game, false if not **getGameId** long getGameId()  returns the game id for the game to be loaded  **Returns:**  long id belonging to the loading game |

## Interface CardService

Der Service wird von den Klassen DeckServiceImpl und GameServiceImpl importiert*.*

|  |
| --- |
| ***Method Details*****getSuits** [List](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/List.html)<[Suit](file:///C:\Users\MariaH\Desktop\Suit.html)> getSuits()  all available Suits are put into a list  **Returns:**  List of Suits **getLabels** [List](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/List.html)<[Label](file:///C:\Users\MariaH\Desktop\Label.html)> getLabels()  all available Labels are put into a list  **Returns:**  List of Labels getCards [List](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/List.html)<[Card](file:///C:\Users\MariaH\Desktop\Card.html)> getCards()  32 cards are initialized. The card stack consists of 4 suits with 8 labels each.  **Returns:**  List of Cards |

## Interface DeckService

*DeckService* wird von *GameServiceImpl* importiert.

|  |
| --- |
| ***Method Details*****createDeck** [Deck](file:///C:\Users\MariaH\Desktop\Deck.html) createDeck([List](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/List.html)<[Card](file:///C:\Users\card\export\Card.html)> cards) throws [IllegalDeckSizeException](file:///C:\Users\MariaH\exceptions\IllegalDeckSizeException.html)  initialise card deck  **Parameters:**  cards - - Card stack that has to add to deck  **Returns:**  deck that include the required card stack  **Throws:**  [IllegalDeckSizeException](file:///C:\Users\MariaH\exceptions\IllegalDeckSizeException.html) - when card stack is empty or has an invalid Suit Label ratio **initialCardDealing** [List](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/List.html)<[Card](file:///C:\Users\card\export\Card.html)> initialCardDealing([Deck](file:///C:\Users\MariaH\Desktop\Deck.html) deck)  at the beginning of a game: depending on the initial number of drawn cards, cards are dealt from the draw pile  **Parameters:**  deck - - card deck  **Returns:**  initial cards for player **getCardsFromDrawPile** [List](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/List.html)<[Card](file:///C:\Users\card\export\Card.html)> getCardsFromDrawPile([Deck](file:///C:\Users\MariaH\Desktop\Deck.html) deck, int numberOfDrawCards)  depending on the number of cards to be drawn, the cards are dealt from the draw pile  **Parameters:**  deck - - card deck  numberOfDrawCards - - number of cards to be drawn  **Returns:**  card list of drawn cards |

## Interface GameService

Das Interface *GameService* wird von der Klasse *AppControllerImpl* importiert.

|  |
| --- |
| ***Method Details*****createGame** [Game](file:///C:\Users\MariaH\Desktop\Game.html) createGame([List](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/List.html)<[Player](file:///C:\Users\player\export\Player.html)> players) throws [IllegalDeckSizeException](file:///C:\Users\deck\exceptions\IllegalDeckSizeException.html), [InvalidPlayerSizeException](file:///C:\Users\MariaH\exceptions\InvalidPlayerSizeException.html)  initializes a game with the desired number of players and the needed card deck  **Parameters:**  players - - list of players participating in the game  **Returns:**  new game  **Throws:**  [IllegalDeckSizeException](file:///C:\Users\deck\exceptions\IllegalDeckSizeException.html) - when deck has not the right size of cards  [InvalidPlayerSizeException](file:///C:\Users\MariaH\exceptions\InvalidPlayerSizeException.html) - when player list size is above four or below two **switchToNextPlayer** void switchToNextPlayer([Game](file:///C:\\Users\\MariaH\\Desktop\\Game.html" \o "class in htw.kbe.maumau.game.export) game)  sets the player for the next round, noting whether the round is played clockwise or counterclockwise.  **Parameters:**  game – **initialCardDealing** void initialCardDealing([Game](file:///C:\\Users\\MariaH\\Desktop\\Game.html" \o "class in htw.kbe.maumau.game.export) game)  at the start of the game, each player is dealt their hand cards  **Parameters:**  game – **giveDrawnCardsToPlayer** void giveDrawnCardsToPlayer(int numberOfDrawnCards, [Game](file:///C:\Users\MariaH\Desktop\Game.html) game)  as many cards are drawn from the draw pile as are indicated and then the cards are dealt to the player  **Parameters:**  numberOfDrawnCards - - number of cards the player must draw  game – |

|  |
| --- |
| **mustPlayerDrawCards** boolean mustPlayerDrawCards([Game](file:///C:\\Users\\MariaH\\Desktop\\Game.html" \o "class in htw.kbe.maumau.game.export) game)  Checks whether the player may play a card in this round. He may not if he has to draw cards (e.g. because a seven is on top and the player has no seven in his hand)  **Parameters:**  game -  **Returns:**  true when player can play a card, false when player has to draw cards instead **applyCardRule** void applyCardRule([Game](file:///C:\\Users\\MariaH\\Desktop\\Game.html" \o "class in htw.kbe.maumau.game.export) game)  checks if the played card match a card rule belonging to the card rule further methods will be executed  **Parameters:**  game - **validateCard** void validateCard([Card](file:///C:\Users\card\export\Card.html) card, [Game](file:///C:\Users\MariaH\Desktop\Game.html) game) throws [PlayedCardIsInvalidException](file:///C:\Users\rule\exceptions\PlayedCardIsInvalidException.html)  validates the card to be played if the card is valid, it is added to the discard pile if a suit wish was played and valid then removes suit wish from game  **Parameters:**  card - - card that wants to be played  game -  **Throws:**  [PlayedCardIsInvalidException](file:///C:\Users\rule\exceptions\PlayedCardIsInvalidException.html) - when the played card is not valid for that round **setPlayersSuitWish** void setPlayersSuitWish([Suit](file:///C:\Users\card\export\Suit.html) userWish, [Game](file:///C:\Users\MariaH\Desktop\Game.html) game)  set a suit wish of a player into the game and set askForSuitWish state to false  **Parameters:**  userWish - - suit wish of the player  game – **isGameOver** boolean isGameOver([Game](file:///C:\\Users\\MariaH\\Desktop\\Game.html" \o "class in htw.kbe.maumau.game.export) game)  calls the game as won when the active player has no more hand cards  **Parameters:**  game -  **Returns:**  true, if active player has no hand cards, false if not **resetPlayersMau** void resetPlayersMau([Game](file:///C:\\Users\\MariaH\\Desktop\\Game.html" \o "class in htw.kbe.maumau.game.export) game)  set 'mau' state to false of active player  **Parameters:**  game - **saveGame** void saveGame([Game](file:///C:\Users\MariaH\Desktop\Game.html) game) throws [DaoException](file:///C:\Users\MariaH\exceptions\DaoException.html)  saves a game in a database  **Parameters:**  game - game that is to be saved  **Throws:**  [DaoException](file:///C:\Users\MariaH\exceptions\DaoException.html) **deleteGame** void deleteGame([Game](file:///C:\Users\MariaH\Desktop\Game.html) game) throws [DaoException](file:///C:\Users\MariaH\exceptions\DaoException.html)  deletes a game from the database  **Parameters:**  game - game that is to be deleted  **Throws:**  [DaoException](file:///C:\Users\MariaH\exceptions\DaoException.html) **hasGame** boolean hasGame() throws [DaoException](file:///C:\Users\MariaH\exceptions\DaoException.html)  says if at leas one game is saved in the database  **Returns:**  true if at least one game is saved in the database, false if not  **Throws:**  [DaoException](file:///C:\Users\MariaH\exceptions\DaoException.html) getSavedGame [Game](file:///C:\Users\MariaH\Desktop\Game.html) getSavedGame(long id) throws [DaoException](file:///C:\Users\MariaH\exceptions\DaoException.html)  return the game that belongs to the id  **Parameters:**  id - id that belongs to the game  **Returns:**  saved game  **Throws:**  [DaoException](file:///C:\Users\MariaH\exceptions\DaoException.html) |

## Interface PlayerService

Das Interface wird in den Klassen *GameServiceImpl* und *AppControllerImpl* benutzt.

|  |
| --- |
| ***Method Details*****createPlayers** [List](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/List.html)<[Player](file:///C:\Users\philipp.schalau\IdeaProjects\mau-mau\docs\Player.html)> createPlayers([List](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/List.html)<[String](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/String.html)> names, int aiPlayers) throws [InvalidPlayerNameException](file:///C:\Users\philipp.schalau\IdeaProjects\mau-mau\exceptions\InvalidPlayerNameException.html)  initializes new players  **Parameters:**  names - list of player names  aiPlayers - number of virtual players  **Returns:**  list of players  **Throws:**  [InvalidPlayerNameException](file:///C:\Users\philipp.schalau\IdeaProjects\mau-mau\exceptions\InvalidPlayerNameException.html) - when the name is empty, blanks only or longer then 15 symbols |

|  |
| --- |
| **addDrawnCards** void addDrawnCards([Player](file:///C:\Users\philipp.schalau\IdeaProjects\mau-mau\docs\Player.html) player,[List](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/List.html)<[Card](file:///C:\Users\philipp.schalau\IdeaProjects\card\export\Card.html)> cards)  adds drawn card(s) to hand cards to given player and hand cards are sorted by suit (from black to red) and label (from ASS to SEVEN).  **Parameters:**  player - who gets drawn cards  cards - card(s) that was drawn **removePlayedCard** void removePlayedCard([Player](file:///C:\Users\philipp.schalau\IdeaProjects\mau-mau\docs\Player.html) player, [Card](file:///C:\Users\philipp.schalau\IdeaProjects\card\export\Card.html) card)  removes card that was played from hand card of player  **Parameters:**  player - player who played the given card  card - played card to be removed **validateName** void validateName([String](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/String.html) name, [List](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/List.html)<[String](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/String.html)> names) throws [InvalidPlayerNameException](file:///C:\Users\philipp.schalau\IdeaProjects\mau-mau\exceptions\InvalidPlayerNameException.html)  validates name  **Parameters:**  name - that has to be validated  names - list of names that is needed to check if names duplicate  **Throws:**  [InvalidPlayerNameException](file:///C:\Users\philipp.schalau\IdeaProjects\mau-mau\exceptions\InvalidPlayerNameException.html) - when String name is empty, has whitespaces, is too long or names duplicate |

## Interface RuleService

Dieses Interface wird von den Klassen *AIServiceImpl* und *GameServiceImpl* importiert.

|  |
| --- |
| ***Method Details***  **getDefaultNumberOfDrawnCards**  int getDefaultNumberOfDrawnCards()  gets default number of drawn cards  **Returns:**  number of cards to be drawn  **validateCard**  void validateCard([Card](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\card\export\Card.html) playedCard,[Card](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\card\export\Card.html) topCard,[Suit](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\card\export\Suit.html) userWish, int drawCounter) throws [PlayedCardIsInvalidException](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\rule\exceptions\PlayedCardIsInvalidException.html)  validates if a card can be played, also if card was JACK, SEVEN and if there is a suit wish  **Parameters:**  playedCard - card user wants to play  topCard - card on top of discard pile  userWish - suit wish after JACK was played  drawCounter - number of drawn cards  **Throws:**  [PlayedCardIsInvalidException](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\rule\exceptions\PlayedCardIsInvalidException.html) - throw exception when card cannot be played  **mustDrawCards**  boolean mustDrawCards([Card](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\card\export\Card.html) topCard)  Checks if a player needs to draw cards  **Parameters:**  topCard - card on top of discard pile  **Returns:**  true if card has label SEVEN, false when not  **mustSuspend**  boolean mustSuspend([Card](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\card\export\Card.html) topCard)  checks if next player is suspended for one round  **Parameters:**  topCard - card on top of discard pile  **Returns:**  true if card is ASS, false if not  **mustDrawCards**  boolean mustDrawCards([Player](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\player\export\Player.html) player, [Card](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\card\export\Card.html) topCard, int drawCounter)  checks if at least one hand card of the player is a SEVEN and top card is a SEVEN and draw counter is greater or equal default number of drawn cards, decides if a player has to draw cards  **Parameters:**  player - player who is in turn  topCard - card on top of discard pile  drawCounter - number of drawn cards  **Returns:**  true if player has to draw cards, false if player has not to draw  **isCardJack**  boolean isCardJack([Card](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\card\export\Card.html) topCard)  Checks if last card played was a JACK and player can make a wish  **Parameters:**  topCard - card on top of discard pile  **Returns:**  true if card was JACK, false if not  **changeGameDirection**  boolean changeGameDirection([Card](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\card\export\Card.html) topCard)  Checks if card was NINE and game direction has to change  **Parameters:**  topCard - card on top of discard pile  **Returns:**  true if card was NINE, false if not  **isPlayersMauInvalid**  boolean isPlayersMauInvalid([Player](file:///C:\Users\MariaH\Desktop\javaDoc\htw\kbe\maumau\player\export\Player.html) player)  Checks if player said 'mau' and if it's valid  **Parameters:**  player - player in turn  **Returns:**  false, if players 'mau' is valid, true if it's invalid |

## Interface AIService

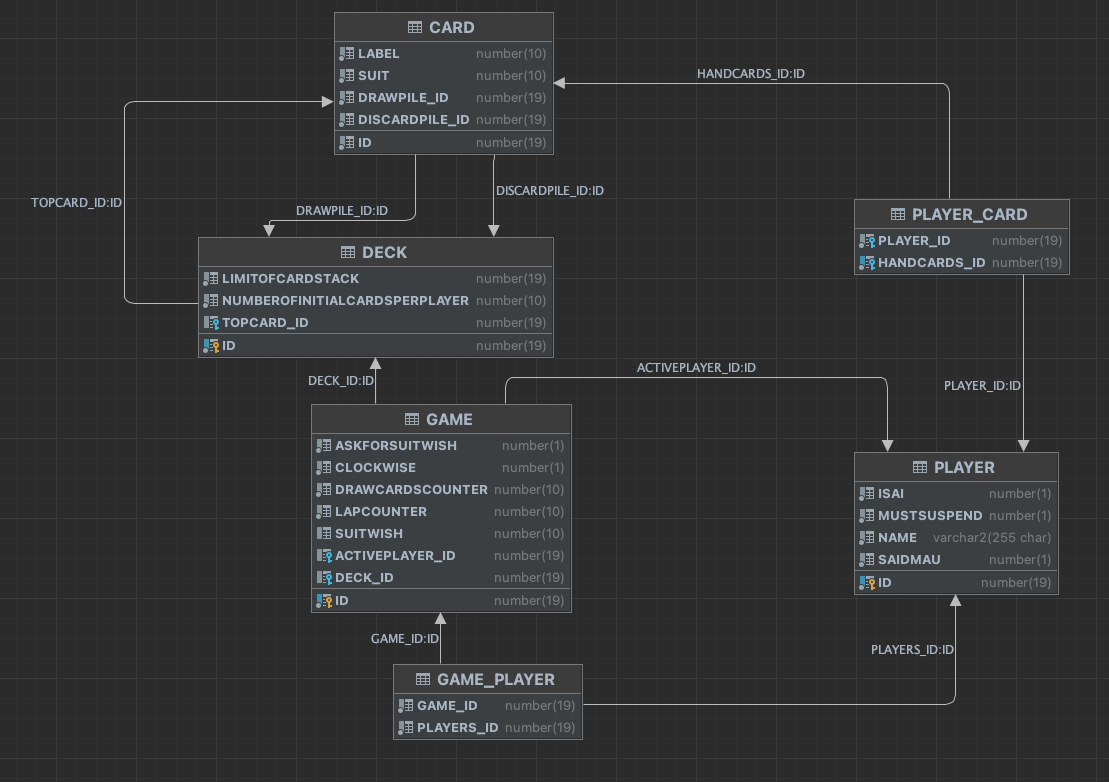
Der *AIService* wird in den Klassen *GameServiceImpl* und *AppControllerImpl* aufgerufen.

|  |
| --- |
| ***Method Details*****getPlayedCard** [Card](file:///C:\Users\card\export\Card.html) getPlayedCard([Player](file:///C:\Users\player\export\Player.html) AI, [Card](file:///C:\Users\card\export\Card.html) topCard, [Suit](file:///C:\Users\card\export\Suit.html) suitWish, int drawCounter)  returns a valid card to play or null when it's not available  **Parameters:**  AI - AI player who is in turn  topCard - card that is on top of discard pile  suitWish - suit that may have been requested  drawCounter - counter that gives the number of drawn cards  **Returns:**  card if valid or null when invalid **saidMau** boolean saidMau([Player](file:///C:\\Users\\player\\export\\Player.html" \o "class in htw.kbe.maumau.player.export) AI)  returns 'mau' state of AI player  **Parameters:**  AI - AI player who is in turn  **Returns:**  true when player has only one hand card left, false if not **getSuitWish** [Suit](file:///C:\Users\card\export\Suit.html) getSuitWish([Player](file:///C:\\Users\\player\\export\\Player.html" \o "class in htw.kbe.maumau.player.export) AI)  returns a suit wish that is also available in the AI player's hand cards  **Parameters:**  AI - AI player who is in turn  **Returns:**  suit wish |

|  |
| --- |
| **removePlayedCard** void removePlayedCard([Player](file:///C:\Users\player\export\Player.html) AI, [Card](file:///C:\Users\card\export\Card.html) card)  removes card that was played from hand card of AI player  **Parameters:**  AI - AI player who played the given card  card - played card to be removed **addDrawnCards** void addDrawnCards([Player](file:///C:\Users\player\export\Player.html) AI, [List](https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/List.html)<[Card](file:///C:\Users\card\export\Card.html)> cards)  adds drawn card(s) to hand cards to given AI player and hand cards are sorted by suit (from black to red) and label (from ASS to SEVEN).  **Parameters:**  AI - AI player who gets drawn cards  cards - card(s) that was drawn |

# Konzeptionelles Datenmodell

Das folgende Datenmodell basiert auf der Grundlage der von uns festgelegten Entitäten und Beziehungen und wurde mit Hilfe des IDE IntelliJ Plugins *Java EE: Persistence (JPA)* generiert. Die daraus entstandenen Tabellen CARD, DECK, GAME und PLAYER sind hierbei gleichnamig zu unseren Komponenten. Die Tabellen PLAYER\_CARD und GAME\_PLAYER sind Zwischentabellen. Die Erste Tabelle PLAYER\_CARD wird benötigt damit eine Karte dem jeweiligen Spieler zugeordnet werden kann (PLAYER\_CARD). Die GAME\_PLAYER Tabelle ist dafür zuständig Spieler dem richtigen Spiel zuzuordnen.

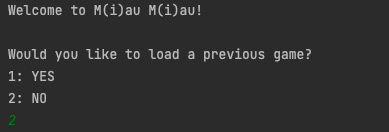
****

# Präsentationsschicht

Das Spiel ist eine interaktive Konsolenanwendung. Demzufolge werden die Benutzereingaben über die Konsole eingetragen und die Ausgaben des Spiels werden ebenfalls darüber angezeigt.

## Spielbeginn

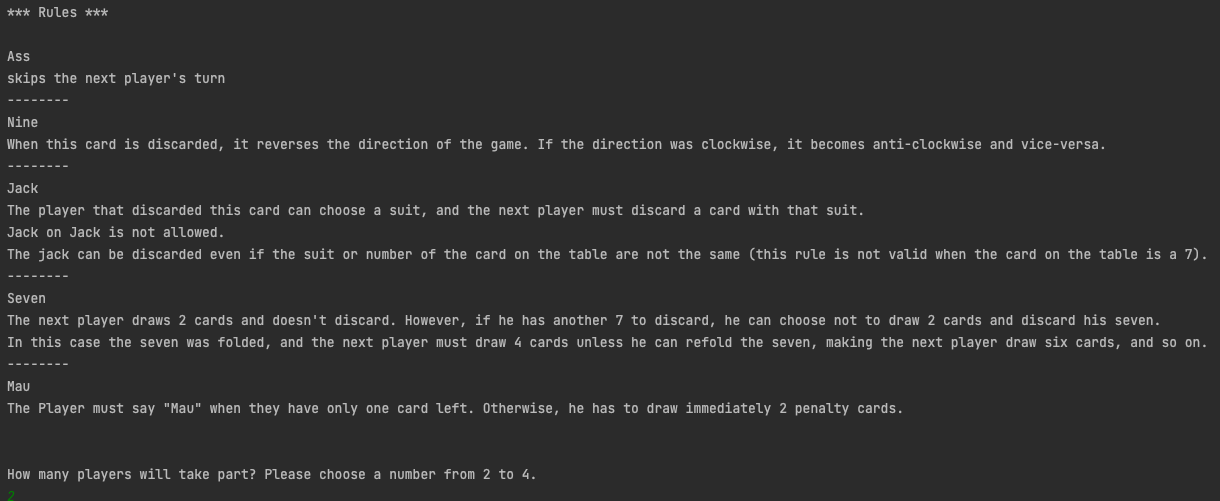
Der Spieler wird begrüßt und gefragt, ob er ein neues Spiel starten möchte oder ein bereits vorhandenes Spiel weiter führen möchte:

** Ein Bild, das Text enthält.

Automatisch generierte Beschreibung**

Möchte er ein bereits vorhandenes Spiel laden wird die ID des Spiels abgefragt. Nach korrekter Eingabe der ID wird das Spiel geladen und ab der zuletzt gespeicherten Runde fortgeführt.

Startet der Spieler ein neues Spiel werden zunächst die Regeln angezeigt sowie die Anzahl der teilnehmenden Spieler erfragt:

****

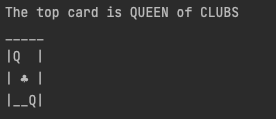
Hat der Spieler die Anzahl der Teilnehmer festgelegt, erfolgt eine Abfrage wie viele dieser Spieler AI Spieler sein sollen. Die Namen der Spieler werden abgefragt, die AI Spieler erhalten automatisch Namen mit dem Zusatz „AI“. Dem Spieler wird die ID des Spiels mitgeteilt. Es ist notwendig, dass der Spieler sich die ID merkt bzw. notiert, um zu einem späteren Zeitpunkt das Spiel wieder laden zu können:

**Ein Bild, das Text enthält.

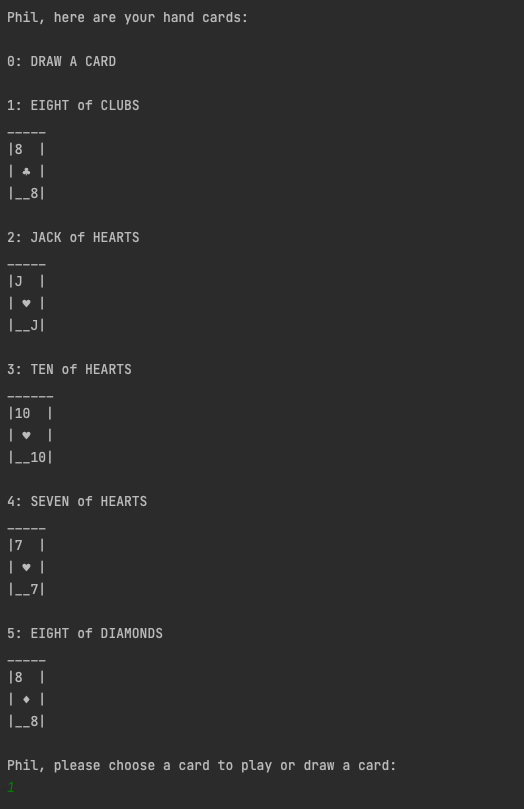
Automatisch generierte Beschreibung**

## Spielablauf

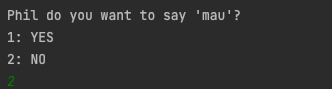
Zu Beginn jeder Spielrunde wird dem Spieler die oberste Karte des Ablagestapels präsentiert:

****

Zusätzlich werden dem Spieler alle Handkarten angezeigt und gefragt, ob er eine Karte ziehen oder eine ablegen möchte:

****

Nach jeder gespielten Karte wird der Spieler gefragt, ob er „Mau“ sagen möchte. Hat der Spieler beim Ablegen der letzten Karte nicht „Mau“ gesagt, muss er eine Strafkarte ziehen.

****

Hat der Spieler „Mau“ gewählt wird dies in der Konsole ausgegeben:

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Wenn ein Bube gelegt wird, wird der Spieler nach seinem Farbwunsch gefragt und in der nachfolgenden Runde alle Spieler über den Farbwunsch informiert:

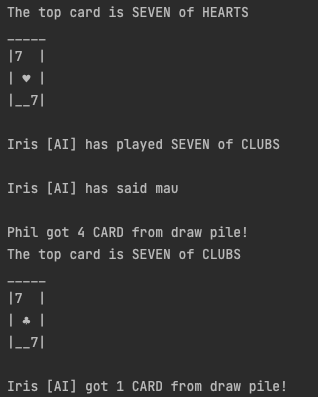
Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

**Ein Bild, das Text enthält.

Automatisch generierte Beschreibung**

Wird eine Karte mit dem Wert 7 gespielt und der nächste Spieler kann sie nicht erwidern, wird angezeigt wie viele Karten dieser Spieler ziehen muss:

****

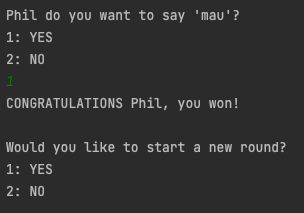
Wird eine Neun gespielt, wird die Spielrichtung geändert und dem Spieler angezeigt:

**Ein Bild, das Text enthält.

Automatisch generierte Beschreibung**

## Spielende

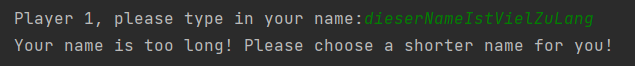
Das Spiel ist beendet, wenn der Spieler keine Karten mehr auf der Hand hat. Der Gewinner des Spiels wird angezeigt. Zusätzlich wird erfragt, ob eine neue Spielrunde gestartet werden soll:

****

## Fehlerausgaben

Hier folgt eine Auswahl von möglichen Fehlermeldungen:

Der Name des Spielers darf nicht länger als 15 Zeichen und nicht kürzer als 3 Zeichen sein. Außerdem darf der Name keine Zahlen enthalten:





Ein Bild, das Text, orange enthält.

Automatisch generierte Beschreibung

Die eingebende Game ID zum Laden des Spiels wird in der Datenbank nicht gefunden:

**Ein Bild, das Text enthält.

Automatisch generierte Beschreibung**

Eine Karte kann nicht gespielt werden, weil die Farbe oder der Wert der Karte nicht passt:

****

Ein Bube darf nicht auf einen Buben gelegt werden:

****

Die oberste Karte ist eine Sieben und der Spieler hat ebenfalls eine Sieben in seiner Hand, wählt jedoch eine andere Karte. Er ist dennoch verpflichtet eine Sieben zu legen oder freiwillig die Karten zu ziehen:

****

Taucht während des Spiels ein Fehler auf, der nicht auf die fehlerhafte Eingabe des Spielers zurückzuführen ist, wird das Spiel mit einer entsprechenden Fehlerausgabe beendet:

**Ein Bild, das Text enthält.

Automatisch generierte Beschreibung**

# Frameworks

Die Software basiert auf der Programmiersprache Java in der Version 17 und wird von Apache Maven, einem Build Management System, verwaltet und gebaut. Das Projekt ist als ein Multimodulprojekt konzipiert.

Um die Erstellung und Durchführung von Tests zu erleichtern, wurden die Frameworks JUnit 5 und Mockito in Maven eingebunden. Letzteres wird speziell für das Mocken von Objekten verwendetet.

Um mit der Oracle Datenbank zu kommunizieren, wird auf den Hibernate EntityManager und der Oracle Java Database Connectivity (ojdbc) zurückgegriffen. Die EntityManager-Api wird für den Zugriff auf die Datenbank verwendetet, d.h. sie kann persistente Entitätsinstanzen erstellen und entfernen, aber auch Entitäten anhand ihrer Primärschlüsselidentität finden und Abfragen über alle Entitäten durchführen. Die Oracle Java Database Connectivity ist eine Java-API, mit dem sich SQL-Anweisungen in Datenbanken ausführen lassen.

Der Spielablauf wird mittels Log4j protokolliert und die Dependency Injection wird mit Hilfe von Spring Context realisiert.

Im Parent pom sind alle Konfigurationen eingetragen, die den Kind-Komponenten vererbt werden:

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> <project xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://maven.apache.org/POM/4.0.0"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">   <modelVersion>4.0.0</modelVersion>  <groupId>htw.kbe</groupId>  <artifactId>maumau</artifactId>  <version>0.0.1-SNAPSHOT</version>   <packaging>pom</packaging>  <name>mau-mau</name>  <description>KBE Mau-Mau Project</description>   <modules>  <module>game</module>  <module>card</module>  <module>deck</module>  <module>rule</module>  <module>player</module>  <module>JaCoCo</module>  <module>controller</module>  <module>gamePlay</module>  <module>virtualPlayer</module>  </modules>   <dependencies>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-context</artifactId>  <version>5.3.20</version>  </dependency>  <dependency>  <groupId>org.junit.jupiter</groupId>  <artifactId>junit-jupiter</artifactId>  <version>5.8.2</version>  <scope>test</scope>  </dependency>  <dependency>  <groupId>org.mockito</groupId>  <artifactId>mockito-junit-jupiter</artifactId>  <version>4.5.1</version>  <scope>test</scope>  </dependency>  <dependency> <!-- NEEDED DEPENDENCY OTHERWISE MOCKITO BREAKS - IS PROBABLY SET TO DIFFERENT VERSION OF SOME TRANSITIVE DEPENDENCIES -->  <groupId>net.bytebuddy</groupId>  <artifactId>byte-buddy</artifactId>  <version>1.12.9</version>  </dependency>  <dependency>  <groupId>org.apache.logging.log4j</groupId>  <artifactId>log4j-api</artifactId>  <version>2.17.2</version>  </dependency>  <dependency>  <groupId>org.apache.logging.log4j</groupId>  <artifactId>log4j-core</artifactId>  <version>2.17.2</version>  </dependency>  <dependency>  <groupId>org.hibernate</groupId>  <artifactId>hibernate-entitymanager</artifactId>  <version>5.6.1.Final</version>  </dependency>  <dependency>  <groupId>com.oracle.ojdbc</groupId>  <artifactId>ojdbc8</artifactId>  <version>19.3.0.0</version>  </dependency>  <dependency>  <groupId>javax.persistence</groupId>  <artifactId>javax.persistence-api</artifactId>  <version>2.2</version>  </dependency>  </dependencies>   <build>  <plugins>  <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-compiler-plugin</artifactId>  <version>3.10.1</version>  <configuration>  <release>17</release>  </configuration>  </plugin>  <plugin>  <artifactId>maven-surefire-plugin</artifactId>  <version>2.22.2</version>  </plugin>  </plugins>  </build>   <properties>  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>  </properties>  </project> |

Um die Testabdeckung des gesamten Projektes zu überblicken, wurde JaCoCo (als separated Modul) eingeführt und zeigt folgende Konfigurationen in der pom:

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> <project xmlns="http://maven.apache.org/POM/4.0.0"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  <parent>  <artifactId>maumau</artifactId>  <groupId>htw.kbe</groupId>  <version>0.0.1-SNAPSHOT</version>  </parent>   <modelVersion>4.0.0</modelVersion>   <artifactId>JaCoCo</artifactId>  <version>0.0.1-SNAPSHOT</version>  <name>JaCoCo</name>   <description>Compute aggregated test code coverage</description>   <properties>  <maven.deploy.skip>true</maven.deploy.skip>  </properties>   <dependencies>  <dependency>  <groupId>htw.kbe</groupId>  <artifactId>deck</artifactId>  <version>0.0.1-SNAPSHOT</version>  <scope>compile</scope>  </dependency>  <dependency>  <groupId>htw.kbe</groupId>  <artifactId>player</artifactId>  <version>0.0.1-SNAPSHOT</version>  <scope>compile</scope>  </dependency>  <dependency>  <groupId>htw.kbe</groupId>  <artifactId>card</artifactId>  <version>0.0.1-SNAPSHOT</version>  <scope>compile</scope>  </dependency>  <dependency>  <groupId>htw.kbe</groupId>  <artifactId>rule</artifactId>  <version>0.0.1-SNAPSHOT</version>  <scope>compile</scope>  </dependency>  <dependency>  <groupId>htw.kbe</groupId>  <artifactId>game</artifactId>  <version>0.0.1-SNAPSHOT</version>  <scope>compile</scope>  </dependency>  <dependency>  <groupId>htw.kbe</groupId>  <artifactId>controller</artifactId>  <version>0.0.1-SNAPSHOT</version>  <scope>compile</scope>  </dependency>  <dependency>  <groupId>htw.kbe</groupId>  <artifactId>virtualPlayer</artifactId>  <version>0.0.1-SNAPSHOT</version>  <scope>compile</scope>  </dependency>  </dependencies>   <build>  <plugins>  <plugin>  <groupId>org.jacoco</groupId>  <artifactId>jacoco-maven-plugin</artifactId>  <version>0.8.8</version>  <executions>  <execution>  <id>report-aggregate</id>  <phase>verify</phase>  <goals>  <goal>report-aggregate</goal>  </goals>  </execution>  </executions>  </plugin>  </plugins>  </build>  </project> |

Die Konfigurationsdatei (log4j2.xml) wurde folgender Maßen für das Protokollieren des Spielverlaufs aufgesetzt:

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <Configuration status="WARN">  <Appenders>  <File name="fout" fileName="logs/app.log" append="true">  <PatternLayout>  <Pattern>%d{yyyy-MM-dd HH:mm:ss} [%c{1}]: %-5p %m%n</Pattern>  </PatternLayout>  </File>  </Appenders>  <Loggers>  <Root level="INFO">  <AppenderRef ref="fout"/>  </Root>  </Loggers> </Configuration> |

Die Konfiguration der persistence.xml sehen wie folgt aus:

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> <persistence version="2.1"  xmlns="http://xmlns.jcp.org/xml/ns/persistence" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence http://xmlns.jcp.org/xml/ns/persistence/persistence\_2\_1.xsd">   <persistence-unit name="MauMau" transaction-type="RESOURCE\_LOCAL">  <provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>  <class>htw.kbe.maumau.game.export.Game</class>  <properties>  <property name="javax.persistence.jdbc.driver" value="oracle.jdbc.OracleDriver" />  <property name="javax.persistence.jdbc.url" value="jdbc:oracle:thin:@oradbs03.f4.htw-berlin.de:1521:oradb1" />  <property name="javax.persistence.jdbc.user" value="u572897" />  <property name="javax.persistence.jdbc.password" value="p572897" />  <property name="hibernate.dialect"  value="org.hibernate.dialect.SQLServer2012Dialect" />  <property name="hibernate.show\_sql" value="true" />  <property name="hibernate.hbm2ddl.auto" value="update" />  </properties>  </persistence-unit> </persistence> |

# Ablaufumgebung

Das Spiel kann auf jedem Rechner laufen, der eine Internetanbindung hat. Der Zugang zum Internet wird benötigt, um eine Verbindung zur Datenbank herzustellen, die das Spiel zur Verfügung stellt. Betriebssysteme wie Windows, macOS und Linux werden unterstützt.

Zusätzlich muss auf dem Rechner Java 17 installiert sein sowie eine Entwicklungsumgebung wie beispielsweise IntelliJ IDEA oder Visual Studio Code.