Projekt-Lösungsdokument   
Snake-Core, SnakeFX, SnakeServer

Ostfalia Fachhochschule für angewandte Wissenschaften

Benjamin Wulfert

Leonard Reidel

Semester: Wintersemester 2020

28. Oktober 2020

Inhaltsverzeichnis

[Projektdokumentation 3](#_Toc55501438)

[Architektur 3](#_Toc55501439)

[Core 3](#_Toc55501440)

[Berechnung der Spieler-Bewegungen & Positions-Berechnung 3](#_Toc55501441)

[SnakeFX 3](#_Toc55501442)

[Login-Screen 3](#_Toc55501443)

[Registrierungs-Prozess 4](#_Toc55501444)

[Login-Prozess 4](#_Toc55501445)

[Register-Screen 4](#_Toc55501446)

[Home-Screen 4](#_Toc55501447)

[Aktive Spieler 4](#_Toc55501448)

[Aktive Spiele 4](#_Toc55501449)

[Spielhistorie-Screen 4](#_Toc55501450)

[New-Game-Screen 4](#_Toc55501451)

[Game-Screen 4](#_Toc55501452)

[SnakeServer 4](#_Toc55501453)

[Persistenz-Layer 5](#_Toc55501454)

[Repository Pattern 5](#_Toc55501455)

[H2 Database Engine 5](#_Toc55501456)

[API-Layer 6](#_Toc55501457)

[JUnit-Tests 6](#_Toc55501458)

[Fehler & Lösungen 6](#_Toc55501459)

# Projektdokumentation

Dieses Dokument stellt die Dokumentation für das Projekt dar. Im Folgenden werden die verschiedenen Aspekte des Projekts beschrieben.

## Architektur

Core

SnakeFX  
Front-End

SnakeServer  
Back-End

Abbildung - Projekt-Architektur

## Core

Das Core-Modul enthält die Kern-Aspekte der Anwendung – dazu zählen beispielsweise die Modelle welche im Klassendiagramm enthalten sind, die Kern-Mechaniken der Anwendung – wie die Berechnung des Spiels, etc. – sowie die Endpunkte der Anwendung (URLs für die API).

### Berechnung der Spieler-Bewegungen & Positions-Berechnung

// TODO – Hier könnten schon die Dokumente welche wir erstellt haben miteinfließen   
// Hier muss jedenfalls der Ablauf für diesen Aspekt des Systems beschrieben werden.

## SnakeFX

Das Modul SnakeFX ist das Front-End der Anwendung. Im Front-End sind die User Interfaces (UI) definiert und implementiert. Des Weiteren enthält konsumiert das Front-End mittels Schnittstelle Daten aus dem Backend.

### Login-Screen

Der Login-Screen enthält vier Schaltflächen für die Interaktion des Benutzers. Zwei Texteingaben zur Angabe eines Benutzernamen und eines Passworts – und zwei Schaltflächen / Buttons um die Benutzerangaben (Name, Passwort) mittels Login oder Registrierung an das Backend zu übertragen. Beide Prozesse werden im folgenden näher beschrieben.

#### Registrierungs-Prozess

Das Front-End sendet einen POST-Request an die HTTP-Schnittstelle des Backends. Diese URL lautet <http://localhost:8080/api/login>. Als Header-Daten des Post-Requests werden der Benutzername sowie dessen Passwort (Hash) versendet. Diese Daten werden auf der Seite des Backends empfangen. Die empfangenen Daten werden mittels Unmarshalling1 vom Backend in eine Instanz der Benutzer-Klasse umgewandelt und in die Datenbank persistiert. Die Nutzerdaten des registrierten Benutzers können anschließend für die Anmeldung am System verwendet werden.

#### Login-Prozess

// TODO

### Register-Screen

// TODO - entfällt, da die Registrierung einfach über den Login-Screen getätigt wird.  
// TODO – muss noch im UML-Diagramm sowie im ScreenFlow aktualisiert werden.

### Home-Screen

Der Home-Screen stellt die zentrale Benutzeroberfläche der Anwendung dar. Der Benutzer erhält darauf Zugriff nach einem erfolgreichen Anmeldeversuch. Der Home-Screen bietet die Möglichkeit, alle aktiven Spieler und alle aktiven Spiele des Systems zu betrachten. Des Weiteren gelangt der Benutzer über den Home-Screen zur Spielhistorie-Oberfläche. Durch Betätigung der Schaltfläche „Neues Spiel“ ist der Benutzer in der Lage neue Spielrunden zu definieren und in der Lobby zu veröffentlichen. Aktive Spieler können dann, sofern noch genügend Kapazitäten vorhanden sind, dem Spiel beitreten.

#### Aktive Spieler

Zeigt alle aktiven Spieler (am System angemeldete Benutzer) tabellarisch an.

#### Aktive Spiele

Zeigt alle aktiven Spiele (Spielrunden des Systems) tabellarisch an. Ein Spieler kann an einem Spiel teilnehmen (sofern das Spiel noch nicht begonnen wurde oder ein Spieler-Slot verfügbar ist).

### Spielhistorie-Screen

// TODO

### New-Game-Screen

// TODO

### Game-Screen

// TODO SNAKE LOGIK HIER IN PRÄGNANTEN KURZEN SÄTZEN

## SnakeServer

Das Snake-Server Module stellt das Backend der Anwendung dar. Teil des Moduls ist eine Datenbank sowie deren Anbindung an das Modul. Des Weiteren stellt SnakeServer die verschiedenen Schnittstellen zur Verfügung welche zur Kommunikation mit dem Front-End benötigt werden.

Das Backend ist in verschiedene Ebenen / Layer unterteilt.

### Persistenz-Layer

Der Persistenz-Layer verwaltet die Speicherung, Aktualisierung und den Bezug von Daten aus dem relationalen Datenbank Management-System (RDBMS).

Als RDBMS verwenden wir H2 – welche sowohl als In-Memory als auch als File-Storage variante verwendet werden kann. Dieses vorgehen erweist sich insbesondere für das Aufsetzen der Anwendung als Hilfreich, da die Installation und Konfiguration des RDBMS entfällt. Des Weiteren vereinfacht das Vorgehen die Realisierung und Nutzung von Unit-Tests.



Abbildung - Übersicht der verwendeten Technologien

### Repository Pattern

// TODO  
// Spring Data liefert das mit aus, wir nutzen es um Entitäten zu persistieren und zu beziehen  
// Wo ist hier evtl der Unterschied zwischen dem EntityManager ?

### H2 Database Engine

Ein weiterer Vorteil der Nutzung von H2 ist das in der Abhängigkeit enthaltenen Weboberfläche zur Verwaltung der Datenbank, sodass auch die Installation und Konfiguration eines solchen Tools entfällt:



Abbildung – Login des User-Interfaces zur Verwaltung der In-Memory / File-Storage Datenbank.  
Die URL zur Datenbank lautet: jdbc:H2:mem:testdb.



Abbildung - User-Interface zur Verwaltung der Tabellen

### API-Layer

Der API-Layer definiert die vom Backend bereitgestellten Schnittstellen welche zur Kommunikation vom Frontend mit dem Backend benötigt werden.

## JUnit-Tests

Sowohl Backend als auch Frontend sollten mehrere Unit-Tests enthalten. Im Bereich des Backends wäre sinnvoll JUnit-Tests zu entwickeln welche die Schnittstelle testen. Im Bereich des Frontend wäre der Konsum der Schnittstelle denkbar.

## Fehler & Lösungen

* Intellij kompiliert Anwendung nicht, obwohl Abhängigkeiten vorhanden sind:  
  https://stackoverflow.com/questions/20137020/package-doesnt-exist-error-in-intellij