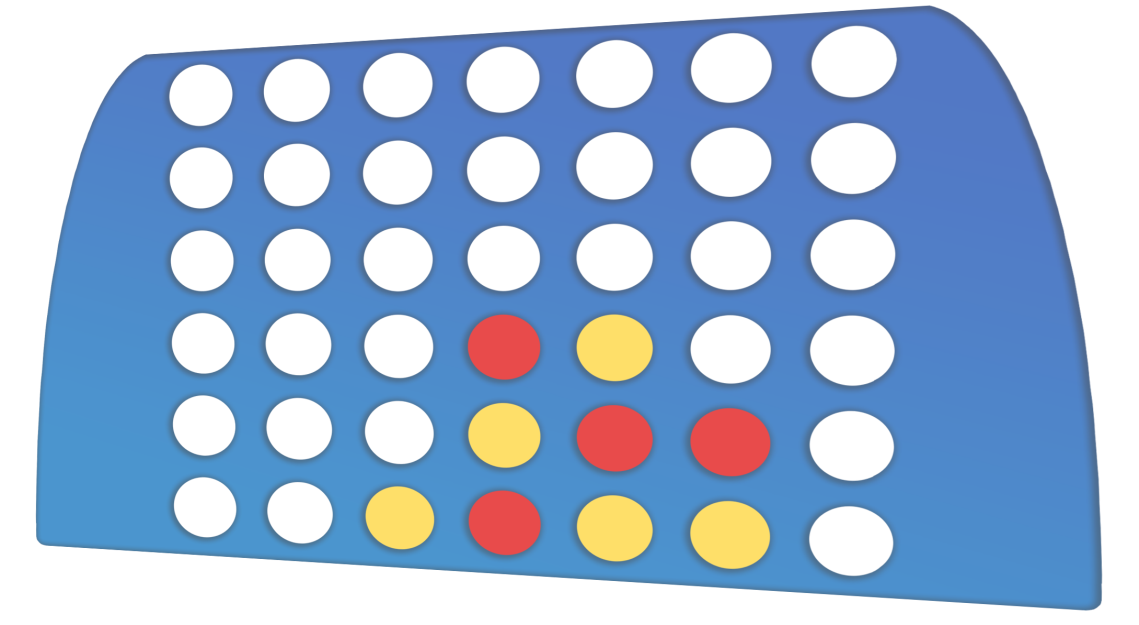
# Entwicklungsdokumentation

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

[Entwicklungsdokumentation 0](#_Toc432432034)

[1 Spezifikation 2](#_Toc432432035)

[1.1 Was leistet sie? 2](#_Toc432432036)

[1.1.1 UML um use case 2](#_Toc432432037)

[1.2 Sicht des Anwenders und Auftraggebers 2](#_Toc432432038)

[1.2.1 Anforderungsbeschreibung und Hilfsmittel 2](#_Toc432432039)

[1.2.2 Funktionale Anforderungen 3](#_Toc432432040)

[1.2.3 Lieferumfang 4](#_Toc432432041)

[1.2.4 Nichtfunktionale Anforderungen 5](#_Toc432432042)

[2 Design 6](#_Toc432432043)

[2.1 Prozess der Softwareentwicklung 6](#_Toc432432044)

[2.1.1 Vorgehensmodell (agiles/scrum) 6](#_Toc432432045)

[2.1.2 Meilensteine 7](#_Toc432432046)

[2.1.3 Verantwortlichkeiten 7](#_Toc432432047)

[2.2 Aufbau, Entwurfsentscheidungen 8](#_Toc432432048)

[2.2.1 Files & folders 8](#_Toc432432049)

[2.2.2 Schichtenarchitektur 9](#_Toc432432050)

[2.2.3 Aktivitätendiagramm 10](#_Toc432432051)

[2.3 Konzept 10](#_Toc432432052)

[2.3.1 Umsetzung der Anforderungsliste? 10](#_Toc432432053)

[2.3.2 Mockups 11](#_Toc432432054)

[2.4 Struktur mit LUIS 13](#_Toc432432055)

[2.4.1 Entitäten (Komponenten, Module, Pakete, Klassen) 13](#_Toc432432056)

[2.4.2 Deren Beziehungen und Abhängigkeiten 13](#_Toc432432057)

[2.4.3 Datenbank (ER-Diagramm) 13](#_Toc432432058)

[2.4.4 Klassen beschreiben 13](#_Toc432432059)

Abschnitt: Entwicklung

# 1 Spezifikation

## Was leistet sie?

## UML um use case

## Sicht des Anwenders und Auftraggebers

## 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 sie einen Software-Agenten, der autonom das Spiel „4 gewinnt“ gegen einen anderen Spieler spielen kann.
* Ihr Agent kommuniziert mit einem Server-Programm, mit dem auch der Gegenspieler, d. h. der Agent des anderen Teams, verbunden ist.
* Den Bau des Agenten führen sie - im Rahmen der vorgegebenen Anforderungen - selbstständig im Team durch. Das bedeutet insbesondere:
  + Sie managen ihr Projekt eigenverantwortlich
  + Sie gestalten eine Lösung entsprechend ihren Vorstellungen.
  + Sie wenden eine Methodik ihrer Wahl an.
  + Implementierungsdetails liegen weitgehend in ihrem Ermessen.
* Der Rahmen der vorgegebenen Anforderungen wird maßgeblich durch die wichtigsten zu liefernden Ergebnisse bestimmt (genauer siehe nächste Seiten):  
  + 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 (Abgabe am Ende der letzten Lehrveranstaltung, d. h. unmittelbar nach dem Turnier).
* 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 (kurz: Datei-Sst und Push-Sst).
  + Während eines Satzes erfolgt kein Wechsel der Sst. Zwischen den Sätzen, d. h. also auch während eines Spiels, kann die Sst. gewechselt werden. Der Wechsel der Sst. wird angesagt.
  + 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):

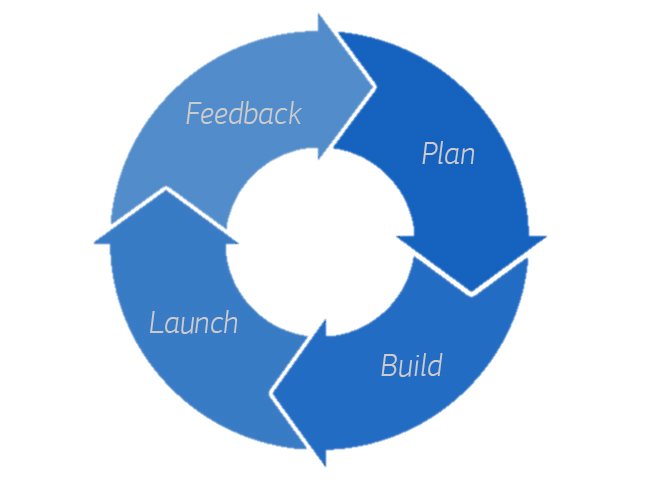
<https://github.com/Steffen911/WIProjekt>

* 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

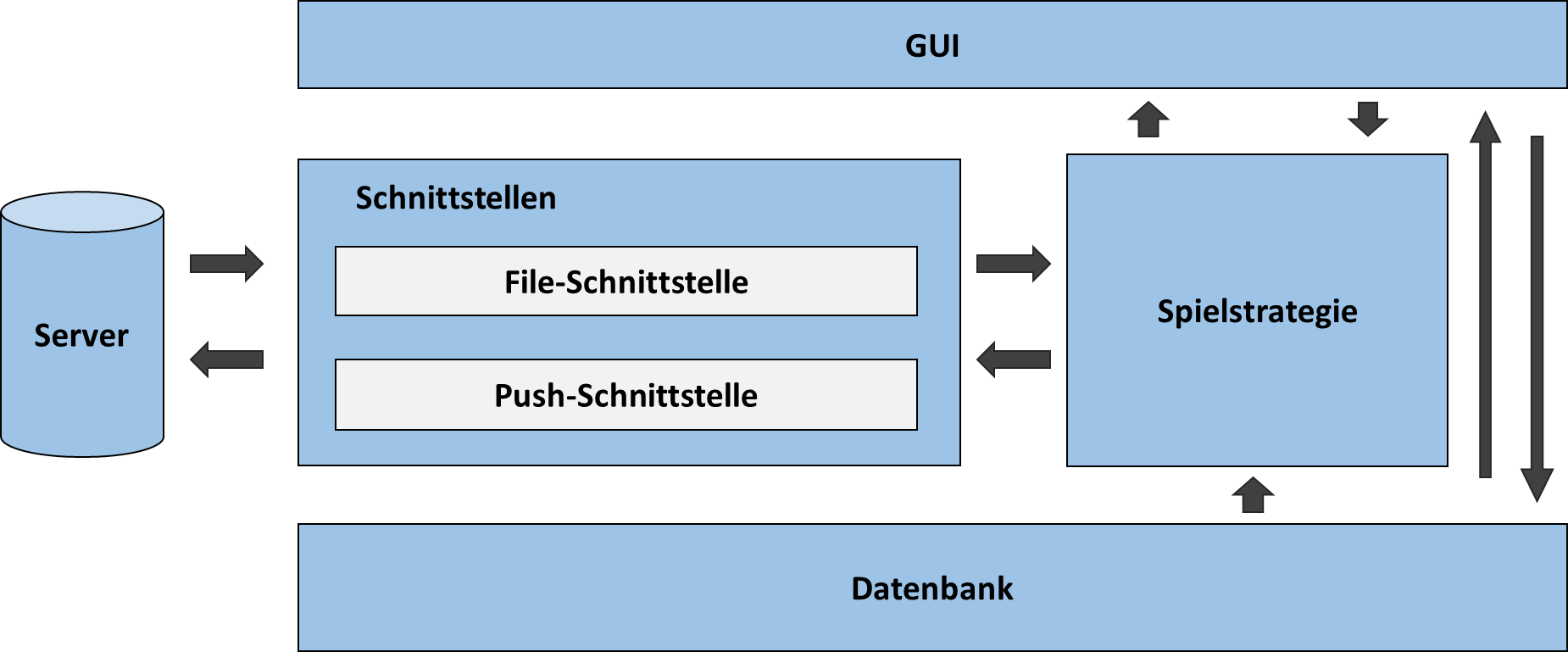
In Eclipse wurde der Aufbau der Struktur in 6 Pakete untergliedert. db, file, gui, ki, pusher und viergewinnt.

* + 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



* + 1. Aktivitätendiagramm
  1. Konzept
     1. Umsetzung der Anforderungsliste?

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

* + 1. Mockups

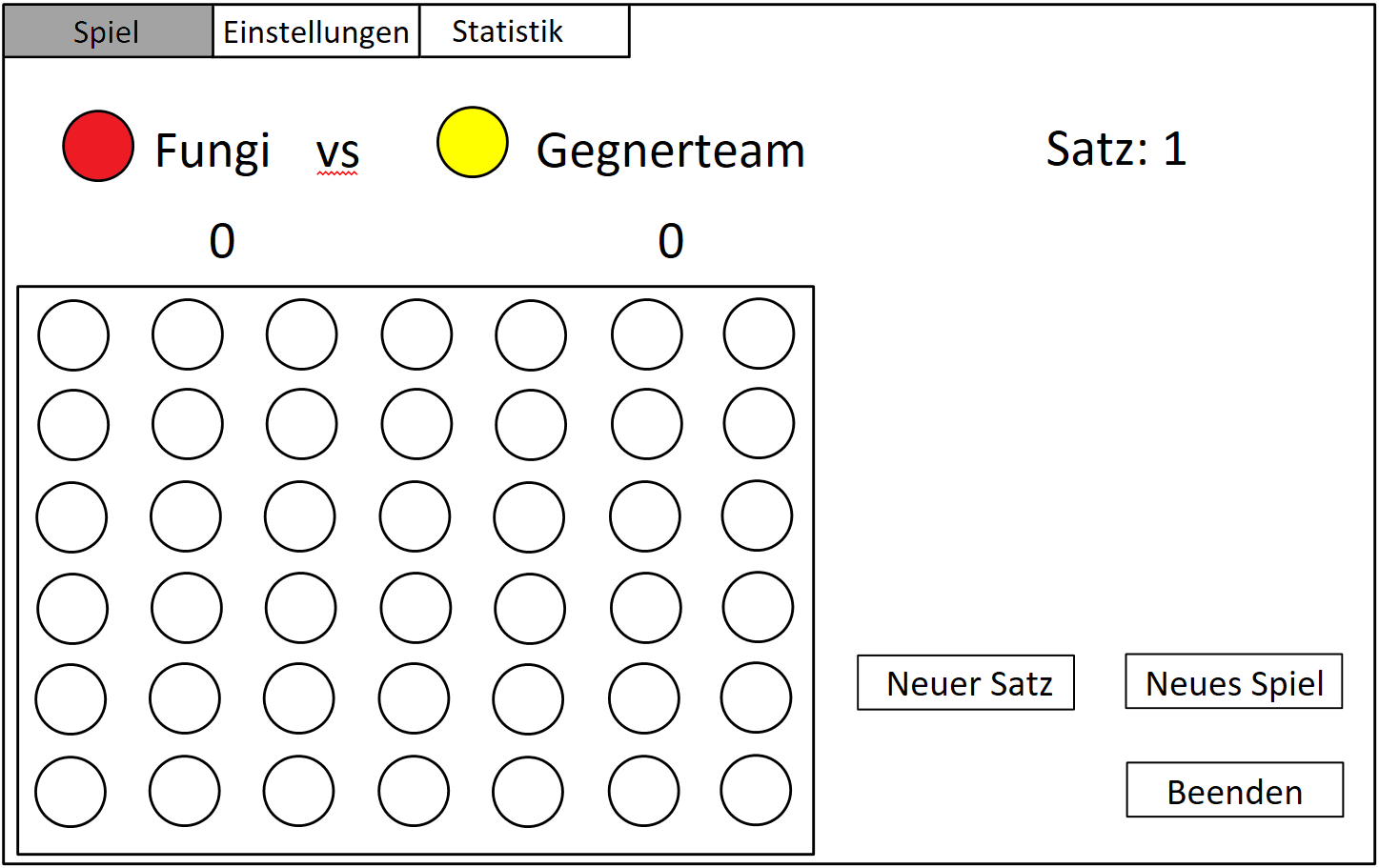


Abbildung 1: Spieloberfläche

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"

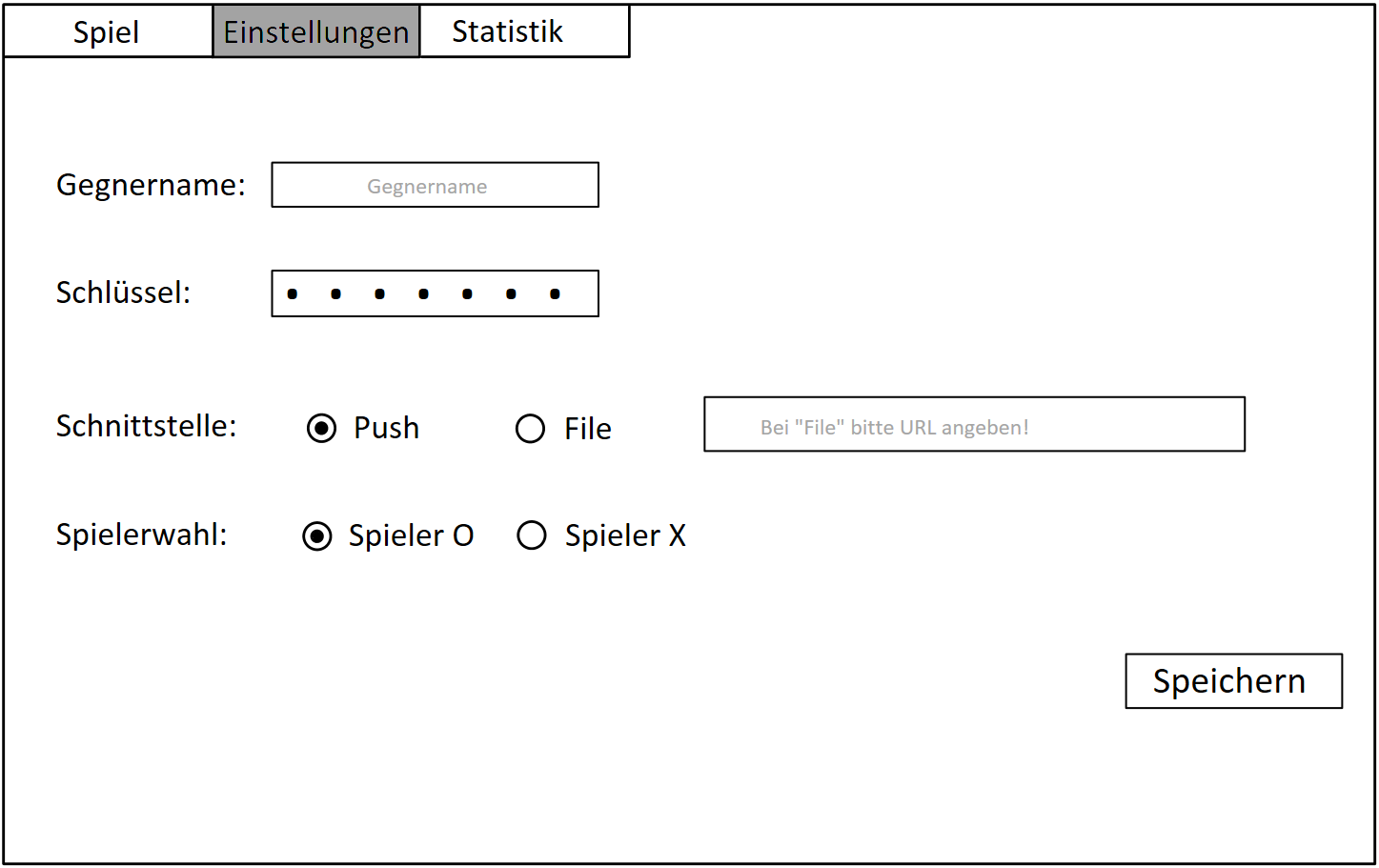
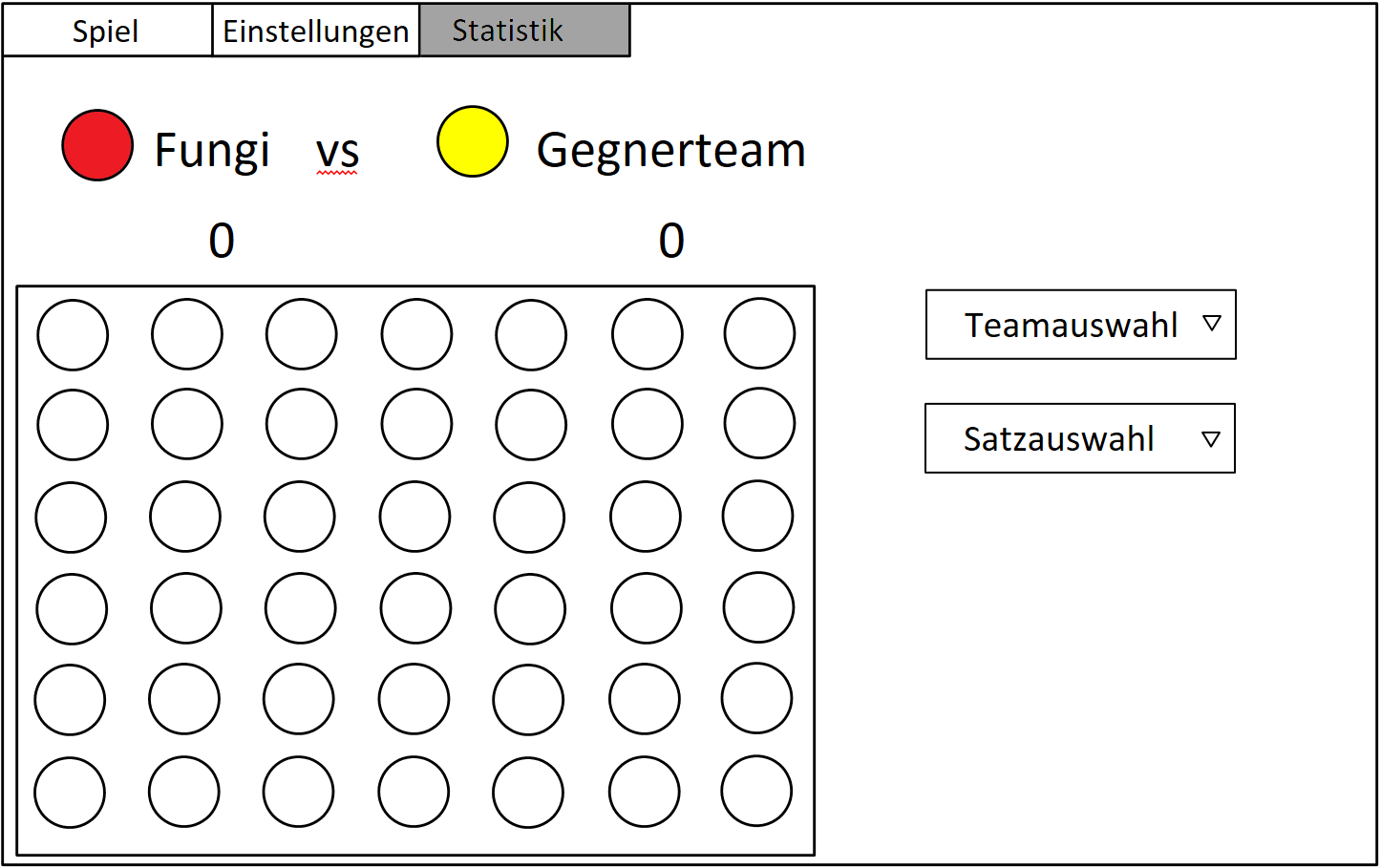
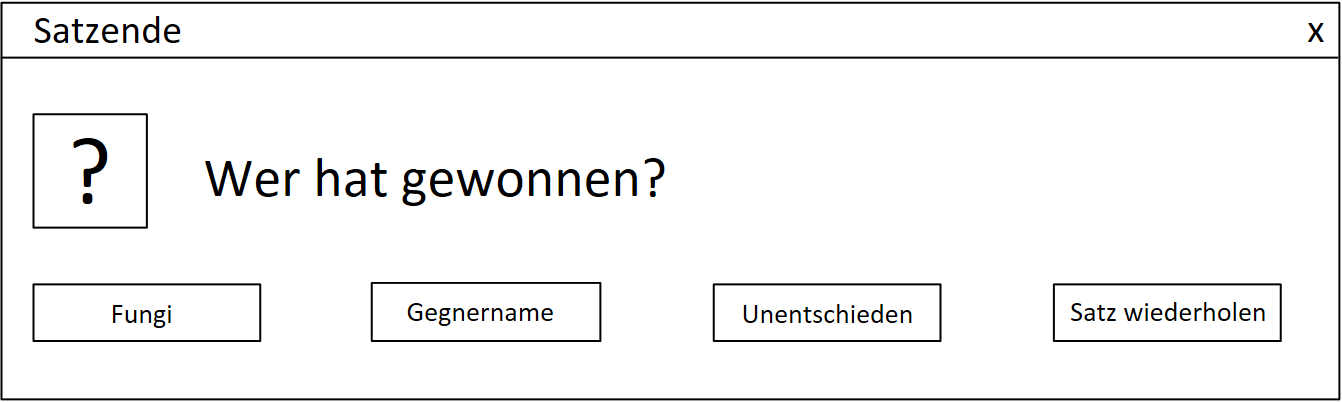


Abbildung 2: Konfigurationsoberfläche

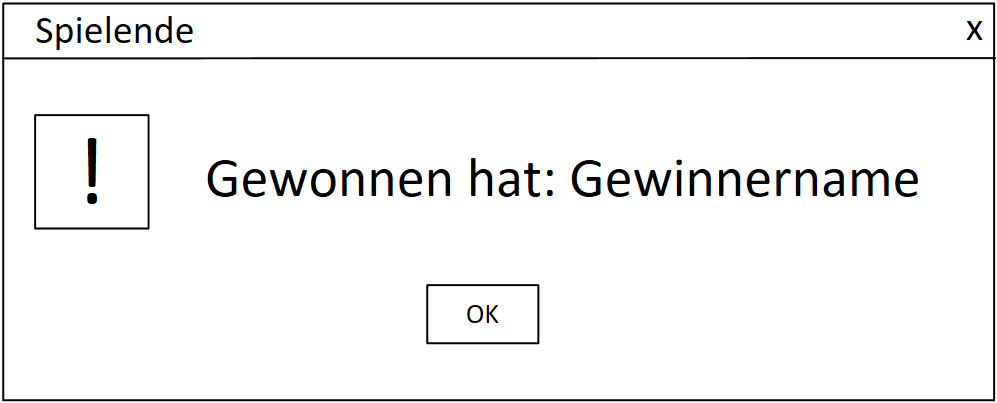
Die "Einstellungs-Seite" dient zur Konfiguration vor dem Spielstart. Einzustellen sind Gegnername, Schlüssel, Schnittstelle und die Spielerwahl. Im Falle der Schnittstelle "File", muss noch der dazugehörige Pfad angegeben werden. Der "Speichern-Button" dient zur Sicherung der Daten in einer lokal gespeicherten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

* Für gui auf swing zugegriffen
* Keinerlei plugins
* Alles aus akutellen eclipse version
* V0.1 version : GUI entwurf
* Vo.12 : infofenster spielende satzende hinzugefügt
* V0.15: spielfeld angelegt, array-tabelle agiert im Hintergrund eines png Spielfelds
* Tabelle die mittels array befüllt wird
* V0.2 : steuerungslogik hinterlegt save der Konfiguration via filewriter
* hier als Prototyp eingereicht
* V0.3 anpassung der spielfläche und der statistik, konfigurationseinstellungen bleiben beim programmende erhalten, optimierung des quellcodes (säubern, novh mehr beschreibungen ergänzen)
* Dann an luis übergeben zur verknüofung der Anwendungslogik
  1. Struktur
     1. Entitäten (Komponenten, Module, Pakete, Klassen)
     2. Deren Beziehungen und Abhängigkeiten
     3. Datenbank (ER-Diagramm)
     4. Klassen beschreiben