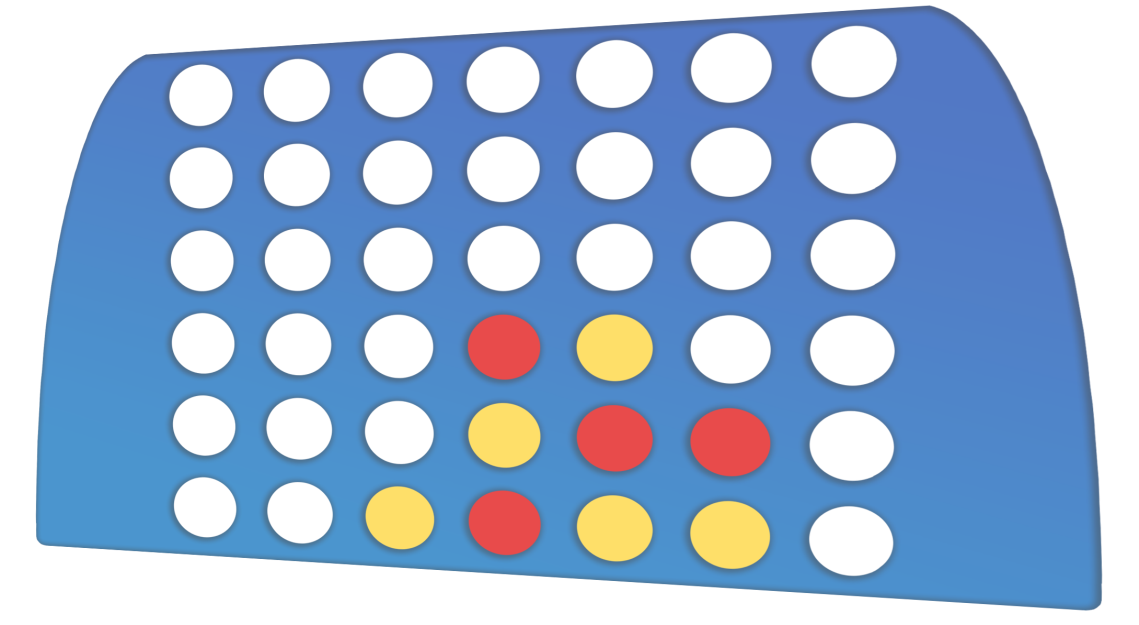
# Entwicklungsdokumentation

Duale Hochschule Baden-Württemberg Mannheim

- Vier gewinnt -

Bearbeitungszeitraum xx. September 2015 – xx. April 2016

****Verfasser: Marcel Cornesse

Dominik Efinger

Irena Männle

Luis Oberste

Shakira Steger

Kurs: WWI 13 SCA

Vorlesung: WI-Projekt

Dozent: Prof. Dr. Matthias Lauterbach

Inhalt

[1 Spezifikation 2](#_Toc433027765)

[1.1 Softwareumgebung 2](#_Toc433027766)

[1.2 Projekt- und Ziel-Software 2](#_Toc433027767)

[1.2.1 Anforderungsbeschreibung und Hilfsmittel 2](#_Toc433027768)

[1.2.2 Funktionale Anforderungen 3](#_Toc433027769)

[1.2.3 Lieferumfang 4](#_Toc433027770)

[1.2.4 Nichtfunktionale Anforderungen 5](#_Toc433027771)

[2 Design 6](#_Toc433027772)

[2.1 Prozess der Softwareentwicklung 6](#_Toc433027773)

[2.1.1 Vorgehensmodell (agiles/scrum) 6](#_Toc433027774)

[2.1.2 Meilensteine 7](#_Toc433027775)

[2.1.3 Verantwortlichkeiten 7](#_Toc433027776)

[2.2 Aufbau, Entwurfsentscheidungen 8](#_Toc433027777)

[2.2.1 Files & folders 8](#_Toc433027778)

[2.2.2 Schichtenarchitektur 9](#_Toc433027779)

[2.2.3 Aktivitätendiagramm 9](#_Toc433027780)

[2.3 Konzept 9](#_Toc433027781)

[2.3.1 Umsetzung der Anforderungsliste? 9](#_Toc433027782)

[2.3.2 Mockups 10](#_Toc433027783)

[2.4 Struktur 12](#_Toc433027784)

[2.4.1 Entitäten (Komponenten, Module, Pakete, Klassen) 12](#_Toc433027785)

[2.4.2 Deren Beziehungen und Abhängigkeiten 12](#_Toc433027786)

[2.4.3 Datenbank (ER-Diagramm) 12](#_Toc433027787)

[2.4.4 Klassen beschreiben 12](#_Toc433027788)

Abschnitt: Entwicklung

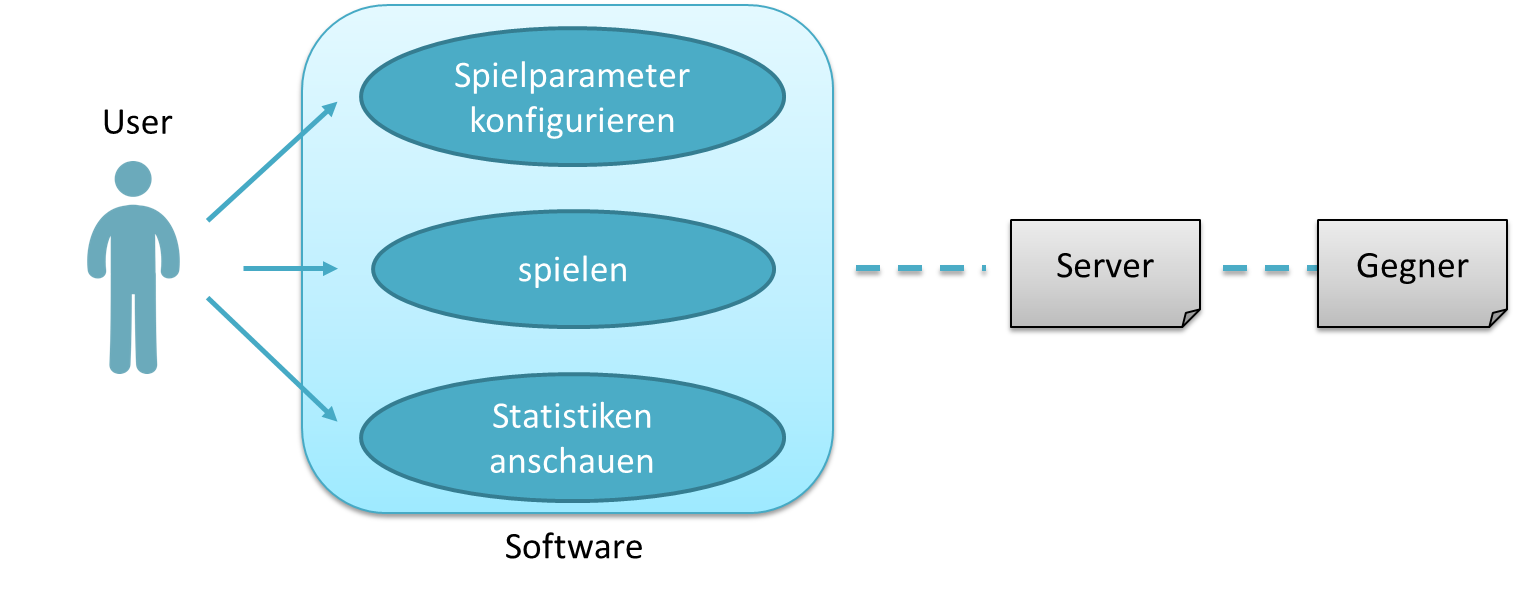
# 1 Spezifikation

## Softwareumgebung

Bietet programm das 4 gewinnt spielt um was geht’s.

|  |  |
| --- | --- |
| Use Case Name: | **4-Gewinnt** |
| Kurzbeschreibung | Der User nutzt die Software, um Voreinstellungen zu treffen und gegen einen Gegner mittels zentraler Serverkommunikation zu spielen, sowie Statistiken einzusehen. |
| Verantwortliche Entwickler | Fungi-Software Solutions |
| Spieler | Agent-Fungi |
| Ziele | Spielen, Statistik anschauen |
| Vorbedingungen | Server der Kommunikation zum Gegner steuert, Gegner-Agent, gegen den gespielt wird |
| Szenario | Standard Szenario |
| Schritte | 1. Spielparameter konfigurieren  2. spielen  3. Statistiken anschauen |
| Fehlerbehandlung | Abgedeckt durch Try-catch und  Array out of bound insbesondere bei KI |

In nachfolgender Graphik wird der zuvor beschriebenen Use Case 4-Gewinnt dargestellt. Als Vorbedingung gilt es den Server der Kommunikation zum Gegner (Gegner-Agent) zu steuern Der User kann mit der Software sowohl Spielparameter konfigurieren, das eigentliche 4-Gewinnt-Spiel zu spielen sowie im Nachhinein die Statistiken betrachten.



## Projekt- und Ziel-Software

## Anforderungsbeschreibung und Hilfsmittel

Nach Angaben des Dozenten, galt es folgende Anforderungen innerhalb des Bearbeitungszeitraumes eigenverantwortlich umzusetzen. Um einen ersten Überblick des Projektumfangs zu erhalten ist im Folgenden die Kurzfassung der vorgegebenen Aufgabenbeschreibung aufgelistet:

* Bauen eines Software-Agenten, der das Spiel „4 gewinnt“ gegen einen anderen Spieler autonom spielen kann
* Der eigene Agent kommuniziert mit einem Server-Programm, mit dem auch der Gegenspieler, d. h. der Agent des anderen Teams, verbunden ist.
* Im Rahmen der vorgegebenen Anforderungen soll der Bau des Agenten selbstständig im Team durchgeführt werden. Das bedeutet insbesondere:
  + eigenverantwortliches managen des Projektes
  + die Lösung entsprechend eigener Vorstellungen gestalten
  + Anwenden einer Methodik nach eigenem Ermessen
  + Implementierungsdetails weitgehend ebenfalls nach eigenem Ermessen
* Der Rahmen der vorgegebenen Anforderungen wird maßgeblich durch die wichtigsten zu liefernden Ergebnisse bestimmt:
  + Agent (Prototyp, Beta und Release)
  + Dokumentation
  + Turnierteilnahme

Ebenfalls wurden die für das Projekt zugelassenen Hilfsmittel im Voraus durch den Dozenten festgelegt und setzten sich aus den folgenden sechs Quellen zusammen:

* Java-Tutorial
* Java-Bücher
* Java-Api-Doc
* XML-Spezifikation
* HSQLDB-User Guide
* bereitgestellte Bausteine

## Funktionale Anforderungen

Im Folgenden werden die umzusetzenden funktionalen Anforderungen aufgelistet. Diese wurden ebenfalls durch den Dozenten vorgegeben.

* Funktionalität
  + Fehlerfreier Ablauf des zentralen Geschäftsprozesses, dem Ablauf eines Satzes des Spiels zwischen zwei Agenten über den Server auf der Basis jeder der vorgegebenen Schnittstellen.
  + Die Funktionsfähigkeit des Agenten wird erstens durch die erfolgreiche Turnierteilnahme sowie zweitens durch das auszuliefernde Release nachgewiesen
* Technologien
  + J2SE (nach Maßgabe der Version im jeweiligen PC-Raum)
  + Swing (alternativ JavaFx)
  + HSQLDB
* Benutzerschnittstelle
  + grafische Oberfläche
  + Anzeige des Satzstatus
  + Anzeige des Spielstandes
  + Anzeige Spielfeldes
  + Anzeige aller Züge des aktuellen Satzes auf dem Spielfeld
  + Realisierung auf der Basis von JavaFx oder Swing
* Serverschnittstellen
  + Agent und Server kommunizieren zu einem Zeitpunkt über eine von zwei Schnittstellen, d. h. über die Datei-Schnittstelle oder die Push-Schnittstelle.
  + Während eines Satzes erfolgt kein Wechsel der Schnittstelle zwischen den Sätzen, d. h. also auch während eines Spiels, kann die Schnittstelle gewechselt werden.
  + Datei-Schnittstelle: Kommunikation von Server und Agents über Dateien auf der Basis von Streams.
  + Push-Schnittstelle: Kommunikation von Server und Agents über Events auf der Basis von WebSockets.
  + Genauere Definitionen bezüglich der Schnittstellen sind der Anforderungsbeschreibung des Dozenten zu entnehmen.
* Anwendungslogik
  + Die Anforderungen an die Anwendungslogik des Agenten werden in Form von Szenarien für das Spielen von Sätzen beschrieben. Diese Szenarien sind der Anforderungsbeschreibung des Dozenten zu entnehmen.
* Datenhaltung
  + Speichern der Daten in einer Datenbank
  + automatische Speicherung der Daten durch den Agenten
  + zu speichernde Daten sind:
    - Gegner, Startspieler, Sieger, Punkte, Spiele, Sätze, Züge
    - minimal: alle Daten nach Maßgabe des Punktemodus
  + Realisierung auf der Basis von HSQLDB
  + wenigstens drei Abfragen bzw. Abfragevarianten implementieren;
  + Minimalanforderung:
    - alle gespielten Spiele anzeigen
    - alle Züge eines frei wählbaren Spiels der angezeigten Spiele anzeigen
  + wenigstens drei Spiele für Beispielabfragen mit Release-Version ausliefern

## Lieferumfang

Im Laufe des Projekts müssen die folgenden Versionen termingerecht abgeliefert werden.

|  |  |
| --- | --- |
| Prototyp  Abgabe:  28.09.15 | * Import muss entsprechend Importanleitung für Dozenten ohne Compilerfehler möglich sein. * Ein auf Studentenrechner lauffähiger und vorzeigbarer Oberflächenprototyp ist gefragt. Die eine oder andere Funktionalität oder * Optional: ein automatischer Demonstrationsablauf |
| Beta  Abgabe: 12.10.15 | * Wie Prototyp zzgl.: mindestens 5 Züge über Server spielen können. Beide Schnittstellen müssen funktionieren. * Zufallszug des eigenen Agents genügt; Siegmustererkennung oder Spalte-ist-voll-Erkennung nicht notwendig. |
| Release  Abgabe:  09.11.15 | * auslieferungsfähiges Bundle (zusammen mit der Dokumentation und dem Quellcode als Menge von Dateien auf CD/DVD) * mit dessen Hilfe ein Anwender auf der Basis der Dokumentation den Agenten installieren, konfigurieren und bedienen kann und ein Entwickler das Programm warten und erweitern kann (Eclipse-Import muss funktionieren). |

## Nichtfunktionale Anforderungen

Die nichtfunktionalen Anforderungen an das Projekt wurden durch den Dozenten wie folgt definiert:

* während der gesamten Projektlaufzeit (d.h. während der Vorlesung) Bereitstellung eines Zugangs zum Teamrepository für den Dozenten
  + Dropbox (Dokumente):

<https://www.dropbox.com/sh/zcf94oay4rgatwo/AAAr3v9L1j9S78Z-awgyyOpEa?dl=0>

* + GitHub (Source-Code):

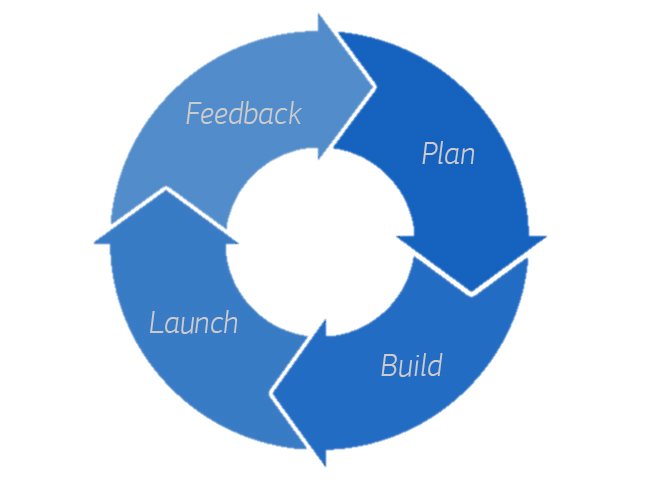
<https://github.com/Steffen911/WIProjekt>

* Die dem Kunden (dem Dozenten) zur Verfügung gestellten Artefakte sollen mit allgemein(st)en Standardprogrammen auf einem normalen MS-Windows-PC angesehen bzw. verwendet werden können. In diesem Sinne geeignete Formate und Werkzeuge sind:
  + \*.java, \*.txt, \*.jpg, \*.bmp, \*.pdf, \*.html, \*.css, \*.jar, \*.zip
  + Open-Office, MS-Office
  + Bitte kein Visio, MS-Projekt, Flash, Silverlight, QuickTime o. ä. und auch keine (anderen) speziellen Plugins oder Clients beim Kunden (Dozenten) voraussetzen.
  + Arbeitsergebnisse, die aus diesen Anforderungen nicht entsprechen, können leider nicht berücksichtigt werden!

# Design

* 1. Prozess der Softwareentwicklung
     1. Vorgehensmodell (agiles/scrum)

Für das Projekt „Fungi“ wurden innerhalb der gemeinsamen Teamarbeit einzelne Aufgabenbereiche zugeteilt und weitgehend selbstständig umgesetzt. Um jedoch einen geordneten Ablauf des Projektes zu garantieren wurde sich für ein agiles Vorgehensmodell entschieden. Diese Art der Vorgehensmodelle wird meist bei Projekten mit überschaubarem Umfang gewählt und ist darauf angewiesen, dass die beteiligten Teammitglieder stetig in Kommunikation über die jeweilige Tätigkeit stehen. In der folgenden Abbildung sind die einzelnen Phasen, in welche das Projekt gegliedert wurde, abgebildet.



Zu Beginn eines jeden Teammeetings wurde der Fortschritt des Projekts besprochen sowie eine detaillierte Rücksprache mittels Feedback der einzelnen Teammitglieder durchgeführt. Anhand der Ergebnisse des Teammeetings konnten sowohl Fort- als auch Rückschritte innerhalb der Excel „Projektstatus“ aktualisiert werden. Daraufhin geht es an die Planung der zu bewältigenden Arbeit bzw. der durchzuführenden Überarbeitungen. Einer der Unterschiede zum klassischen Wasserfallmodell ist das Durchlaufen der einzelnen Phasen, die klassische Variante beginnt erst mit der folgenden Phase nach Abschluss der vorherigen Phase. Bei der agilen Variante kann jedoch zu einer vorherigen Phase zurückgesprungen werden um Verbesserungen bzw. Korrekturen durchzuführen. Dadurch ist es wichtig in ständiger Kommunikation miteinander zu stehen und sich strikt an die Teammeetings zuhalten um das Projekt präzise durchzuführen.

* + 1. Meilensteine

Folgende Abbildung zeigt eine Zeitleiste inklusive der allgemeinen Terminplanung und den festgelegten Meilensteinen die zu Beginn des Projektes gemeinsam innerhalb des Teams festgelegt wurden und einer zeitlichen Planungs-Überblick bieten.

Meilenstein

Der Zeitstrahl zeigt den Start des Projektes am 07.09.2015 sowie dessen Ende am 09.11.2015. Ebenfalls zu sehen sind die drei gesetzten Meilensteine. Die Prototyp-Abgabe erfolgt am 28.09.2015. Die Abgabe der Beta-Version erfolgt wiederum zwei Wochen später, am 12.09.2015. Als letzter Meilenstein gilt die letzte Abgabe, das Release am 09.11.2015.

* + 1. Verantwortlichkeiten

Die Verantwortlichkeiten wurden gleich zu Beginn des Projekts gemeinsam innerhalb des Teams verteilt. Je nach Stärken der jeweiligen Teammitglieder konnten die Verantwortungsbereiche bestmöglich verteilt werden. Auch wenn jedes Mitglied selbst dafür zuständig ist seine Anforderungen zeitgerecht und funktionsfähig abzuliefern, werden auftretende Probleme gemeinsam in der Gruppe angegangen und gelöst.

Im nachfolgenden werden die Aufgabenbereiche der einzelnen Projektmitglieder tabellarisch dargestellt.

****

* 1. Aufbau, Entwurfsentscheidungen
     1. Files & folders

In Eclipse wurde der Aufbau der Struktur in 6 Pakete untergliedert. db, file, gui, ki, pusher und viergewinnt.

Db:

* + - die hsqldb hinterlegt, bestehend aus der accessDB.java klasse , die die datenbankabfragen zur verfügung stellt
    - Bestehend ebenfalls aus myDB.properties was meta informationen enthält zur DB
    - myDB.script die die db inkrementell speichert

file

* + - Changelistener.java, setzt einen Thread um der die kommunikation auf dateibasis umsetzt
    - FileImpl.java, die implementierung des interfaces der dateischnittstelle
    - FileController.java, interface für dateischnittstelle

GUI:

KI:

* + - Intelligenz.java beinhaltet programmlogik und spielstrategien

Pusher:

* + - PusherController.java, interface für pusherschnittstelle
    - PusherImpl.java, die implementierung des interfaces der pusherschnittstelle

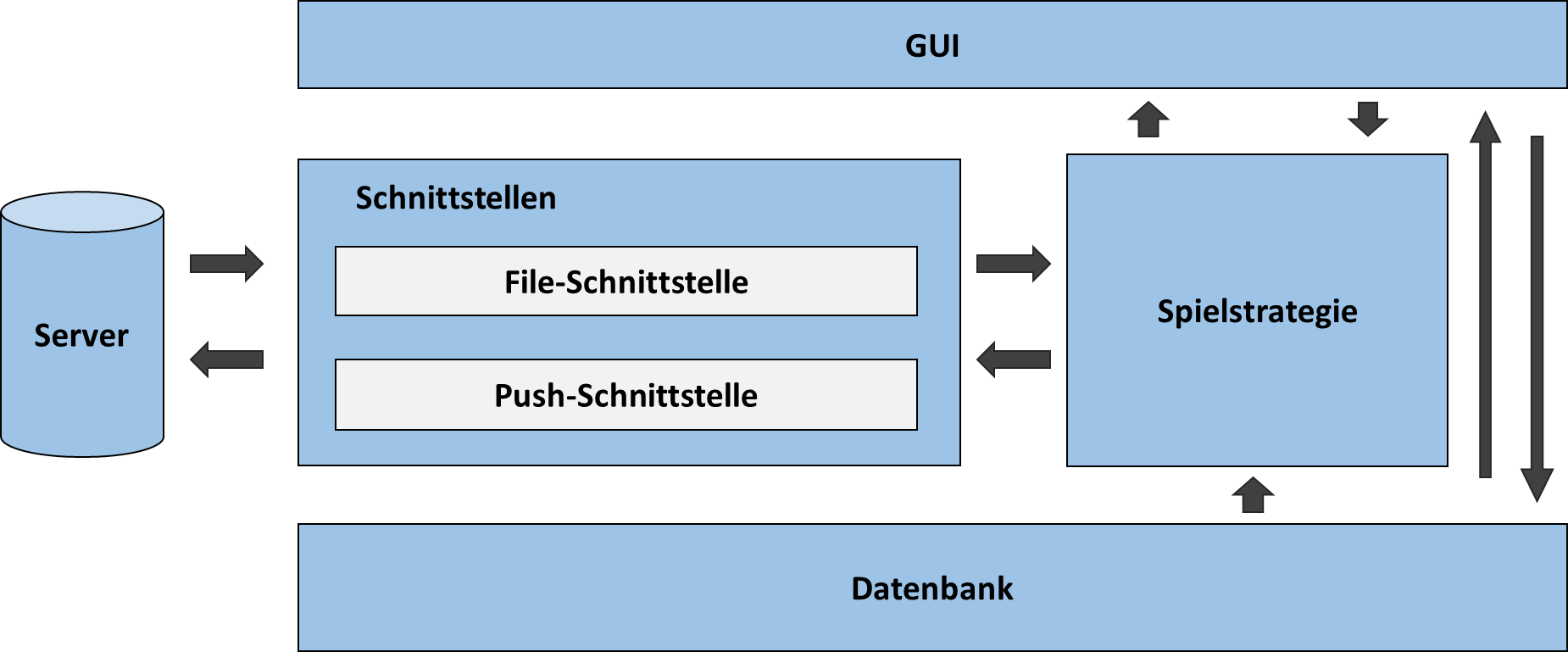
Viergewinnt:

* + - CommunicationController.java, allgemeines abstraktes komunnikations interface, das von file und pusher schnittstelle erweitert wird
    - ViergewinntMain.java, erfolgt der programmaufruf
    1. Schichtenarchitektur

Für eine bessere Strukturierung des Projekts und der einzelnen Programmbestandteile hilft ein Schichtenmodell bzw. eine Schichtenarchitektur. Im Projekt „Fungi“ handelt es sich um eine drei Schichten Architektur, diese ist nachfolgend abgebildet.

Diese besteht aus der Datenhaltungsschicht, Fachkonzeptschicht und der GUI-Schicht.

Die unterste Schicht, die Datenhaltungsschicht ist dafür zuständig, dass die erfassten Daten konsistent aufbewahrt werden. Die mittlere Schicht, die Fachkonzeptschicht (KI und Schnittstellenkommunikation) steuert alle Spielabläufe und die GUI sorgt als oberste Schicht für die graphische Darstellung des Programms.

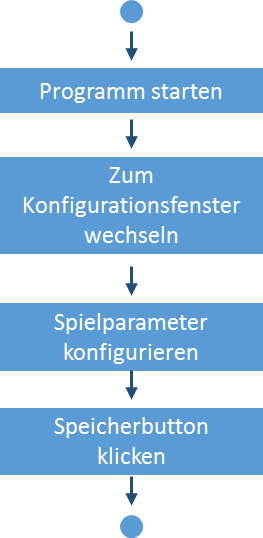


* + 1. Aktivitätsdiagramm

Aktivitätsdiagramme zeigen die Aktivitäten eines Programmes und dienen zur Modellierung des Kontrollflusses und des Datenflusses. Somit können sowohl komplexe Abläufe vereinfacht dargestellt werden als auch die zeitliche Reihenfolge der Ausführung verdeutlicht werden.

In folgender Abbildung sieht man das Aktivitätsdiagramm für die Konfiguration der Spielparameter.

Nach dem Programmstart wird ein Wechsel zum Konfigurationsfenster benötigt. Hier werden die benötigten Parameter eingelesen und durch den vorhandenen Button abgespeichert.

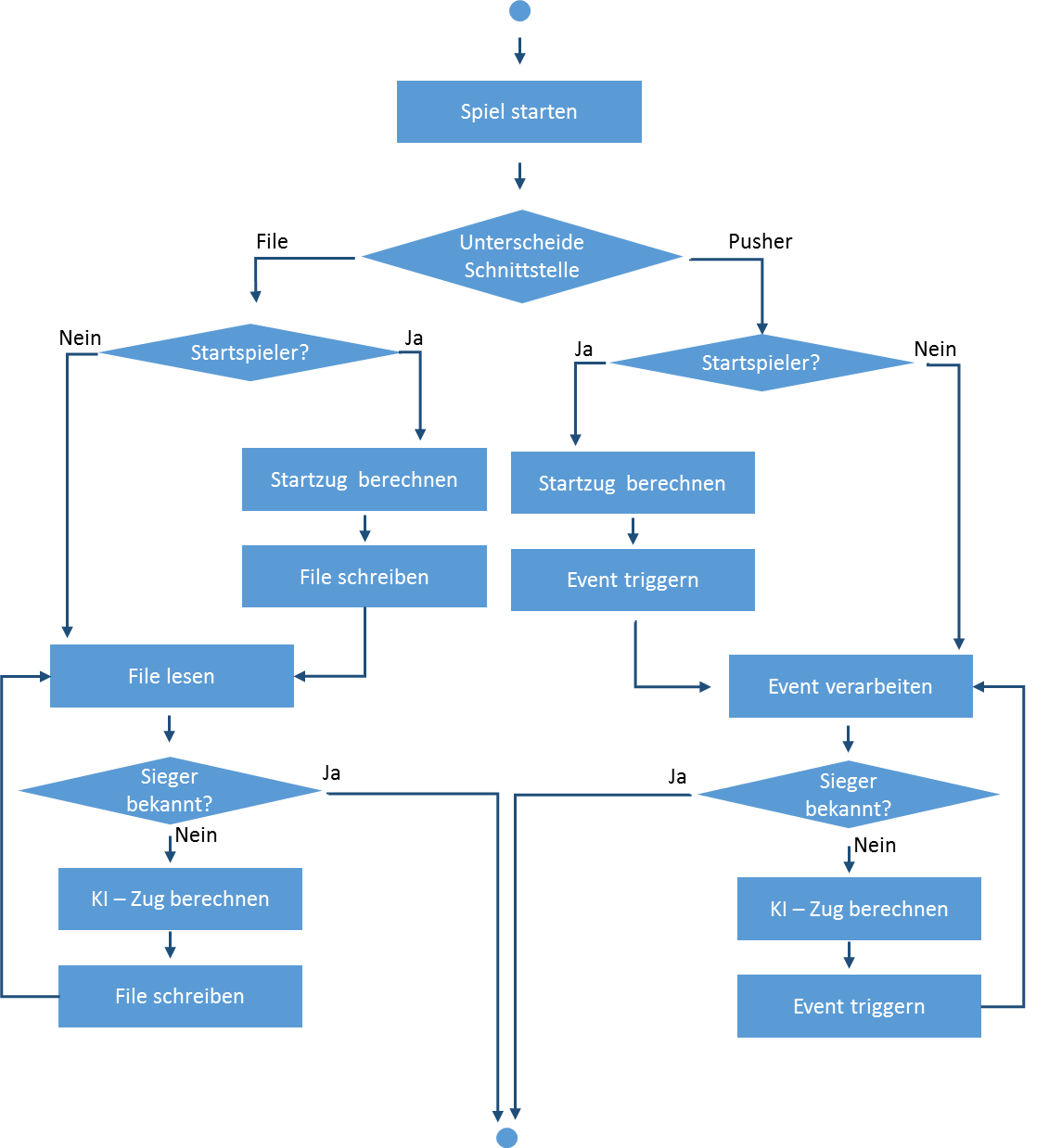


Konfiguration der Spielparameter

Das zweite Aktivitätsdiagramm zeigt den eigentlichen Programmablauf von Fungi 4-Gewinnt.

Sobald der Agent und der Server fehlerfrei laufen, kann das Spiel gestartet werden. Zu Beginn wird vom Agent überprüft, ob die Schnittstelle zum Server mittels eines Files oder eines Pushers realisiert wird. Anschließend wird ausfindig gemacht, ob der Agent beginnt oder der Gegner bereits angefangen hat.

Danach folgt in einer Schleife abwechselnd das Auslesen des Files bzw. das Verarbeiten des Events, zudem das Berechnen des nächsten Zugs durch die KI und das anschließende File schreiben bzw. triggern des eigenen Wurfs. Dabei wird immer die Prüfung vorangestellt, ob bereits ein Spieler gewonnen hat. Ist dies der Fall, dann wird der Satz beendet.



Programmablauf

* 1. Konzept
     1. Umsetzung der Anforderungsliste?

Nachfolgenden werden sowohl die bereits festgelegten Anforderungen sowie deren Umsetzung innerhalb des Projektes mittels Tabelle dargestellt.

|  |  |
| --- | --- |
| Anforderung | Umsetzung |
| Graphische Oberfläche | Swing |
| Anzeige des Spielfeldes |  |
| Anzeige aller Spielstände |  |
| Anzeige aller Züge |  |
|  |  |
| Kommunikationsschnittstellen |  |
| Zug berechnen |  |
| Sieger erkennen |  |
| Starten, zurücksetzen und beenden der Anwendung |  |
|  |  |
| Technologiebasis Datenbank | hasqldb |
| Speichern Statistik relevanter Daten |  |
| Drei Datenbankabfragen über Statistiken |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

* + 1. Mockups

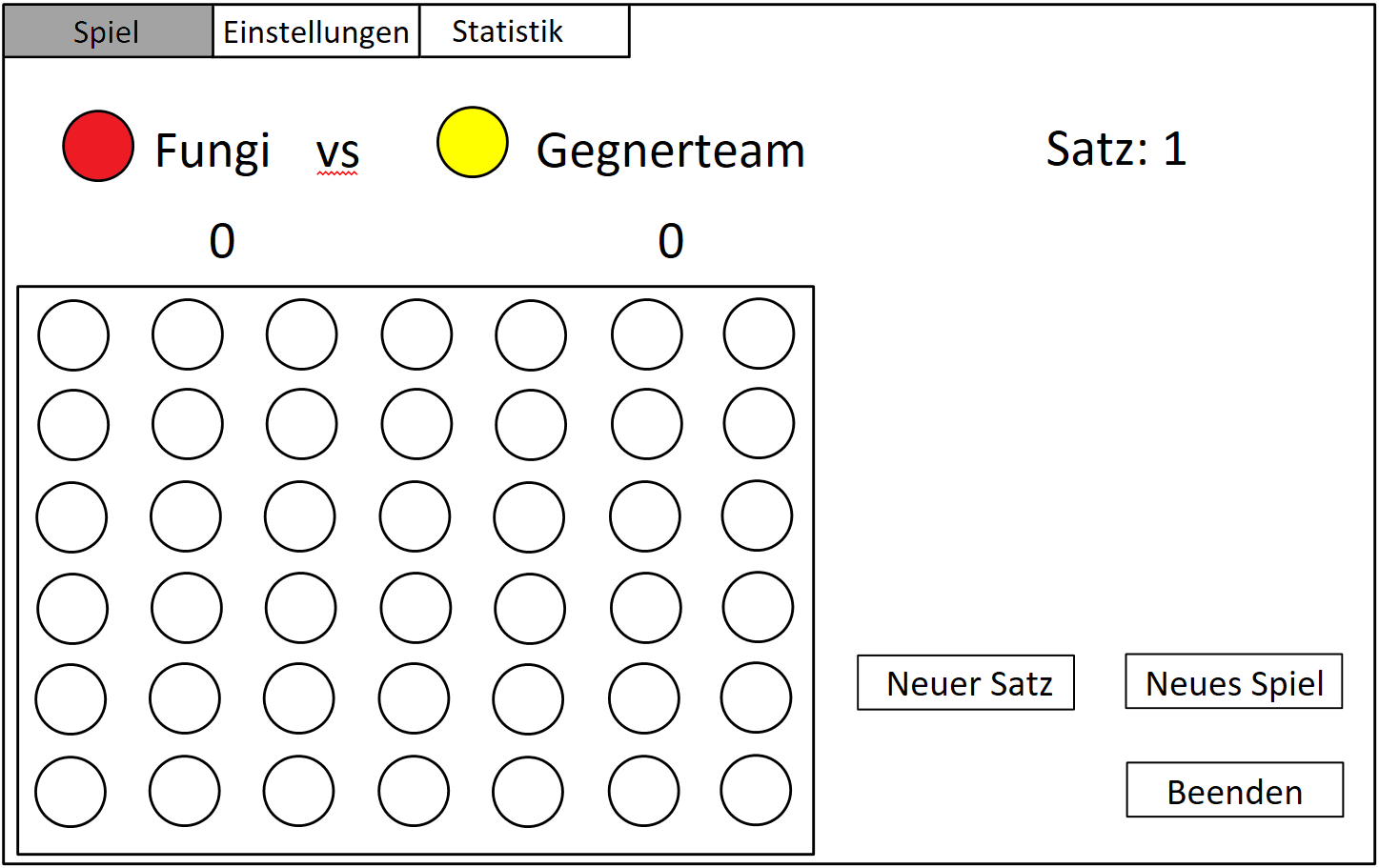


Abbildung 1: Spieloberfläche

Die GUI besteht aus den 3 Registerkarten "Spiel", "Einstellungen" und "Statistik". Im oberen Bereich werden die Teams angezeigt, die das aktuelle Duell bestreiten, sowie der Spielstand und der aktuelle Satz. Der untere Bereich beinhaltet das Spielfeld auf dem die Spielzüge visuell angezeigt werden. Des Weiteren gibt es die Buttons "Neuer Satz", "Neues Spiel" und "Beenden"

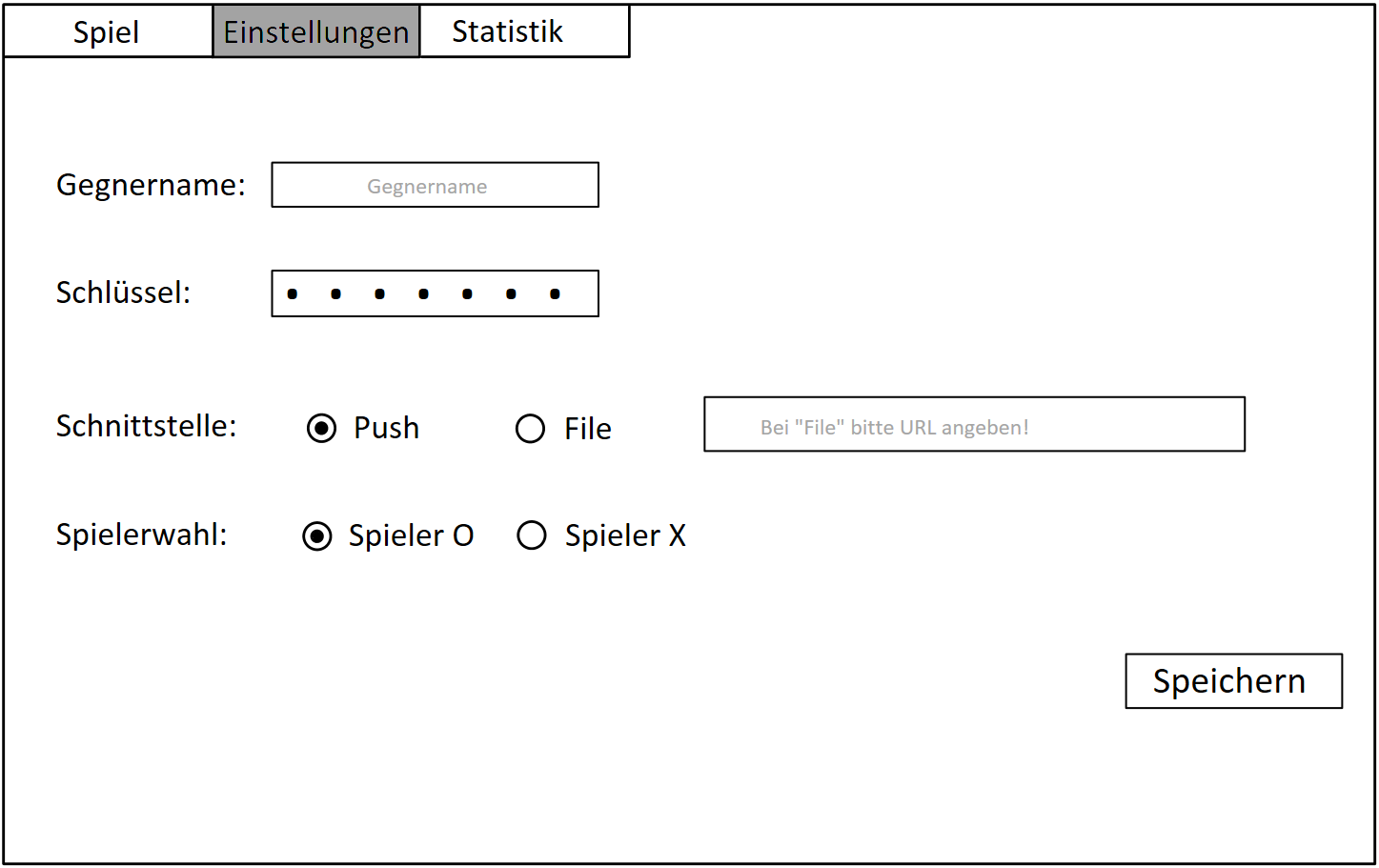
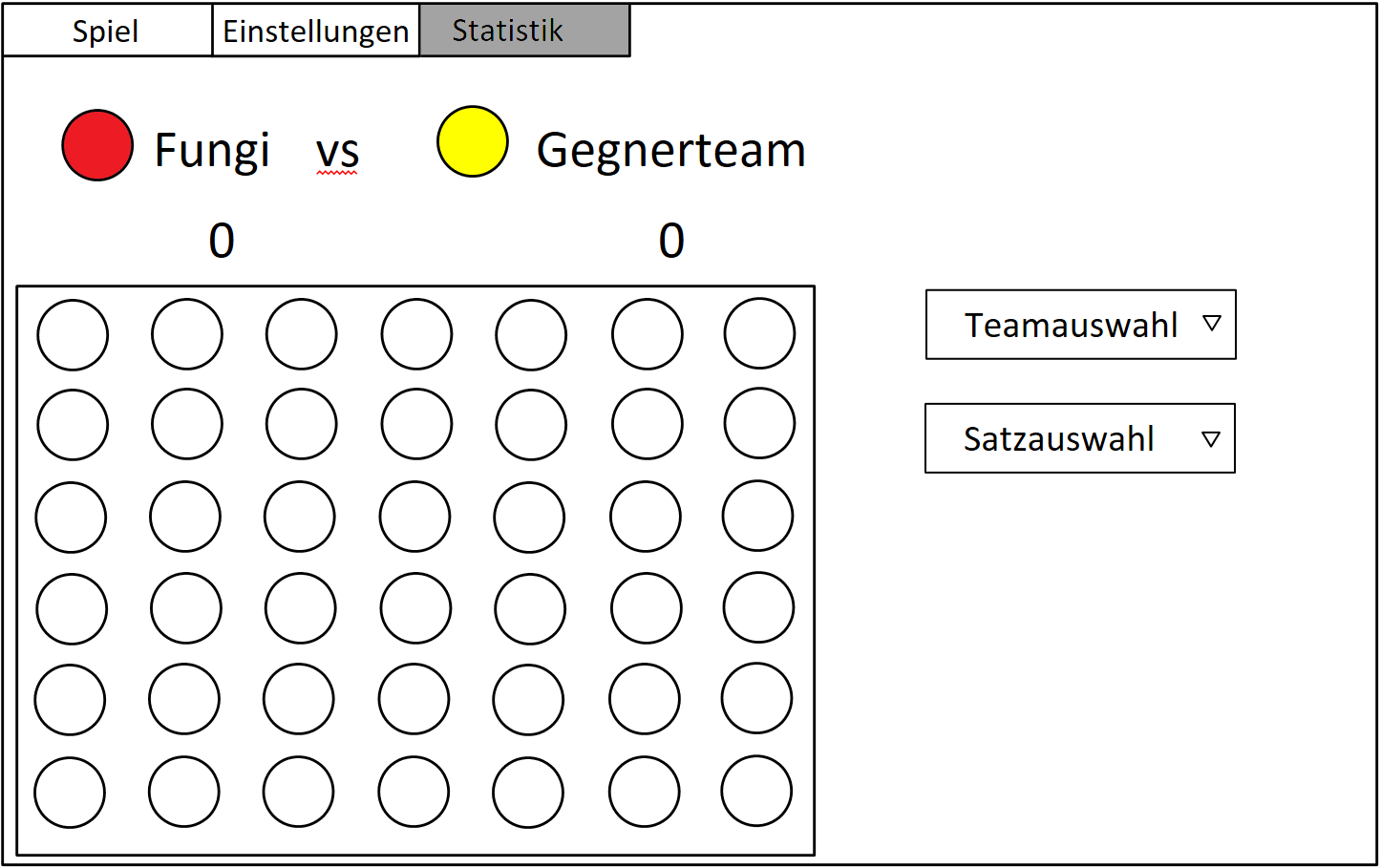
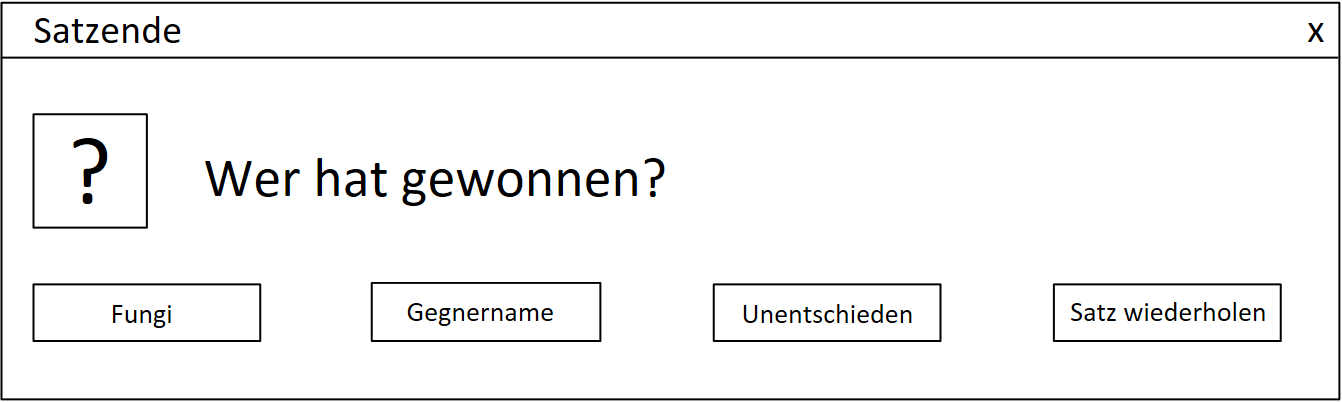


Abbildung 2: Konfigurationsoberfläche

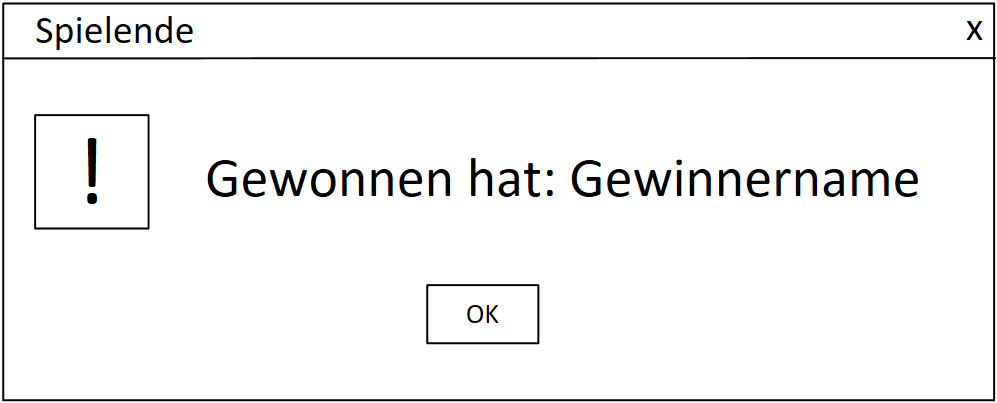
Die "Einstellungs-Seite" dient zur Konfiguration vor dem Spielstart. Einzustellen sind Gegnername, Schlüssel, Schnittstelle und die Spielerwahl. Im Falle der Schnittstelle "File", muss noch der dazugehörige Pfad angegeben werden. Der "Speichern-Button" dient zur Sicherung der Daten in einer lokal gespeicherten ".ini-Datei"



Die "Statistik-Seite" bietet die Möglichkeit den letzten Spielzug vergangener Sätze erneut grafisch darzustellen. Selektiert wird hierfür nach Gegner und Satz. Dies ist möglich, durch zwei Dropdown-Menüs auf der rechten Seite der GUI. Wie bereits auf der "Spiel-Seite" wird auch hier wieder das Spielfeld, sowie die Teamnamen und der Spielstand angezeigt.



Nach jedem Satz erscheint das "Satzende-Popup", dass nach dem Gewinner des Satzes fragt. Auswahlmöglichkeiten sind neben den spielenden Teams sind "Unentschieden" und "Satz wiederholen". "Satz wiederholen" wird benötigt, wenn beispielsweise ein Spieler die Verbindung zum Server verliert und der Satz erneut gespielt wird.



Nach jedem Spiel erscheint das "Spielende-Popup", dass anzeigt, wer das Spiel gewonnen hat. Dies dient lediglich zur Information des Spielers.

Bevor das GUI-Team von Fungi-Software-Solutions mit der Programmierung der GUI beginnen konnte, musste zuerst die Entscheidung getroffen werden, auf welcher Basis die GUI „laufen“ soll. Die durch Anforderungen begrenzten Möglichkeiten waren JavaFx oder Swing. Aufgrund bereits gesammelter Erfahrungen mit Swing fiel die Entscheidung gegen JavaFx. Um die spätere Nutzung auf verschiedenen Rechnern zu vereinfachen, einigte man sich zusätzlich darauf, keine Plugins zu verwenden und die aktuellste Version von Eclipse für die anstehende Programmierung zu verwenden. Die Dokumentation der GUI-Entwicklung erfolgt anhand der einzelnen Arbeitsschritte, die in einzelnen Versionsnummer dargestellt werden.

V0.1: Der erste Schritt ist der erste Entwurf der GUI. Hierfür wurde ein grobes Gerüst gebaut und die Verwendung einzelner Komponenten wie JOptionPane vorbereitet. Der Entwurf bestand aus dem Programmfenster, der Registerkartenauswahl (Spiel, Optionen und Statistik), sowie sämtlicher benötigter Buttons.

V0.12: In der zweiten Version wurden sämtliche Popup-Infofenster programmiert und mittels Buttons zu Testzwecken aufrufbar gemacht. Da das „Satzende-Popup“ den Spieler auffordert den Gewinner zu bestätigen, musste hier eine Auswahloption implementiert werden. Hierfür wurde JOptionPane.showOptionDialog verwendet, die mehrere Auswahlmöglichkeiten anbietet und je nach Antwort einen vorher definierten Wert zurückgibt. Das „Spielende-Popup“ hingegen dient lediglich als Information, wer das gesamte Spiel gewonnen hat und muss durch einen Klick auf den „OK“ Button bestätigt werden. Hierfür reichte eine Nutzung von JOptionPAne.showMessageDialog.

V0.15: Nun wurde das Spielfeld in der Registerkarte „Spiel“ angelegt. Das grafische Spielfeld besteht aus einer png-Datei im Vordergrund, dass durch eine Tabelle im Hintergrund vervollständigt wird. Durch ein Array werden die Tabellenfelder farbig gefüllt und somit der Einwurf der Spielsteine simuliert.

V0.2: In Version 0.2 wurde ein großer Schritt abgeschlossen, in dem die Steuerungslogik hinterlegt wurde. Hierbei wurden nicht nur die Buttons mit Funktionen hinterlegt, sondern auch alle Schnittstellen nach außen funktional umgesetzt. Des Weiteren werden nun alle getätigten Einstellungen mittels FileWriter in einer Konfigurationsdatei gespeichert, die bei erneutem Programmstart wieder geladen. Dies ermöglicht ein noch besseres Spielererlebnis, da die Einstellungen nun nicht mehr vor jedem Spiel neu auf die Bedürfnisse des Kunden manuell von diesem eingestellt werden müssen. Somit ist höchster Komfort für den glücklichen Spieler geboten.

V0.3: Im letzten großen Schritt wurde die Spielfläche und die Statistik noch optimal angepasst. Zudem wurde der Quellcode noch optimiert um eine maximale Performance zu ermöglichen und den hohen Ansprüchen von Fungi-Software-Solutions gerecht zu werden.

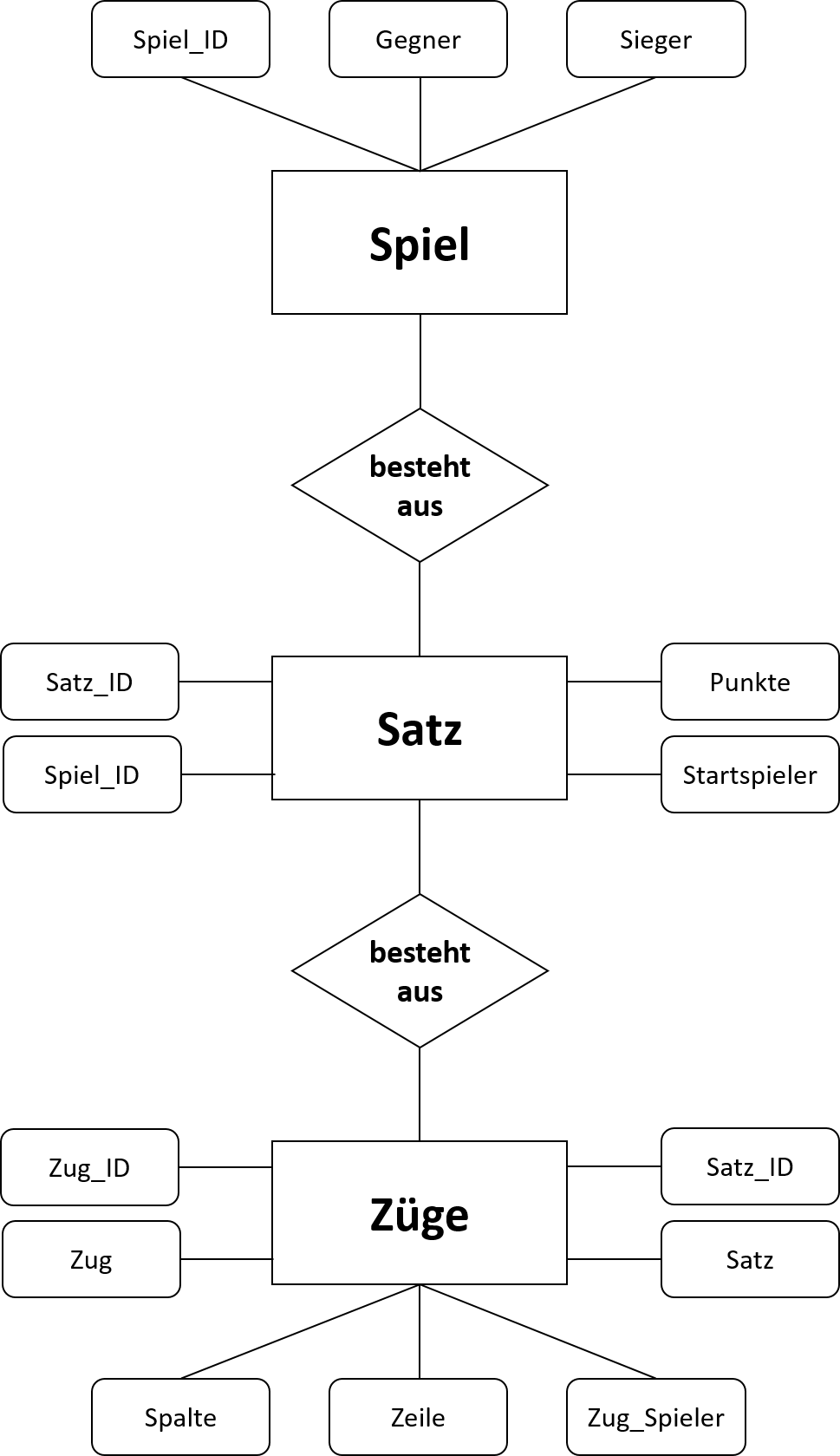
Nach Abschluss der Arbeiten wurde die Implementierung an den hochqualifizierten Modellierer weitergegeben, der für eine ideale Implementierung der Anwendungslogik zuständig ist.

* 1. Struktur   
     1. Entitäten (Komponenten, Module, Pakete, Klassen)

Luis

* + 1. Deren Beziehungen und Abhängigkeiten

* + 1. Datenbank (ER-Diagramm)



Bevor eine relationale Datenbank entwickelt werden kann, muss ein Modell entworfen werden das für einen besseren Überblick sorgt. Das Entity-Relationship-Diagramm (ER-Diagramm) wird hier als Methode zur Datenmodellierung benutzt, es besteht im Allgemeinen aus Rechtecken (Objektklassen) und Rauten (Beziehungen zwischen den Klassen). Um die Rechtecke sammeln sich die Attribute der Objektklassen, diese stellen die spezifischen Eigenschaften der einzelnen Objekte dar. Die Abbildung zeigt das ER-Diagramm der fungi-Datenbank, bestehend aus den drei Objektklassen Spiel, Satz und Züge. Diese besitzen jeweils ihre eigenen Attribute und werden durch die Beziehung „besteht aus“ miteinander verknüpft.

* + 1. Klassen beschreiben