

## Verteilte Systeme I

### Dokumentation

**Thema:**

Konzeptionierung und Implementierung von TicTacToe-Online

**Projektteilnehmer:**

Mai, Richard  
Sutheswaran, Suwhathi  
Becker, Sascha  
Haas, Hendrik

Pinnecker, Joschua  
Klein, Michael  
Dobicki, Jan

**Betreuende Korrektoren:**

Prof. Dr. Markus Esch  
Dipl.-Ing. (FH) Michael Sauer DEA (UVigo)

**Bildungseinrichtung:**

Hochschule für Technik und Wirtschaft des Saarlandes

**Abgabedatum:**

19.09.18

# Inhaltsverzeichnis

1	Einführung.....	5
1.1	Bausteinsicht / Verteilungssicht.....	6
1.1.1	Ebene 0.....	6
1.1.2	Ebene 1.....	6
1.1.3	Ebene 2.....	6
1.2	Darstellung der Komponenten.....	7
2	Client /Server.....	11
2.1	Spielverlauf.....	11
2.2	Spieler ist aktiv.....	12
2.3	Quickplay.....	13
3	GUI.....	14
3.1	Startside.....	14
3.1.1	Registry.....	15
3.1.2	Login.....	16
3.2	Logout.....	17
3.3	Spielraum.....	18
3.4	TicTacToe.....	20
3.5	Spielverlauf.....	21
4	Lastenheft für TicTacToe-Online.....	23
4.1	Genutzte Technologien.....	23
4.2	Zielbestimmung.....	23
4.3	Zielgruppen.....	23
4.4	Produktübersicht.....	24
4.5	Produktfunktionen.....	25
4.5.1	LF10.....	25
4.5.2	LF20.....	25
4.5.3	LF30.....	25
4.5.4	LF40.....	25
4.5.5	LF50.....	26
4.5.6	LF60.....	26
4.5.7	LF70.....	26
4.5.8	LF80.....	26
4.5.9	LF90.....	27
4.5.10	LF100.....	27
4.5.11	LF110.....	27
4.6	Produktdaten.....	28
4.7	Produktleistungen.....	28
4.8	Qualitätsanforderungen.....	28
4.9	Glossar TicTacToe-Online.....	29
4.9.1	Spieleraccount.....	29
4.9.2	Spielerdaten.....	29
4.9.3	Login.....	29
4.9.4	Logout.....	29
5	Usecases.....	30
5.1	UC 10 – Login.....	30
5.1.1	UC 20 – Registrieren.....	31

5.2 UC 30 User erstellen.....	32
5.3 UC 40-Spielerliste ansehen.....	33
5.3.1 UC 50-Spieler herausfordern.....	34
5.4 UC 60 Logout.....	35

## Abbildungsverzeichnis

Abbildung 1: Bausteinsicht-Ebene 0.....	6
Abbildung 2: Bausteinsicht-Ebene 1.....	6
Abbildung 3: Bausteinsicht-Ebene 2.....	6
Abbildung 4: Verwendete Technologien.....	7
Abbildung 5: Klassendiagramm - Userinterface.....	8
Abbildung 6: Klassendiagramm - Model.....	9
Abbildung 7: Klassendiagramm - Messages.....	10
Abbildung 8: Spielverlauf - Sequenzdiagramm.....	11
Abbildung 9: Der Zug - Sequenzdiagramm.....	12
Abbildung 10: Quickplay - Sequenzdiagramm.....	13
Abbildung 11: Startside.....	14
Abbildung 12: Register.....	15
Abbildung 13: Login - Sequenzdiagramm.....	16
Abbildung 14: Logout - Sequenzdiagramm.....	17
Abbildung 15: Spielraum.....	18
Abbildung 16: Spielraum - Sequenzdiagramm.....	19
Abbildung 17: TicTacToi.....	20
Abbildung 18: Spielverlauf - Sequenzdiagramm.....	21
Abbildung 19: Spielverlauf - Aktivitätsdiagramm.....	22
Abbildung 20: TicTacToe-Online.....	24

# 1 Einführung

Das Ziel dieses Projektes ist die Implementierung des Spiels TicTacToe. Das eigentliche Spiel läuft online auf einem Server auf den mittels Client zugegriffen wird.

Die Implementierung des Projekts erfolgt mit der Programmiersprache Java. Für die Kommunikation zwischen Server und Client werden Java-Sockets verwendet. Die Gestaltung der GUI Grafical User Interface erfolgt durch die Software Scene Builder. Für Ihre Implementierung wird JavaFX verwendet.

In der späteren Software haben zwei oder mehr Spieler die Möglichkeit in TicTacToe gegeneinander anzutreten. Zur Teilname an einem Spiel ist es für den Spieler zunächst notwendig sich einen Spieleraccount anzulegen und sich einzuloggen.

Nach dem erfolgreichen Login ist dem Spieler über die Funktion QuickPlay ein sofortiger Spielstart gegen einen zufällig ausgewählten Gegner möglich. Ob Mitspieler online sind kann über einen Spielerliste eingesehen werden. Weiterhin ist es möglich über die Spielerliste gezielt bestimmte Mitspieler zum Spiel herauszufordern.

## 1.1 Bausteinsicht / Verteilungssicht

### 1.1.1 Ebene 0

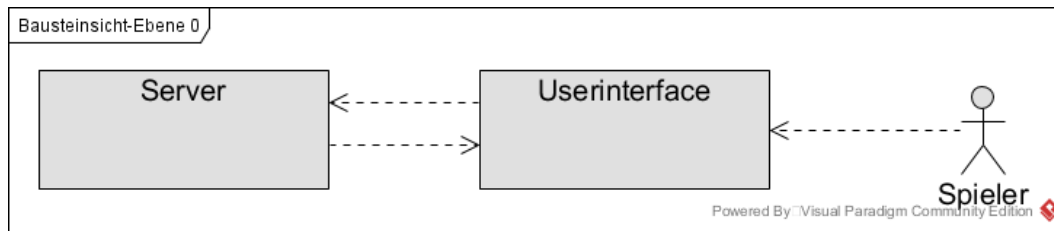


Abbildung 1: Bausteinsicht-Ebene 0

### 1.1.2 Ebene 1

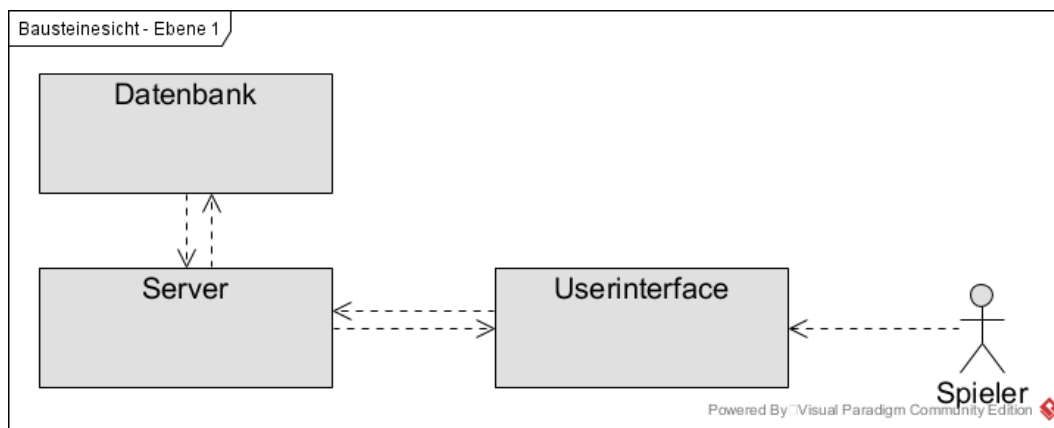


Abbildung 2: Bausteinsicht-Ebene 1

### 1.1.3 Ebene 2

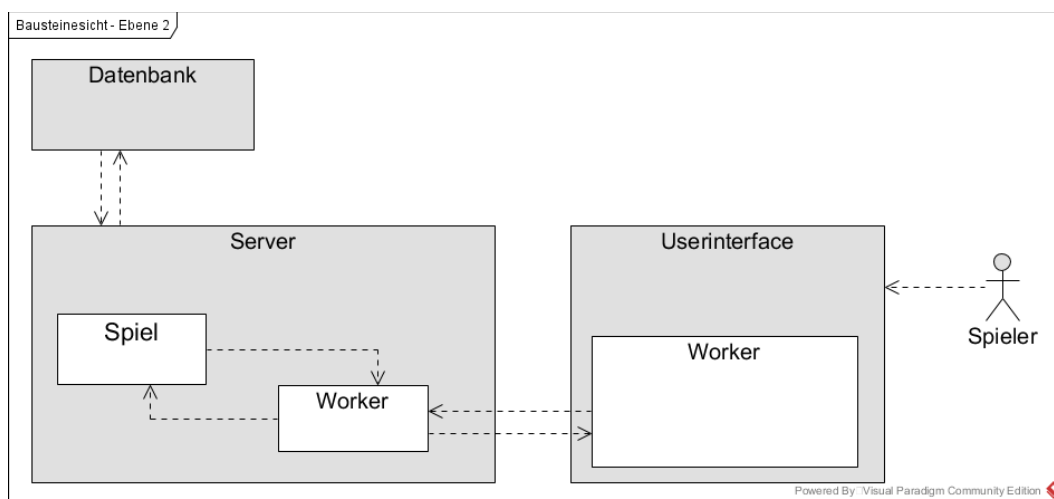


Abbildung 3: Bausteinsicht-Ebene 2

## 1.2 Darstellung der Komponenten

# Verwendete Technologien

 **GitHub**



IntelliJ



Scene Builder



**Maven™**

---

Visual Paradigm

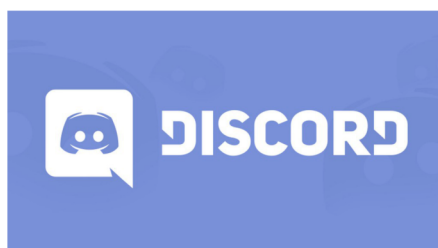


Abbildung 4: Verwendete Technologien





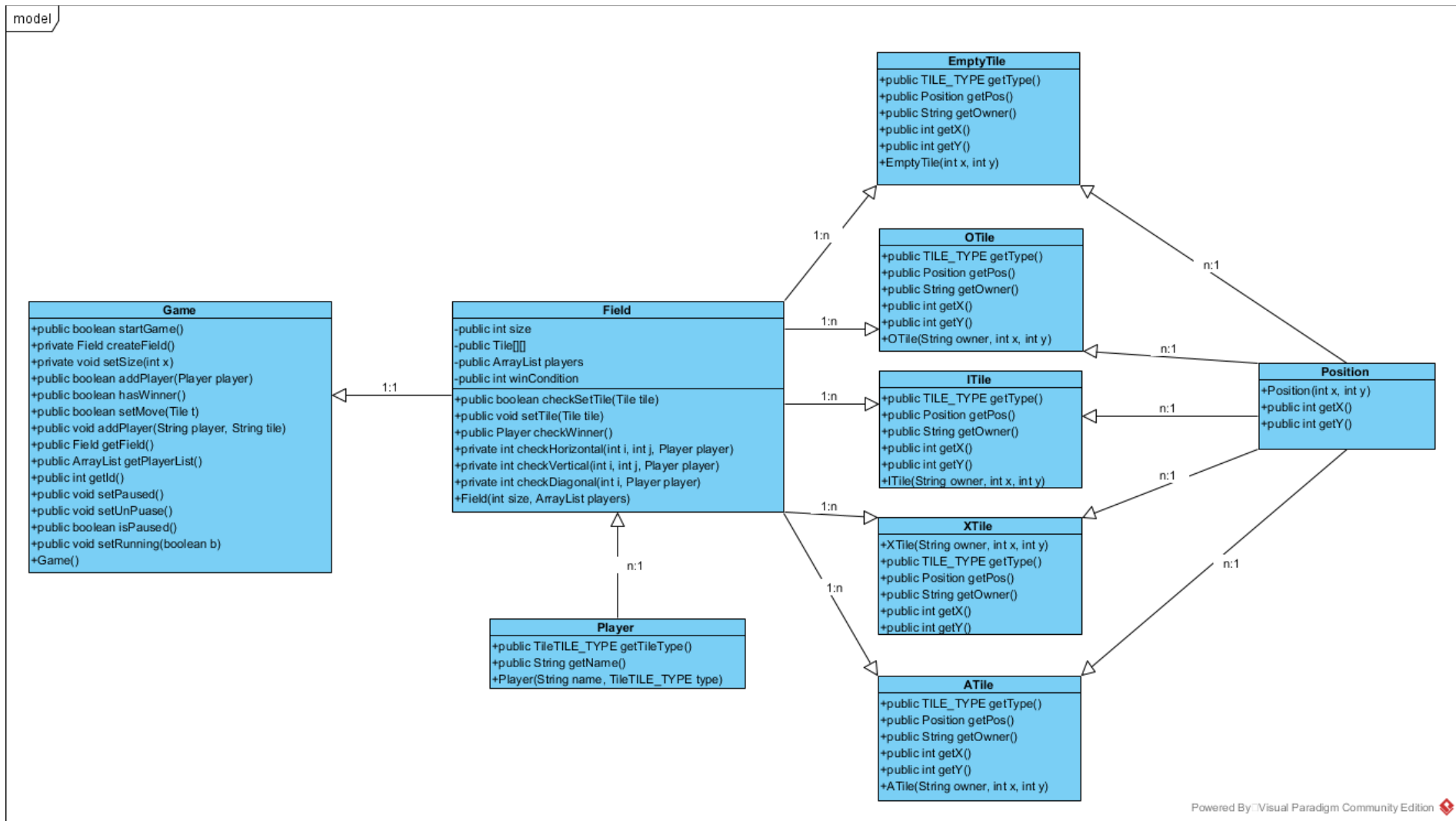


Abbildung 6: Klassendiagramm - Model

## messages

**GameFinish**

```
+public void executeEvent(Game game)
+public boolean validateEvent(Game game)
+public void sendCommand(ClientConnection cc)
+public String getText()
+public String getNameWnr()
+GameFinish(int id, String NameWnr)
```

**NewPlayer**

```
+public void executeEvent(Game game)
+public boolean validateEvent(Game game)
+public void sendCommand(ClientConnection cc)
+public String getText()
+NewPlayer(String name, String type)
```

**NewGame**

```
+public void executeEvent(Game game)
+public boolean validateEvent(Game game)
+public void sendCommand(ClientConnection cc)
+public String getText()
+getAttribute()
+setAttribute(attribute) : void
+NewGame()
```

**Move**

```
+public void executeEvent(Game game)
+public boolean validateEvent(Game game)
+public void sendCommand(ClientConnection cc)
+public String getText()
+public String getNameWnr()
+Move(int x, int y, String tile, String name)
```

**CommandFactoryImpl**

```
+public CommlDs getCommlDs()
+public int getPlayerCount()
+public Command createRegister(int var1, String var2, String var4)
+public Command createDoneActing(int var1)
+public Command createWatch(int var1)
+public Command createNewGame(int c)
+public Command createGameFinished(int c, int var1, String msg)
+public Command createGamePaused(int c, int var1)
+public Command lostPassword(int var1)
+public Command createChat(int c, String msg)
+public Command createPlayer(int c, String n, String t)
+public Command createLogin(int c, String a, int b)
+CommandFactoryImpl(CommlDs commlDs)
```

**EventFactoryImpl**

```
+public Event createRegister()
+public Event createRegistrationAbort()
+public Event createActNow(int var1)
+public Event createDoneActing(int var1)
+public Event createWinner(String var1)
+public Event createMove(int x, int y, String tile, String name)
+public Event createNewGame()
+public Event createGameFinish(int id, String name)
+public Event createGamePause(int id)
+public Event createChat(String message)
+public Event createPlayer(String name, String type)
+public Event createLogin(String name, int id)
+EventFactoryImpl()
```

**Command**

```
+public void executeCommand(Game game)
+public boolean validateCommand(Game game)
+public void sendResults(ServerConnection sc)
```

**Chat**

```
+public void executeEvent(Game gamet)
+public boolean validateEvent(Game game)
+public void sendCommand(ClientConnection cc)
+public String getText()
+Chat(String message)
```

**GamePause**

```
+public void executeEvent(Game game)
+public boolean validateEvent(Game game)
+public void sendCommand(ClientConnection cc)
+public String getText()
+GamePause(int gameId)
```

**Login**

```
+public void executeEvent(Game game)
+public boolean validateEvent(Game game)
+public void sendCommand(ClientConnection cc)
+public String getText()
+Login(String name, int id)
```

Abbildung 7: Klassendiagramm - Messages

## 2 Client /Server

### 2.1 Spielverlauf

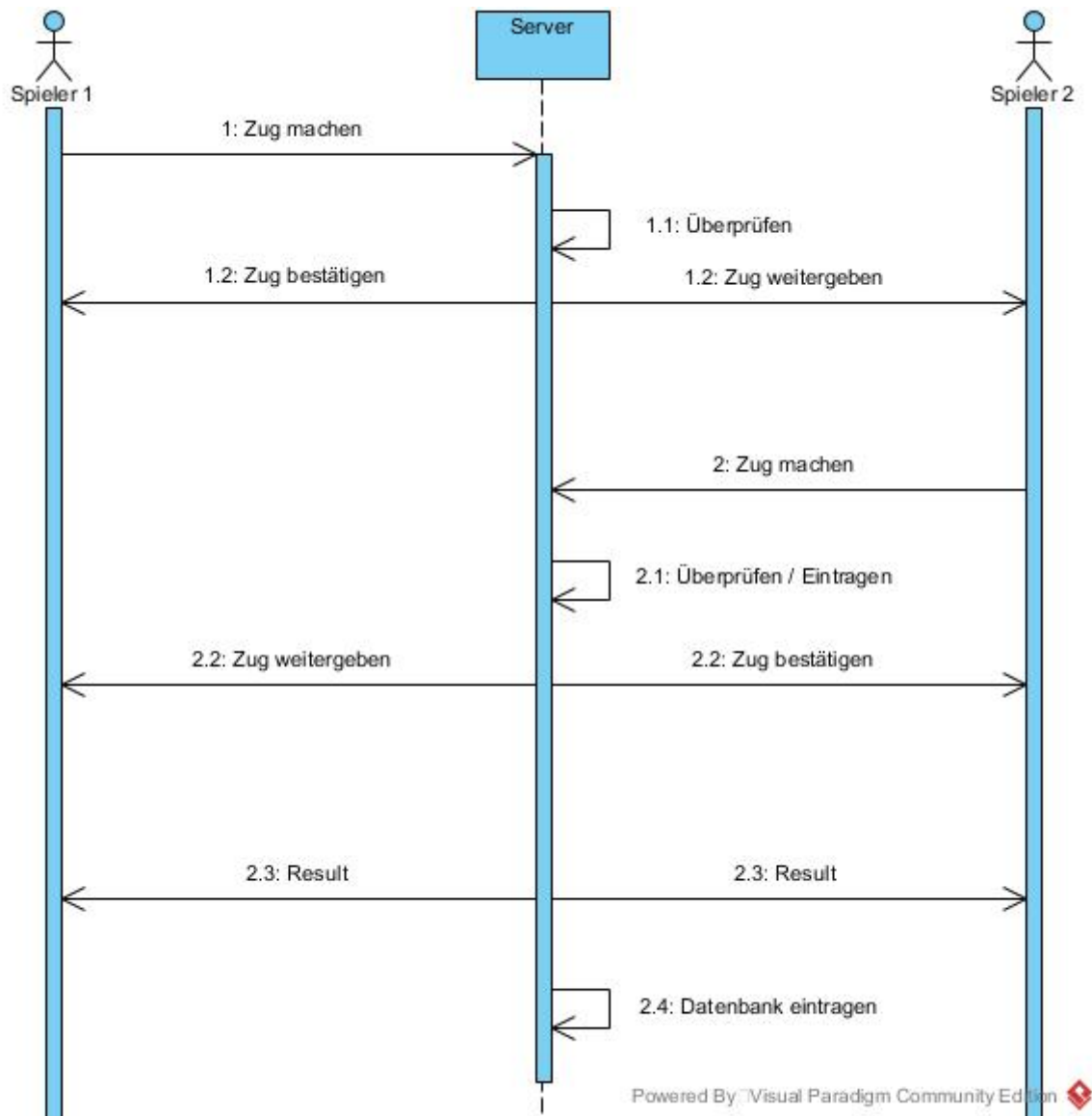


Abbildung 8: Spielverlauf - Sequenzdiagramm

## 2.2 Spieler ist aktiv

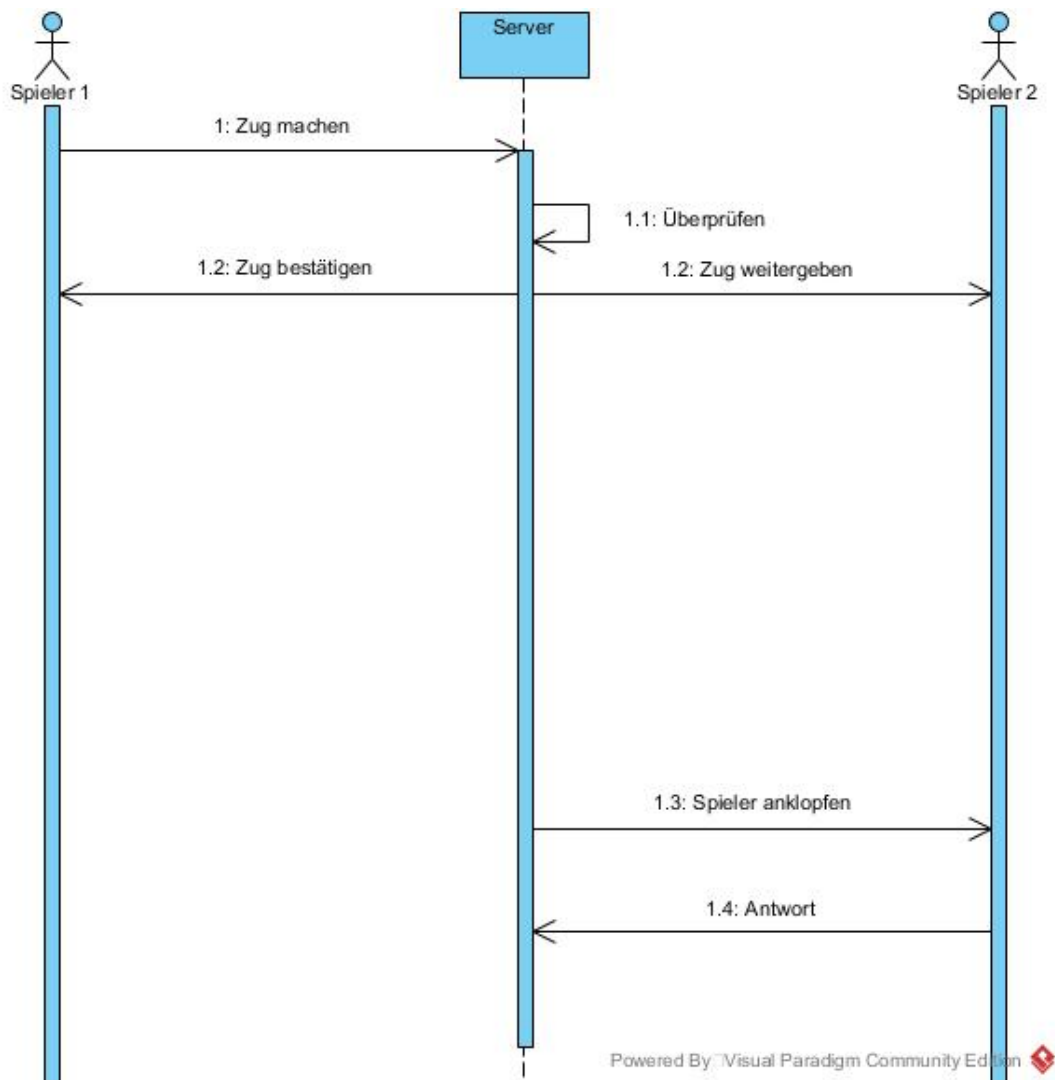


Abbildung 9: Der Zug - Sequenzdiagramm

## 2.3 Quickplay

Beim Start eines Quickplay-Spiels durch einen Spieler, wird eine Nachricht an den Spieleserver geschickt, dass nach einem zufälligen Gegner gesucht wird. Die Suchanfrage landet in einer Liste mit den Quick-Play-Suchanfragen der anderen Spieler. Der Server sucht dann per Zufallsgenerator zwei dieser Spieler aus um gegeneinander anzutreten und schickt an die Client-Software der entsprechenden Spieler eine entsprechende Nachricht.

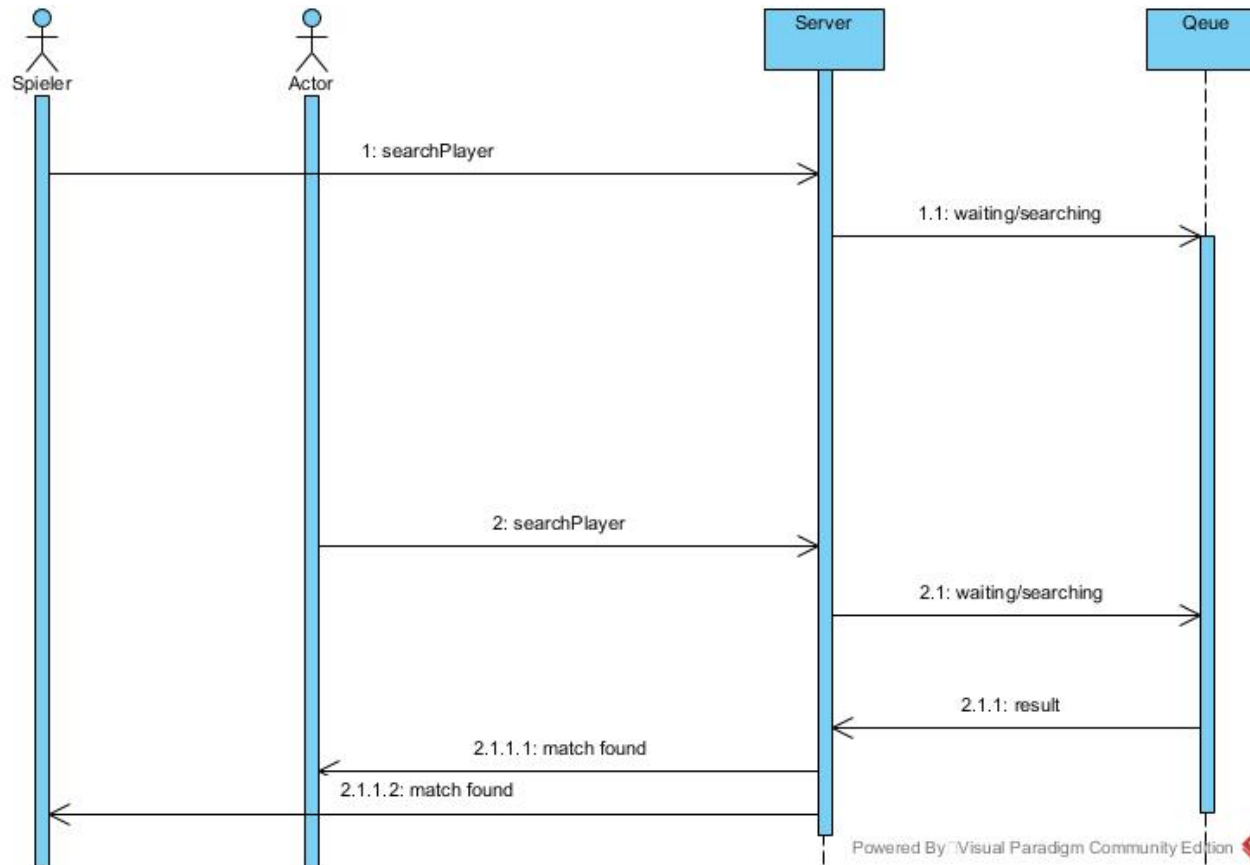


Abbildung 10: Quickplay - Sequenzdiagramm

## 3 GUI

### 3.1 Startside

Startside ist der Einstieg in das Spiel. Auf dieser Oberfläche finden sich die Spieler wieder sobald sie die Client-Software starten. Neue Spieler haben hier die Möglichkeit sich für das Spiel zu registrieren. Bereits registrierte Spieler haben hier die Möglichkeit sich ins Spiel einzuloggen. Über den Button 'Quit' wird die Client-Software beendet.

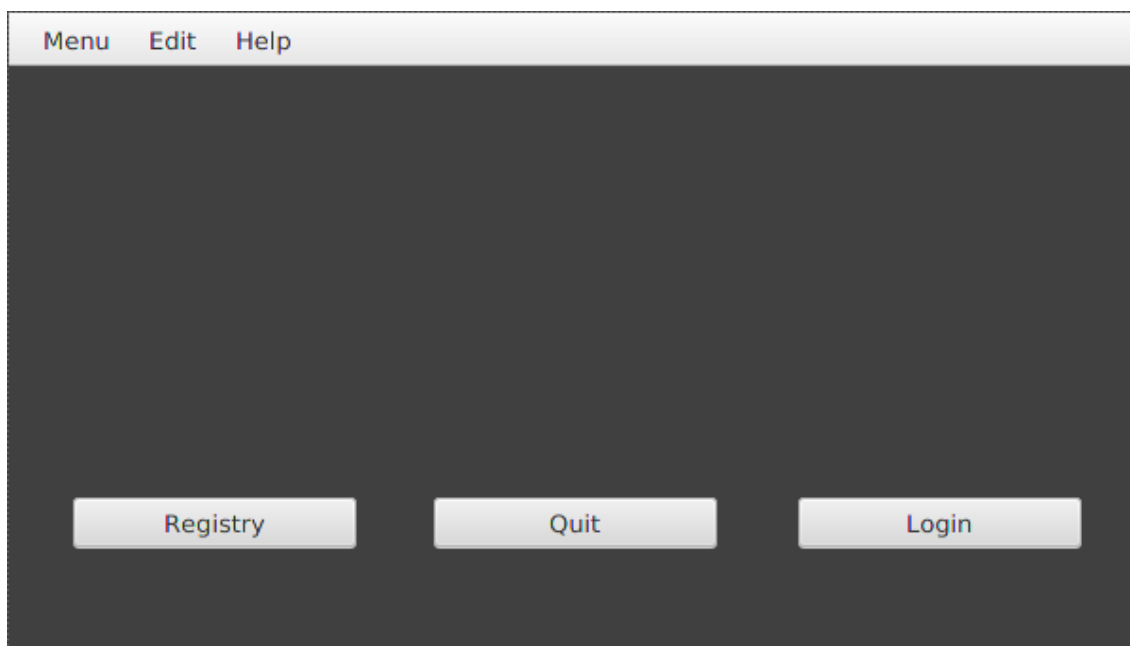


Abbildung 11: Startside

### 3.1.1 Registry

Die Registry ist unabdingbar für die Verwendung des Spiels. Hierbei muss der Spieler eine gültig E-Mail-Adresse einen Aliasnamen und ein Passwort hinterlegen, welche auf dem Server hinterlegt werden. Mit diesen Daten erfolgt die eindeutige Identifikation eines Spielers gegenüber anderen Spielern.

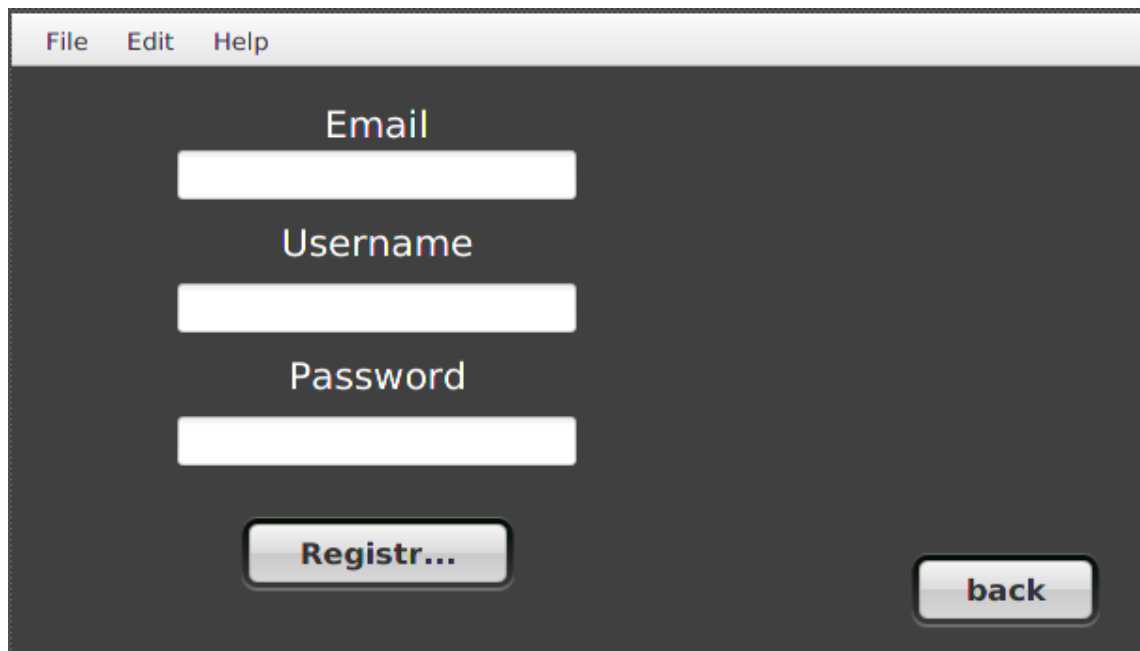
The image shows a graphical user interface for a registration process. It features a dark gray background with a light gray title bar at the top containing the menu items 'File', 'Edit', and 'Help'. The main area contains three white text input fields stacked vertically. The first field is labeled 'Email', the second 'Username', and the third 'Password'. Below these fields are two buttons: a 'Registr...' button on the left and a 'back' button on the right. Both buttons have a light gray gradient and a dark border.

Abbildung 12: Register

### 3.1.2 Login

Der Login startet durch die Eingabe der Spielerdaten durch den Spieler. Die eingegebenen Daten werden danach geprüft ob es bereits ein Spieler mit diesen Daten gibt. Ist die Prüfung erfolgreich erscheint vor dem Spieler das Hauptmenü. Ist sie nicht erfolgreich kehrt der Spieler zum Login-Menü zurück.

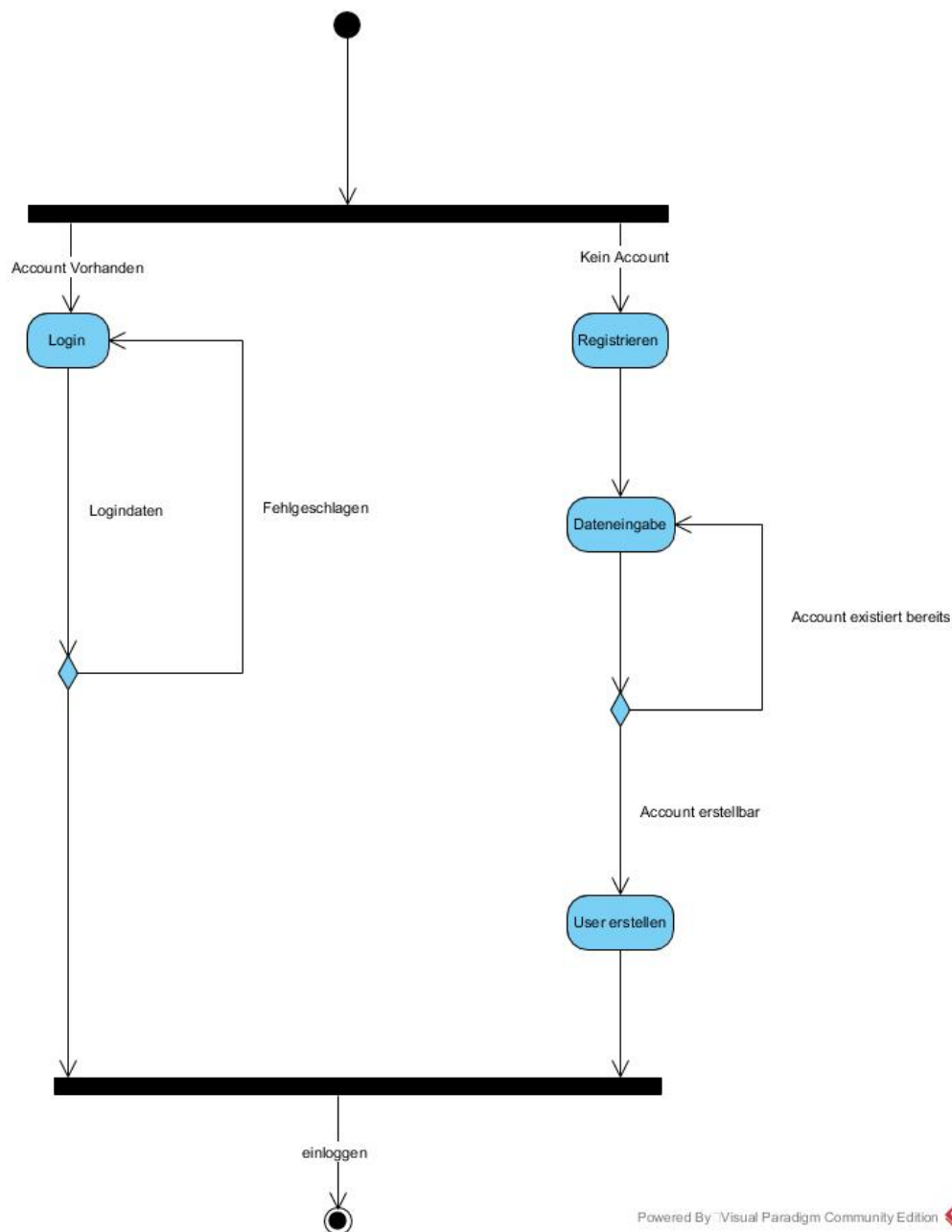


Abbildung 13: Login - Sequenzdiagramm



### 3.2 Logout

Nachdem ein Spieler seinen letzten Zug gemacht hat und das Spiel beenden möchte wird eine Nachricht an den Server geschickt, die Verbindung zum Client des Spielers zu beenden. Der Server schickt daraufhin eine Nachricht an den Client des anderen Spielers, dass der Gegner das Spiel verlassen hat. Zusätzlich speichert der Server die Spieldaten der beiden Spieler in der Datenbank.

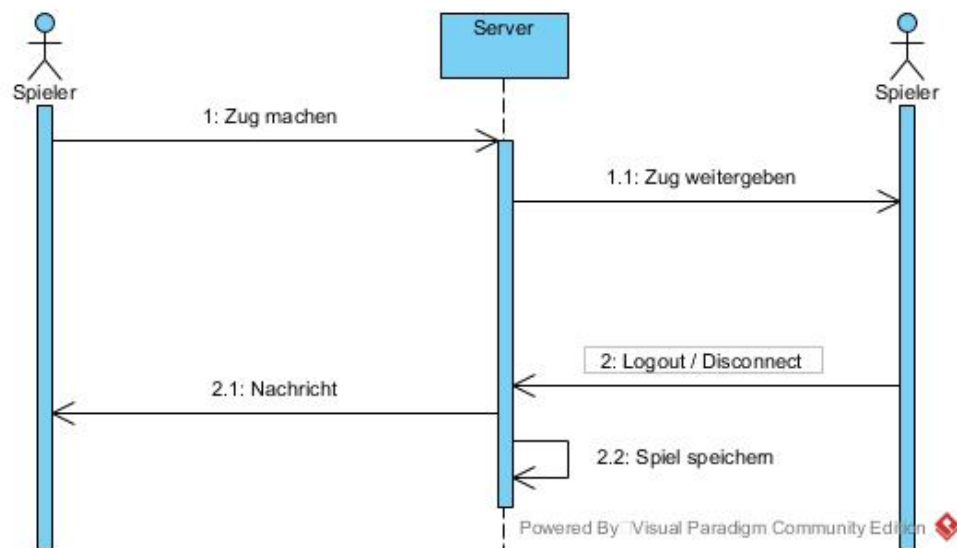


Abbildung 14: Logout - Sequenzdiagramm

### 3.3 Spielraum

Der Spielraum ist eine Liste die in einem eigenen Menüfenster angezeigt wird. Hier ist zu sehen, welche Spieler aktuelle online sind. Spieler die online sind können hier direkt zum Spiel herausgefordert werden.



Abbildung 15: Spielraum

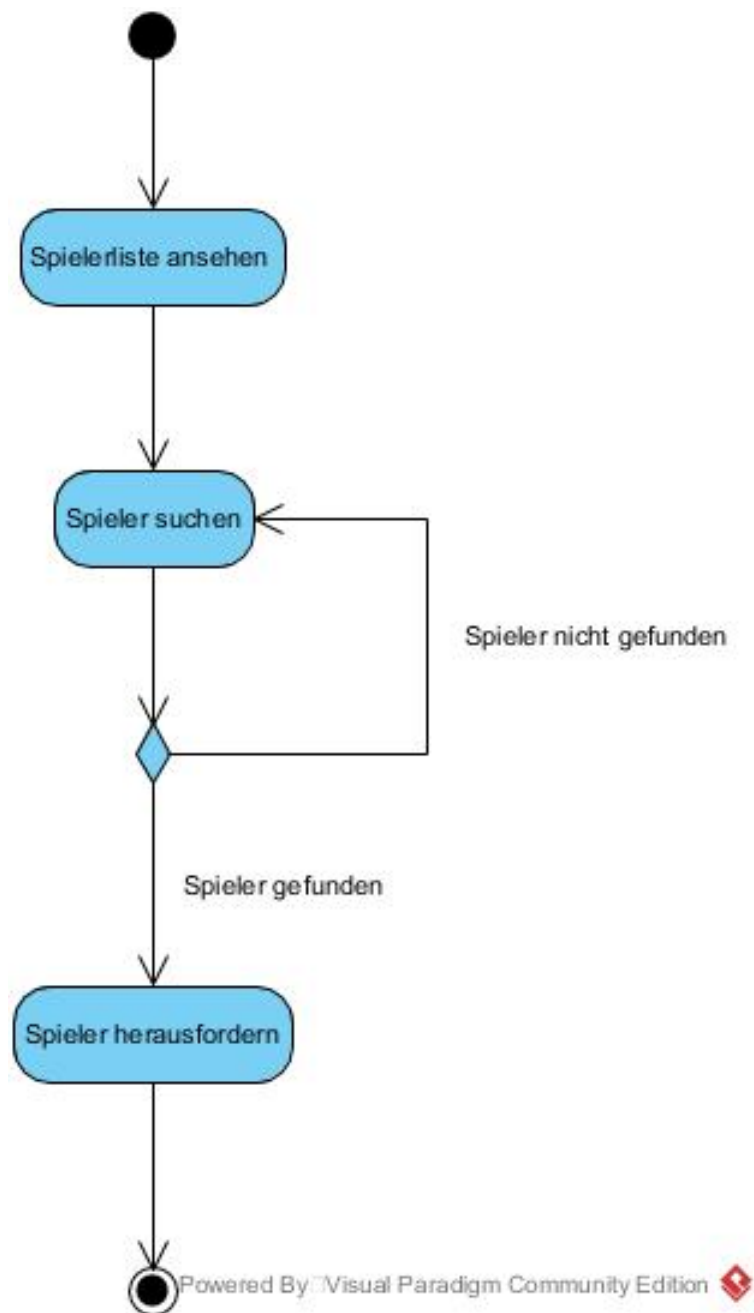


Abbildung 16: Spielraum - Sequenzdiagramm

### 3.4 TicTacToi

Das Spielfeld ist der Ort an dem die Gegner sich schlussendlich, virtuell treffen. In seiner Standardausführung von 3x3-Felder können zwei Spieler gemäß den Regeln von TicTacToi gegeneinander antreten. Die Auswahl eine größeren Spielfeldes und somit das Aufeinandertreffen mehrerer Gegner ist möglich.

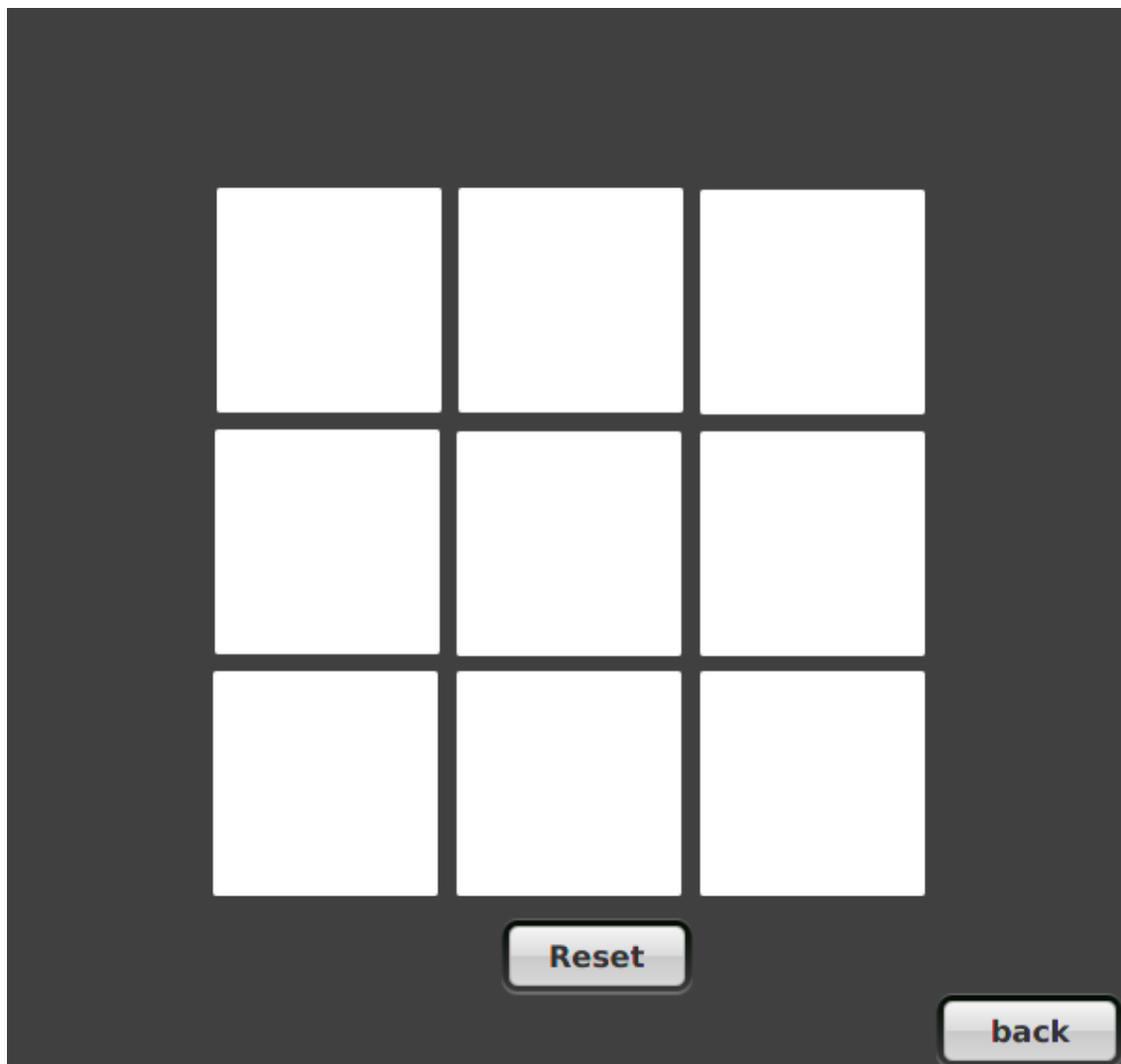


Abbildung 17: TicTacToi

### 3.5 Spielverlauf

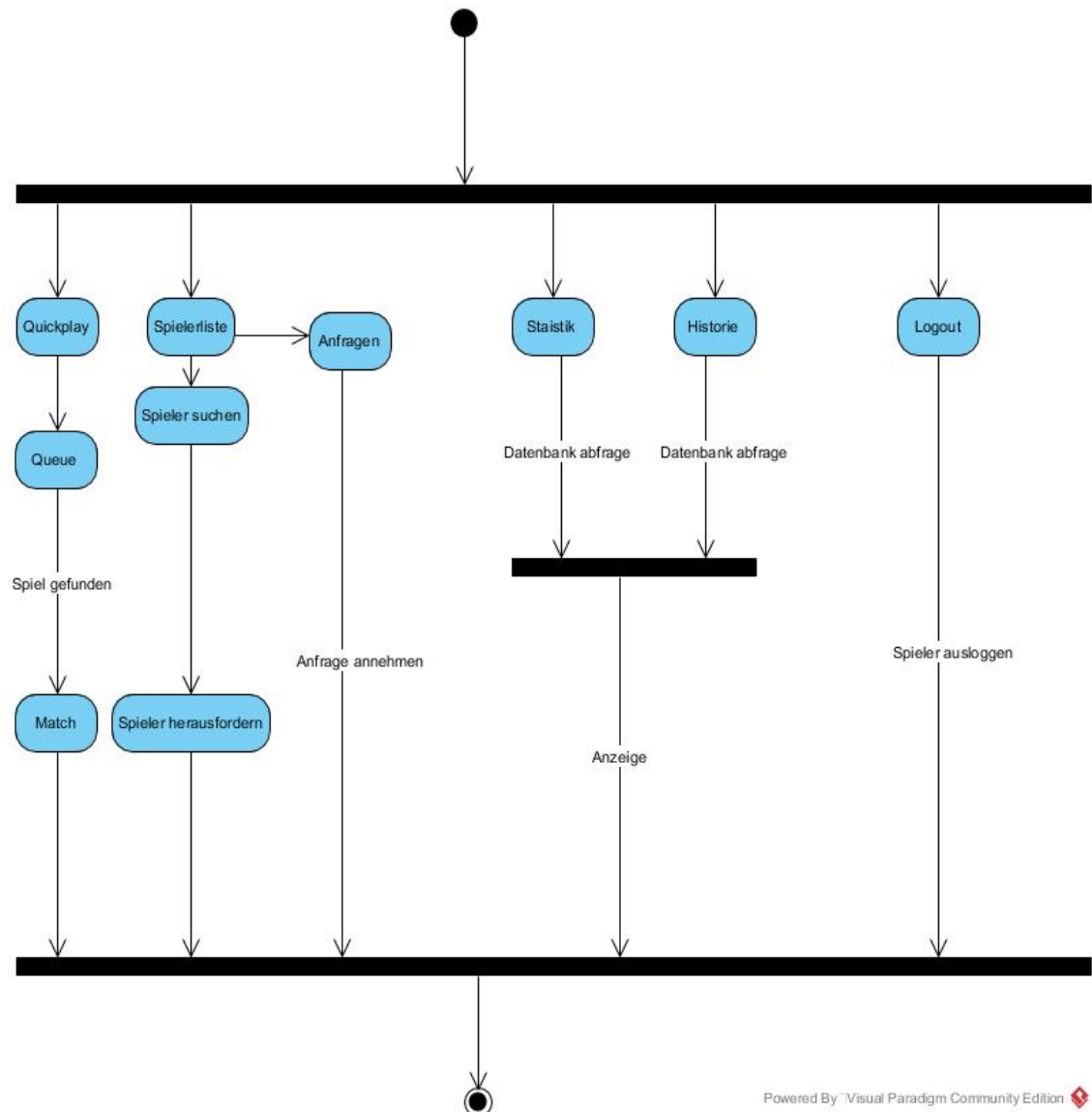


Abbildung 18: Spielverlauf - Sequenzdiagramm

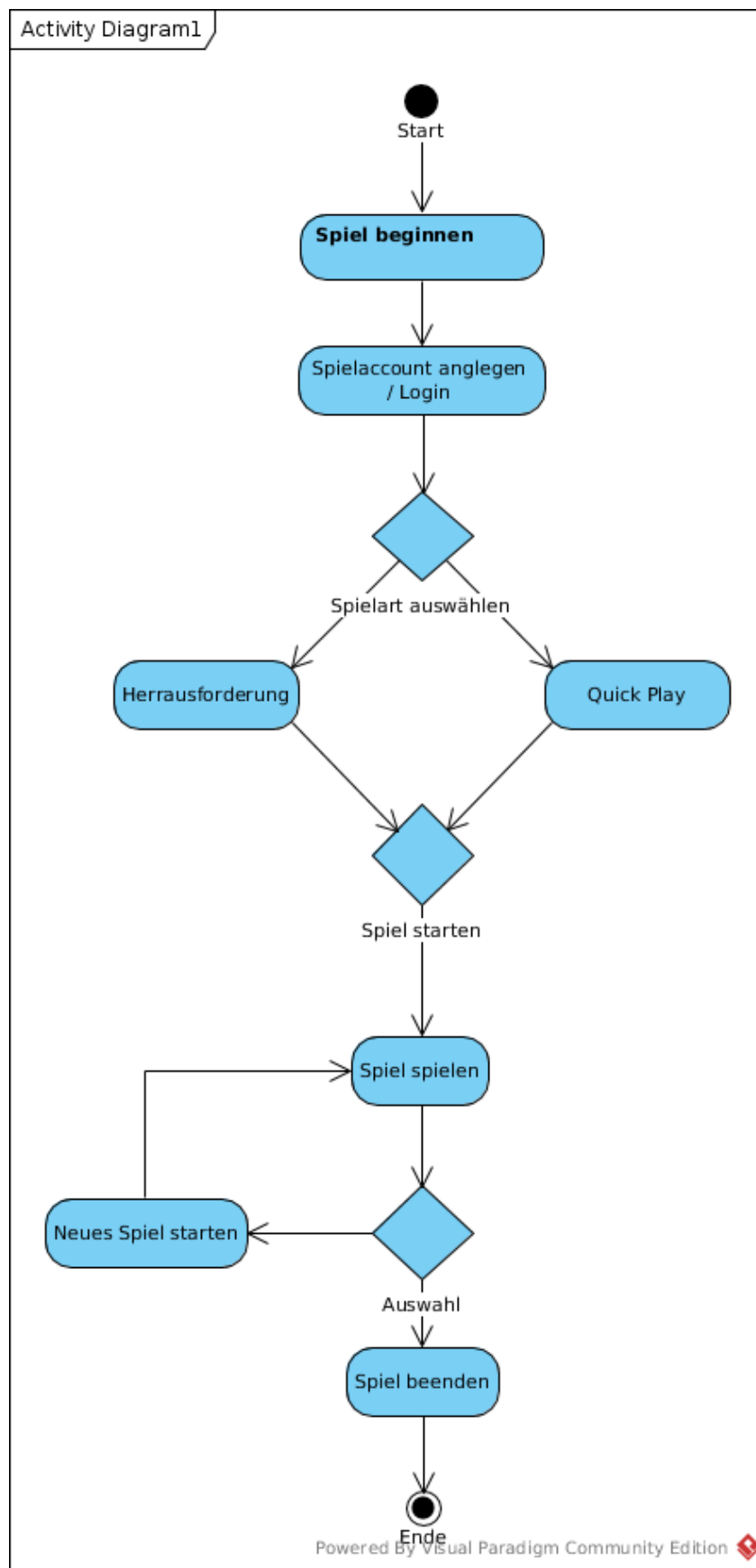


Abbildung 19: Spielverlauf - Aktivitätsdiagramm

## **4 Lastenheft für TicTacToe-Online**

### **4.1 Genutzte Technologien**

Programmiersprache: Java

Datenbank: MySQL

Kommunikation: Discord, WhatsApp

Datenaustausch: Github

Aufgabenverteilung: Trello

Schnittstelle: Javsockets

### **4.2 Zielbestimmung**

Die zu entwickelnde Software soll die Möglichkeit bieten das Spiel TicTacToe mit anderen Spielern über das Internet zu spielen.

### **4.3 Zielgruppen**

Die Zielgruppe des Produkts sind alle die, die sich für das Spiel TicTacToe interessieren.

## 4.4 Produktübersicht

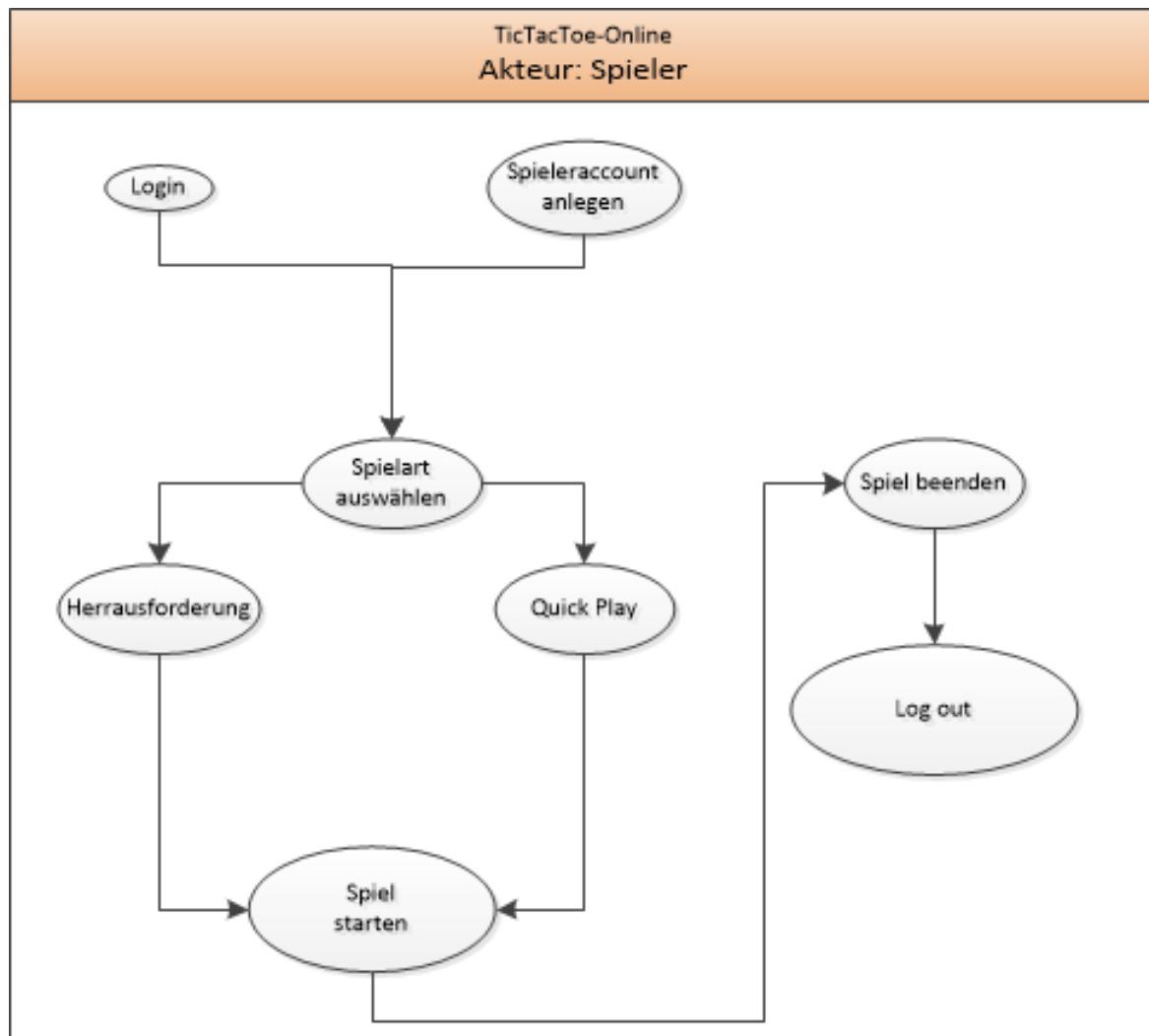


Abbildung 20: TicTacToe-Online



## **4.5 Produktfunktionen**

### **4.5.1 LF10**

Anwendungsfall: Spieleraccount anlegen

Akteur: Spieler

Beschreibung: Ein Spieler, der neu in das Spiel dazu kommt, legt sich über die Client-Software einen Spieleraccount an. Dazu sind eine gültige E-Mail-Adresse, ein Passwort und einen Nickname für das Spiel notwendig. Die eingegebenen Daten werden in einer Datenbank hinterlegt.

### **4.5.2 LF20**

Anwendungsfall: Login

Akteur: Spieler

Beschreibung: Ein Spieler, der bereits einen Spieleraccount hat, logt sich mit der Eingabe von Nickname und Passwort über die Client-Software in das Spiel ein.

### **4.5.3 LF30**

Anwendungsfall: Spielart auswählen

Akteur: Spieler

Beschreibung: Bevor ein Spiel gestartet werden kann, muss sich der Spieler zwischen der Spielart Quick Play und der Spielart Herausforderung entscheiden. Quick Play ermöglicht einen sofortigen Spielstart gegen einen zufällig ausgewählten Gegner. Herausforderung ermöglicht es gezielt gegen einen bestimmten, selbst ausgewählten Gegner anzutreten.

### **4.5.4 LF40**

Anwendungsfall: Spiel starten

Akteur: Spieler

Beschreibung: Die ausgewählten Spieler werden auf das Spielfeld weiter geleitet und können das Spiel beginnen.

#### **4.5.5 LF50**

Anwendungsfall: Spiel beenden

Akteur: Spieler

Beschreibung: Nachdem der Sieger feststeht, können sich die Spieler über den Logout-Button aus dem Spiel ausloggen und mit einem weiteren Klick auf den Button Quit den Spielclient beenden.

#### **4.5.6 LF60**

Anwendungsfall: Kontrolle der Züge

Akteur: Server

Beschreibung: Der Server kontrolliert ob die Züge korrekt abgelaufen sind. Korrekt bedeutet, ob ein Spieler ein Feld auswählen will bei dem dies zulässig ist. Es gilt zu verhindern, dass ein Feld welches bereits mit einem X oder Kreis versehen ist dem Server zweimal ausgewählt bekannt gemacht wird.

#### **4.5.7 LF70**

Anwendungsfall: Verwaltung der Datenbank

Akteur: Server

Beschreibung: Die Daten der Spieler und ihrer einzelnen Spielzüge sind zu speichern und zu verwalten. Im Fall, dass ein gespeichertes Spiel später fortgesetzt wird muss eine exakte Rekonstruktion aus den gespeicherten Daten möglich sein. Beim Aufruf der Spielerstatistiken muss eine korrekte Anzeige der Punktestände gewährleistet werden.

#### **4.5.8 LF80**

Anwendungsfall: Öffnen von einer oder mehrerer Spielesessions

Akteur: Server

Beschreibung: Das Spiel ist darauf ausgelegt, dass auch mehr als zwei Spieler sich gleichzeitig auf dem Server zu Spielen einfinden können.

**4.5.9 LF90**

Anwendungsfall: User erstellen

Akteur: Datenbank

Beschreibung: Die Daten der einzelnen Spieler werden, jeweils beim Anlegen eines neuen Spielers, in einer MySQL-Datenbank gespeichert. Zu den Spielerdaten gehören:

- 1 Spielername
- 2 E-Mail-Adresse
- 3 Passwort

**4.5.10 LF100**

Anwendungsfall: User verwalten

Akteur: Datenbank

Beschreibung: Neben dem anlegen eines neuen Spielers muss ein bereits bestehender Spieler die Möglichkeit haben, seinen Account zu verwalten und falls nötig auch seine Spielerdaten zu ändern.

**4.5.11 LF110**

Anwendungsfall: Spiel speichern

Akteur: Datenbank

Beschreibung: Wenn ein laufendes Spiel von den Spielern unterbrochen werden muss, können sie ihren bisherigen Spielstand speichern um das Spiel später zu beenden. In diesem Fall werden alle Daten des bisherigen Spiels in der Datenbank gespeichert um das unterbrochene Spiel später wieder herstellen zu können.

## 4.6 Produktdaten

- 1 LD10 Login-Daten
- 2 LD20 Statistiken der User
- 3 LD30 Spielverläufe

## 4.7 Produktleistungen

LL10 Alle Reaktionszeiten auf Benutzerinteraktionen müssen unter 0,5 Sekunden liegen.

## 4.8 Qualitätsanforderungen

	Sehr gut	Gut	Normal	Nicht relevant
Funktionalität		X		
Zuverlässigkeit			X	
Benutzbarkeit	X			
Effizienz			X	
Änderbarkeit			X	
Übertragbarkeit			X	

## **4.9 Glossar TicTacToe-Online**

### **4.9.1 Spieleraccount**

Der Spieleraccount stellt die virtuelle Identität eines Spielers dar. Die Informationen aller Spiele an denen der Spieler beteiligt ist, werden mit seinem Account verknüpft.

### **4.9.2 Spielerdaten**

Daten des Spieler die erfasst werden müssen um einen Spieleraccount anzulegen. Die notwendigen Daten sind die E-Mail-Adresse, das Passwort und der Nickname.

### **4.9.3 Login**

Der Spieler startet die Client-Software wo er sich, sofern noch nicht geschehen, einen Spieleraccount anlegt oder sich gleich mit seinen Login-Daten, Nickname und Passwort, einloggt. Bei dem Login wird die Verbindung zwischen der Client-Software und dem Server hergestellt.

### **4.9.4 Logout**

Nach dem Abschluss seines letzten gewünschten Spiels logt sich der Spieler aus dem Spiel aus. Die Verbindung zwischen Client-Software und Server wird dabei getrennt.

## 5 Usecases

### 5.1 UC 10 – Login

<b>Name:</b>	UC 10 – Login
<b>Ziel:</b>	Die Logindaten des Spielers müssen an den Server geschickt und von diesem auf ihre Authentizität überprüft werden.
<b>Kategorie:</b>	Primär
<b>Vorbedingung:</b>	<ul style="list-style-type: none"> <li>➤ Der Server muss 24 Stunden am Tag 7, Tage die Woche, 365 Tage im Jahr online sein.</li> <li>➤ Spieler möchte sich einloggen</li> </ul>
<b>Nachbedingung-Erfolg:</b>	Beispiel: <ul style="list-style-type: none"> <li>➤ „Der Login des Spielers war erfolgreich.“</li> </ul>
<b>Nachbedingung-Fehl-schlag</b>	Beispiele: <ul style="list-style-type: none"> <li>➤ Zu den eingegebenen Benutzerdaten existiert kein Account.</li> <li>➤ Die Benutzerdaten wurden fehlerhaft oder unvollständig eingegeben.</li> </ul>
<b>Akteure:</b>	Spieler, Server
<b>Auslösendes Ereignis:</b>	Spieler startet Client-Software auf seinem PC.
<b>Beschreibung:</b>	<ul style="list-style-type: none"> <li>➤ Spieler startet Client-Software auf seinem PC.</li> <li>➤ Spieler gibt Logindaten ein.</li> <li>➤ Logindaten werden an den Server geschickt</li> <li>➤ Server überprüft Logindaten</li> <li>➤ Login erfolgreich</li> <li>➤ Spieler wird zum Hauptmenü weiter geleitet.</li> </ul>
<b>Bemerkungen:</b>	<ul style="list-style-type: none"> <li>➤ XXX</li> </ul>
<b>Alternative:</b>	<ul style="list-style-type: none"> <li>➤ Spieler startet Client-Software auf seinem PC.</li> <li>➤ Spieler gibt Logindaten ein.</li> <li>➤ Logindaten werden an den Server geschickt</li> <li>➤ Server überprüft Logindaten</li> <li>➤ Login fehlgeschlagen siehe Nachbedingung-Fehl-schlag</li> <li>➤ Spieler wird zum Loginmenü zurück geleitet.</li> </ul>
<b>Bemerkung:</b>	<ul style="list-style-type: none"> <li>➤ XXX</li> </ul>

### 5.1.1 UC 20 – Registrieren

<b>Name:</b>	UC 20 – Registrieren
<b>Ziel:</b>	Ein neuer Spieler soll seine Benutzerdaten an Server schicken können um einen neuen Benutzer anzulegen.
<b>Kategorie:</b>	Primär
<b>Vorbedingung:</b>	<ul style="list-style-type: none"> <li>➤ Der Server muss 24 Stunden am Tag 7, Tage die Woche, 365 Tage im Jahr online sein.</li> <li>➤ Ein neuer Spieler möchte sich registrieren.</li> </ul>
<b>Nachbedingung-Erfolg:</b>	Beispiel: <ul style="list-style-type: none"> <li>➤ „Der Benutzeraccount wurde erfolgreich angelegt.“</li> </ul>
<b>Nachbedingung-Fehlschlag</b>	Beispiele: <ul style="list-style-type: none"> <li>➤ Die eingegebene E-Mail-Adresse ist nicht gültig.</li> <li>➤ Für Benutzername und/oder Passwort wurden unzulässige Zeichen verwendet.</li> <li>➤ Der gewählte Benutzername ist schon vergeben.</li> </ul>
<b>Akteure:</b>	Spieler, Server
<b>Auslösendes Ereignis:</b>	Spieler startet Client-Software auf seinem PC.
<b>Beschreibung:</b>	<ul style="list-style-type: none"> <li>➤ Spieler startet Client-Software auf seinem PC.</li> <li>➤ Spieler gibt Benutzerdaten ein.</li> <li>➤ Benutzerdaten werden an den Server geschickt</li> <li>➤ Server überprüft Benutzerdaten</li> <li>➤ Registrierung erfolgreich</li> <li>➤ Spieler wird zum Hauptmenü weiter geleitet.</li> </ul>
<b>Bemerkungen:</b>	<ul style="list-style-type: none"> <li>➤ XXX</li> </ul>
<b>Alternative:</b>	<ul style="list-style-type: none"> <li>➤ Spieler startet Client-Software auf seinem PC.</li> <li>➤ Spieler gibt Benutzerdaten ein.</li> <li>➤ Benutzerdaten werden an den Server geschickt</li> <li>➤ Server überprüft Benutzerdaten</li> <li>➤ Registrierung fehlgeschlagen siehe Nachbedingung-Fehlschlag</li> <li>➤ Spieler wird zum Registrierungs Menü zurück geleitet.</li> </ul>
<b>Bemerkung:</b>	<ul style="list-style-type: none"> <li>➤ XXX</li> </ul>

## 5.2 UC 30 User erstellen

<b>Name:</b>	UC 30 – User erstellen
<b>Ziel:</b>	Ein Datenbankeintrag für einen neu registrierten Spieler soll angelegt werden.
<b>Kategorie:</b>	Primär
<b>Vorbedingung:</b>	<ul style="list-style-type: none"> <li>➤ Der Server muss 24 Stunden am Tag 7, Tage die Woche, 365 Tage im Jahr online sein.</li> <li>➤ Ein neuer Spieler möchte sich registrieren.</li> </ul>
<b>Nachbedingung-Erfolg:</b>	Beispiel: <ul style="list-style-type: none"> <li>➤ „Der Benutzeraccount wurde erfolgreich angelegt.“</li> </ul>
<b>Nachbedingung-Fehl-schlag</b>	Beispiele: <ul style="list-style-type: none"> <li>➤ Die eingegebene E-Mail-Adresse ist nicht gültig.</li> <li>➤ Für Benutzername und/oder Passwort wurden unzulässige Zeichen verwendet.</li> <li>➤ Der gewählte Benutzername ist schon vergeben.</li> </ul>
<b>Akteure:</b>	Server
<b>Auslösendes Ereignis:</b>	Der Spieler schickt die eingegebenen Benutzerdaten vom Client an den Server.
<b>Beschreibung:</b>	<ul style="list-style-type: none"> <li>➤ Der Spieler schickt die eingegebenen Benutzerdaten vom Client an den Server.</li> <li>➤ Server überprüft die Benutzerdaten</li> <li>➤ Benutzerdaten sind korrekt.</li> <li>➤ Die Benutzerdaten werden als neuer Datensatz in die Datenbank geschrieben.</li> <li>➤ Der Server schickt die Erfolgsmeldung an den Client des Spielers.</li> </ul>
<b>Bemerkungen:</b>	<ul style="list-style-type: none"> <li>➤ XXX</li> </ul>
<b>Alternative:</b>	<ul style="list-style-type: none"> <li>➤ Der Spieler schickt die eingegebenen Benutzerdaten vom Client an den Server.</li> <li>➤ Server überprüft die Benutzerdaten</li> <li>➤ Benutzerdaten sind ungültig.</li> <li>➤ Die Benutzerdaten werden verworfen.</li> <li>➤ Der Server schickt eine Fehlermeldung an den Client des Spielers.</li> </ul>
<b>Bemerkung:</b>	<ul style="list-style-type: none"> <li>➤ XXX</li> </ul>



### 5.3 UC 40-Spielerliste ansehen

<b>Name:</b>	UC 40 – Spielerliste ansehen
<b>Ziel:</b>	Die Liste aller Spieler die aktuell online sind, soll angezeigt werden.
<b>Kategorie:</b>	Primär
<b>Vorbedingung:</b>	<ul style="list-style-type: none"> <li>➤ Der Server muss 24 Stunden am Tag 7, Tage die Woche, 365 Tage im Jahr online sein.</li> <li>➤ Ein neuer Spieler möchte die Spielerliste einsehen.</li> </ul>
<b>Nachfolgebedingung-Erfolg:</b>	Beispiel: <ul style="list-style-type: none"> <li>➤ Die Spielerliste wird als Fenster innerhalb der Client-Software angezeigt.</li> </ul>
<b>Nachbedingung-Fehlschlag</b>	Beispiele: <ul style="list-style-type: none"> <li>➤ Es konnte von der Clientsoftware keine Verbindung zum Server hergestellt werden.</li> <li>➤ Der Server ist offline.</li> </ul>
<b>Akteure:</b>	Server, Clientsoftware
<b>Auslösendes Ereignis:</b>	Der Spieler ruft die Spielerliste auf.
<b>Beschreibung:</b>	<ul style="list-style-type: none"> <li>➤ Der Spieler ruft über die Clientsoftware die Spielerliste auf.</li> <li>➤ Der Client stellt eine Verbindung zum Server her.</li> <li>➤ Der Onlinestatus der Spieler wird aus der Datenbank gelesen.</li> <li>➤ Die Daten der Spieler, die online sind, werden an den Client geschickt.</li> <li>➤ Der Client stellt die empfangenen Daten in einer Tabelle dar.</li> </ul>
<b>Bemerkungen:</b>	<ul style="list-style-type: none"> <li>➤ XXX</li> </ul>
<b>Alternative:</b>	<ul style="list-style-type: none"> <li>➤ Der Spieler ruft über die Clientsoftware die Spielerliste auf.</li> <li>➤ Der Client stellt eine Verbindung zum Server her.</li> <li>➤ Die Verbindung zum Server kann nicht hergestellt werden. Siehe Nachfolgebedingung-Fehlschlag.</li> </ul>
<b>Bemerkung:</b>	<ul style="list-style-type: none"> <li>➤ XXX</li> </ul>

### 5.3.1 UC 50-Spieler herausfordern

<b>Name:</b>	UC 50 – Spieler herausfordern
<b>Ziel:</b>	Basierend auf der erfolgreichen Durchführung von UC 40. Ein Spieler aus der Spielerliste soll zu einem Spiel herausgefordert werden
<b>Kategorie:</b>	Primär
<b>Vorbedingung:</b>	<ul style="list-style-type: none"> <li>➤ Der Server muss 24 Stunden am Tag 7, Tage die Woche, 365 Tage im Jahr online sein.</li> <li>➤ Ein Spieler möchte einen anderen Spieler herausfordern.</li> </ul>
<b>Nachbedingung-Erfolg:</b>	Beispiel: <ul style="list-style-type: none"> <li>➤ „Der Gegner hat die Herausforderung angenommen.“</li> </ul>
<b>Nachbedingung-Fehlschlag</b>	Beispiele: <ul style="list-style-type: none"> <li>➤ „Der Gegner hat die Herausforderung abgelehnt.“</li> <li>➤ Die Verbindung von einem bzw. beiden Spieleclients zum Server wurde unterbrochen.</li> </ul>
<b>Akteure:</b>	Server. Clientsoftware, Spieler
<b>Auslösendes Ereignis:</b>	Ein Spieler wählt den von ihm gewünschten Gegner aus der Spielerlist und schickt ihm eine Herausforderung.
<b>Beschreibung:</b>	<ul style="list-style-type: none"> <li>➤ Ein Spieler wählt den von ihm gewünschten Gegner aus der Spielerliste.</li> <li>➤ Der Spieler schickt seinem potentiellen Gegner eine Herausforderung.</li> <li>➤ Der Gegner nimmt die Herausforderung an.</li> <li>➤ Beide Spieler werden auf das eigentliche Spielfeld weiter geleitet.</li> </ul>
<b>Bemerkungen:</b>	<ul style="list-style-type: none"> <li>➤ XXX</li> </ul>
<b>Alternative:</b>	<ul style="list-style-type: none"> <li>➤ Ein Spieler wählt den von ihm gewünschten Gegner aus der Spielerliste.</li> <li>➤ Der Spieler schickt seinem potentiellen Gegner eine Herausforderung.</li> <li>➤ Das Spiel kommt nicht zustande.</li> <li>➤ Siehe Nachbedingung-Fehlschlag</li> <li>➤ Die Spieler werden zum Loginmenü zurück geschickt.</li> </ul>
<b>Bemerkung:</b>	<ul style="list-style-type: none"> <li>➤ XXX</li> </ul>

## 5.4 UC 60 Logout

<b>Name:</b>	UC 60 – Logout
<b>Ziel:</b>	Nach dem letzten Spiel soll die Verbindung zwischen Client und Server getrennt werden
<b>Kategorie:</b>	Primär
<b>Vorbedingung:</b>	<ul style="list-style-type: none"> <li>➤ Der Server muss 24 Stunden am Tag 7, Tage die Woche, 365 Tage im Jahr online sein.</li> <li>➤ Ein Spieler möchte die Verbindung zwischen seiner Clientsoftware und dem Server beenden.</li> </ul>
<b>Nachbedingung-Erfolg:</b>	Beispiel: <ul style="list-style-type: none"> <li>➤ „Die Verbindung zum Server wurde getrennt.“</li> </ul>
<b>Nachbedingung-Fehl-schlag</b>	Beispiele: <ul style="list-style-type: none"> <li>➤ „Die Verbindung konnte nicht getrennt werden.“</li> </ul>
<b>Akteure:</b>	Server. Clientsoftware, Spieler
<b>Auslösendes Ereignis:</b>	Das letzte Spiel zwischen zwei Spielern wurde beendet.
<b>Beschreibung:</b>	<ul style="list-style-type: none"> <li>➤ Ein Spieler verlässt das Spielfeld und kehrt zum Login-menü zurück.</li> <li>➤ Die Verbindung zum Server wird getrennt.</li> </ul>
<b>Bemerkungen:</b>	<ul style="list-style-type: none"> <li>➤ XXX</li> </ul>
<b>Alternative:</b>	<ul style="list-style-type: none"> <li>➤ Ein Spieler beendet seine Clientsoftware.</li> <li>➤ Die Verbindung zum Server wird getrennt.</li> </ul>
<b>Bemerkung:</b>	<ul style="list-style-type: none"> <li>➤ XXX</li> </ul>