HTL - Perg

Höhere Abteilung für Informatik

Diplomarbeit

Cross-Platform Mobile Game Development

Projektteam: Simon Angerbauer

Roman Socovka

Projektbetreuer: Prof. Mag. Rupert Obermüller

In Zusammenarbeit mit dem Institute of Telecooperation der

Johannes Kepler Universität Linz

Betreuer: Dipl. -Ing. Matthias Steinbauer

Bearbeitungszeitraum: 06.07.2015 – 08.04.2016

Eidesstattliche Erklärung

Hiermit versichern wir, die vorliegende Arbeit selbständig, ohne fremde Hilfe und ohne Benutzung anderer als der von uns angegebenen Quellen angefertigt zu haben. Alle Stellen, die wörtlich oder sinngemäß aus fremden Quellen direkt oder indirekt übernommen wurden, sind als solche gekennzeichnet.

Perg, _____ Unterschrift _____

Simon Angerbauer

Perg, _____ Unterschrift _____

Roman Socovka

Kurzfassung

Aufgabenstellung:

Mobile Gelegenheitsspiele wie *Doodle Jump, Flappy Bird oder Candy Crush* erleben derzeit einen großen Aufschwung. Um auch auf den Zug aufzuspringen, zielt unsere Diplomarbeit daher auf die Entwicklung eines Cross-Platform-Spiels ab.

Der Begriff Cross-Platform bedeutet "plattformunabhängig" und heißt, dass das von uns produzierte Ergebnis auf mehreren Plattformen ausführbar ist und so möglichst viele potenzielle Benutzer erreicht.

Um das Spielerlebnis zu erweitern, soll das Spiel an soziale Netzwerke angebunden werden. Dies hat den Vorteil, dass der Spieler nicht unter einem Alias spielt, sondern direkt mithilfe seines eigenen Namens, welchen er bei der sozialen Plattform angegeben hat. So können die eigenen Freunde herausgefordert und gegen diese gespielt werden.

Zusätzlich sollen Spieldaten wie Spielerscore und gesammelte Münzenanzahl am Server gesammelt und ausgewertet werden.

Realisierung:

Das Cross-Platform-Spiel wurde mithilfe der Unity Game Engine realisiert.

Zuerst wurde der Spielzyklus konzeptioniert und die Designs der Spielelemente angefertigt.

Es wurde in Unity die Szenen aus den zuvor entworfenen Elementen aufgebaut und die dazugehörige Spiellogik in Skripten implementiert.

Der Mehrspieler-Modus wurde realisiert, indem eine Anbindung an die soziale Plattform Facebook implementiert wurde. Falls die Freunde das Spiel noch nicht nutzen sollten, kann der Spieler seinen Freunden Spieleeinladungen zu schicken. Die Spieleeinladungen werden dabei über das Graph API verschickt, welches von Facebook zur Verfügung gestellt wird.

Die gesamte Aufzeichnung der Spieldaten erfolgt über einen WildFly-Server, welcher die Daten mithilfe des kompakten Formates JSON vom Client empfängt und diese verarbeitet zurück an den Client versendet. Die dauerhafte Speicherung dieser Daten erfolgt auf einem MySQL-Server.

Ergebnis:

Das Ergebnis dieser Diplomarbeit ist ein Spiel, welches im Einzelspieler-Modus und im Mehrspieler-Modus über Facebook läuft. Die Spiele zwischen Benutzern sowie die erreichte Punkteanzahl werden über ein Service auf einer Datenbank gespeichert. Das Spiel ist lauffähig auf Android-und Windows-Geräten.

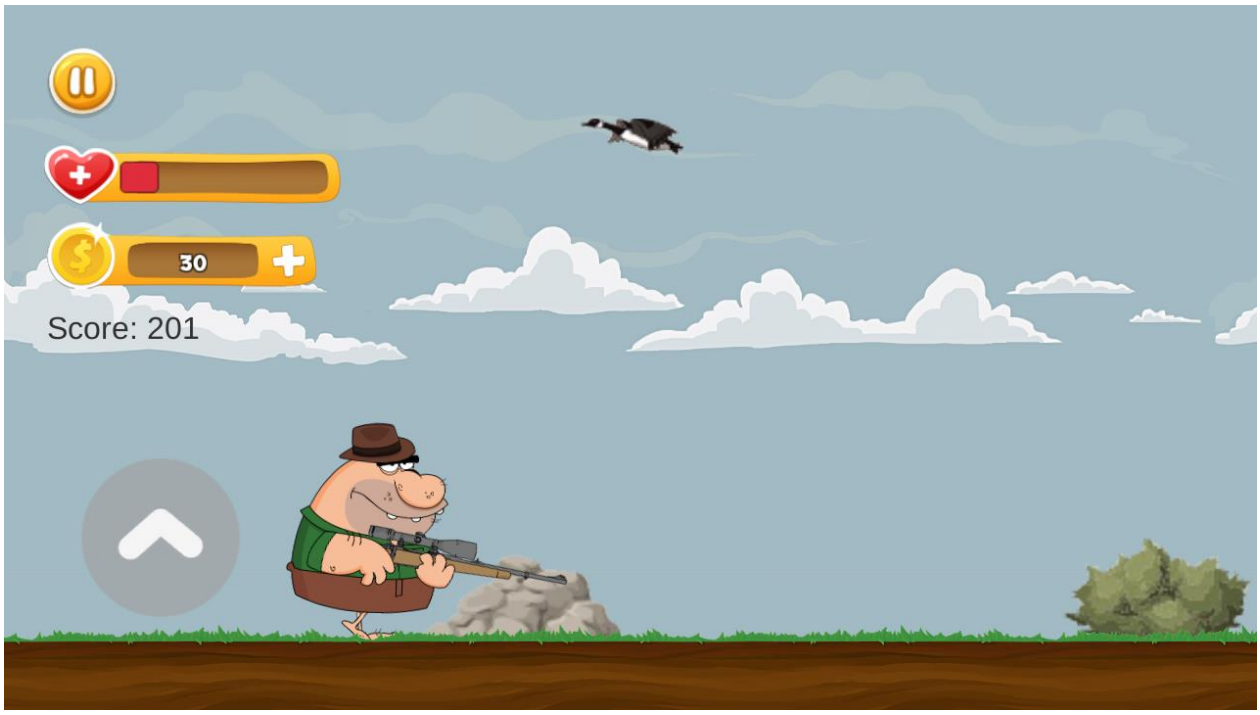


Abbildung 1: Ergebnis der Diplomarbeit

Abstract

Task description:

Mobile casual games like Doodle Jump, Flappy Bird or Candy Crush are booming. To join the trend, our thesis aims to develop a cross-platform game.
The term cross-platform means that the product developed by us is not specialized just for one platform. This has the advantage of supporting even more devices, and logically, to reach even more potential customers.

To expand the experience, the game should be linked to social networks. This means that instead of playing with an alias, the user takes his own name and uses it to play against his friends.

Implementation:

The cross-platform game was developed by using the Unity Game Engine.
At the beginning, we worked on the game cycle and designed the graphic elements. After that, we recreated the game cycle in Unity by using those graphic elements and implemented the game logic.

To make it possible for the players to challenge their friends, we developed a multiplayer mode, which is able to connect to the social media platform Facebook. The advantage is that the player is able to challenge any friend that he has in his friends list. If the challenged friend hasn't installed the app yet, it is possible for the challenger to send game invites to him in order to draw his attention. The game invitations are being sent by the Graph API, which is provided by Facebook.

The records are saved on a WildFly server. The client sends his data to the WildFly server by using the compact format JSON. After a positive validation, the server saves the data into a MySQL database.

Result:

The result of our thesis is a game that is playable in both a single player-and in a multi-player mode. The multi-player mode was realized by using Facebook.
Data like player score, last login time and the challenges are saved over a service on a database.
The game is executable on the platforms Windows and Android.

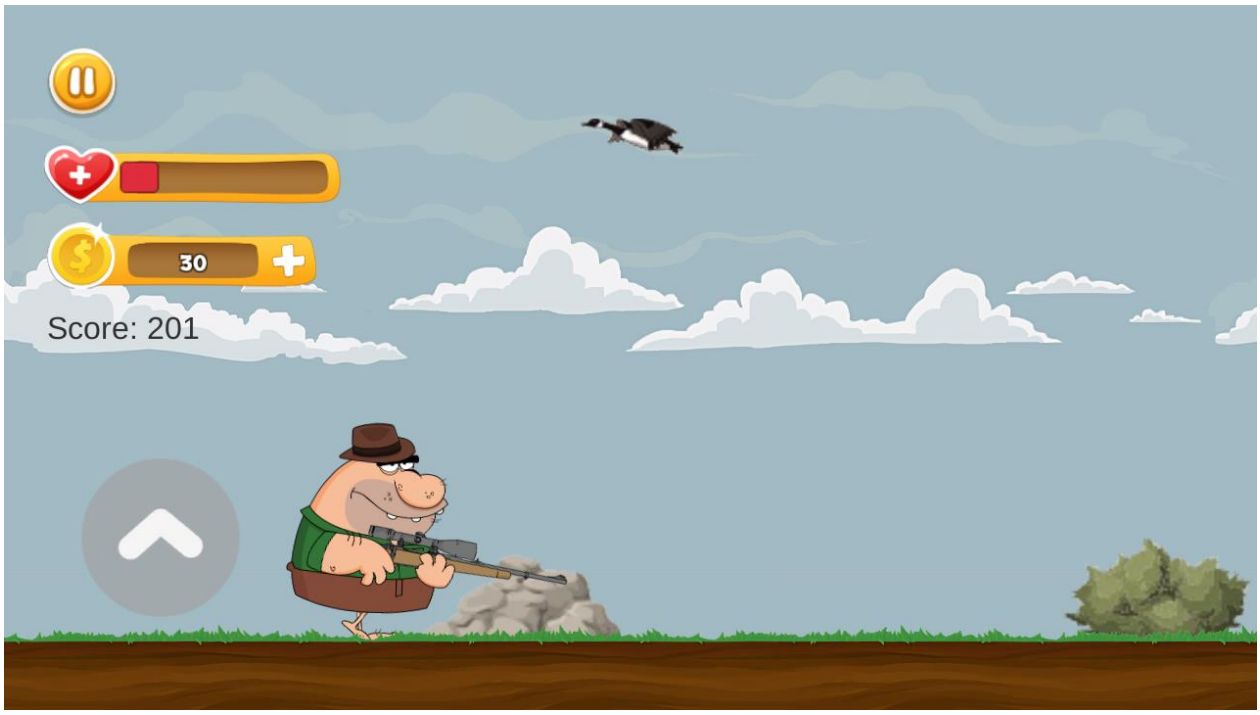


Abbildung 2: Result of the thesis

Danksagung

Wir bedanken uns bei allen Unterstützern, die zum Erfolg des Projektes mit ihrer Hilfe beigetragen haben.

Besonderer Dank gilt unserem Betreuungslehrer Prof. Mag Rupert Obermüller, der uns bei etwaigen Fragen stets zur Seite stand.
Darüber hinaus bedanken wir uns bei der Kontaktperson des Auftraggebers, Dipl. Ing. Matthias Steinbauer, der uns bei der Ideenfindung unterstützt hat.
Ein wichtiger Beitrag zu unserem Projekt kam von Celine Daliot, welche unseren finalen Charakter entworfen hat.

Herzlichen Dank!

Inhaltsverzeichnis

1 Projektumgebung

Im folgenden Kapitel werden die beteiligten Person und deren Verantwortlichkeiten während des Projektablaufs beschrieben.

1.1 Diplomanden

Abbildung 3: Profilbild Simon Angerbauer

Simon Angerbauer

Geburtsdatum	26. März 1997
Adresse	Putznsiedlung 26 4441 Behamberg
Email	simon.angerbauer@gmx.at
Schulausbildung	2003 – 2007 Volksschule Behamberg 2007 – 2011 BRG Steyr 2011 – 2016 HTL für Informatik in Perg
Berufserfahrung	2014: Telekom Austria, 4 Wochen 2015: Johannes Kepler Universität, 4 Wochen

Abbildung 4: Profilbild Roman Socovka

Roman Socovka

Geburtsdatum	30. Juni 1995
Adresse	Kaplanstraße 44 4300 St. Valentin
Email	socovka@gmx.at
Schulausbildung	2002 – 2005 Volksschule Secovce, Slowakei 2005 – 2007 Volksschule Ennsdorf 2007 – 2011 IBM St. Valentin 2011 – 2016 HTL für Informatik in Perg
Berufserfahrung	2013: Salvagnini, 4 Wochen 2014: voestalpine Stahl GmbH, 4 Wochen 2015: Libro, teilzeitbeschäftigt 2016: Elecks, teilzeitbeschäftigt

1.2 Auftraggeber

Auftraggeber war das Institute of Telecooperation an der Johannes Kepler Universität Linz.

Altenbergerstraße 66
4040 Linz

Abbildung 5: [JKU LINZ]

1.3 Projektumfeld

Das Projektumfeld umfasst jene Personen bzw. Anstalten, die im Laufe unserer Diplomarbeit mit uns zusammengearbeitet haben.

1.3.1 Schule

Im Rahmen der Matura an der HTL Perg wurde diese Diplomarbeit als Bestandteil dieser durchgeführt.

Abbildung 6: HTL Perg Logo [HTL PERG]

Machlandstraße 48
4320 Perg

Tel. 0 72 62 / 53 9 26

1.3.2 Betreuungslehrer

Abbildung 7: [OBERMÜLLER, Rupert]

Aufgrund der technischen sowie organisatorischen Aspekte unserer Arbeit stand während der Entwicklungsphase der Diplomarbeit Herr Professor Prof. Mag. Rupert Obermüller zur Verfügung.

Werdegang:

- Lehramtstudium Mathematik, Leibeserziehung
- HTL I Linz
- HTL Leonding

1.3.3 Ansprechpartner des Auftraggebers

Abbildung 8: [STEINBAUER, Matthias]

Auftraggeber der Diplomarbeit ist die Johannes Kepler Universität Linz.

Aufgrund seiner fundierten Kenntnisse im Bereich Cross-Platform Entwicklung war der dortige Ansprechpartner Dipl. –Ing. Matthias Steinbauer.

1.4 Projektorganigramm

Das Projektorganigramm stellt die Beziehungen zwischen den jeweiligen Projektpartnern graphisch dar.

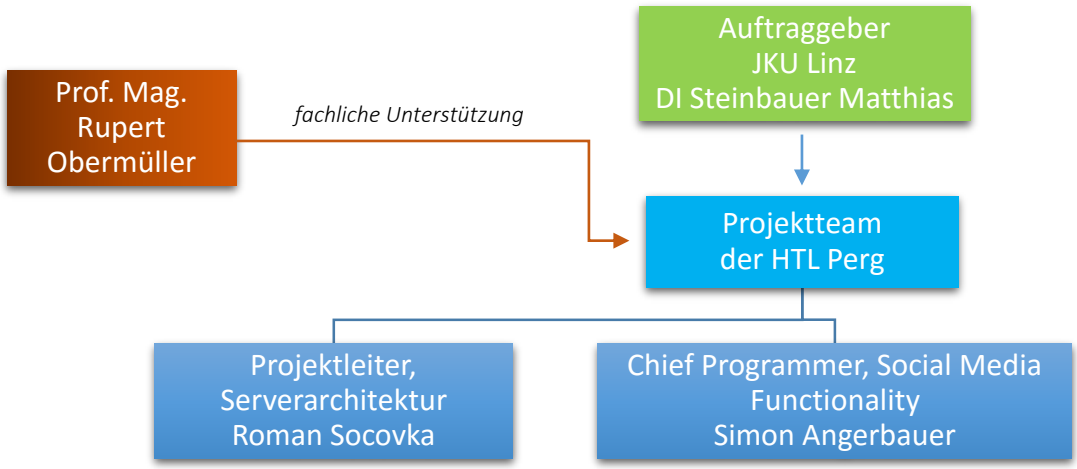


Abbildung 9: Projektorganigramm

1.5 Zuständigkeiten

Um die Verantwortlichkeiten der einzelnen Projektbeteiligungen darzustellen dient folgende IMV-Matrix.

Die Bezeichnungen setzen sich aus folgenden Zuständigkeiten zusammen:

I...Information

M...Mitarbeit

V...Verantwortung

Funktion	Angerbauer	Socovka	Steinbauer	Obermüller
Projektauftrag	MV	MV	IM	I
Pflichtenheft	MV	MV	I	I
Designentwurf	MV	MV	I	I
Implementierung des Prototypen	MV	MV	I	I
Anbindung an Datenbank	I	MV	I	I
Anbindung an soziale Medien	MV	I	I	I
Testen	MV	MV	I	I
Diplomschrift	MV	MV	I	I
Projektabnahme	MV	MV	IM	I

Tabelle 1: IMV-Matrix

2 Entstehung

Das Kapitel *Entstehung* beschreibt das konkrete Thema, die Aufgabenstellung, die Zielsetzung sowie die Ausgangssituation unserer Diplomarbeit.

2.1 Thema und Aufgabenstellung

Unsere Aufgabenstellung beschreibt die Entwicklung eines Cross-Platform-Spiels.
Der Begriff Cross-Platform bedeutet "plattformunabhängig" und heißt, dass das von uns produzierte Ergebnis auf mehreren Plattformen ausführbar ist.

Um das Spielerlebnis zu erweitern, soll das Spiel an soziale Netzwerke angebunden werden.
Dies hat den Vorteil, dass der Spieler nicht unter einem Benutzernamen spielt, sondern sein Name beim Charakter angezeigt wird. Somit ist dieser für den Spieler nicht nur eine „fiktive Figur“, sondern ein reales Objekt, das seinen Namen trägt.
Außerdem bietet der Zugang zu sozialen Netzwerken die Möglichkeit, die eigenen Freunde herauszufordern und gegen diese zu spielen.

Zusätzlich werden Spieldaten wie Spiellänge und gesammelte Punkteanzahl am Server gesammelt und ausgewertet.

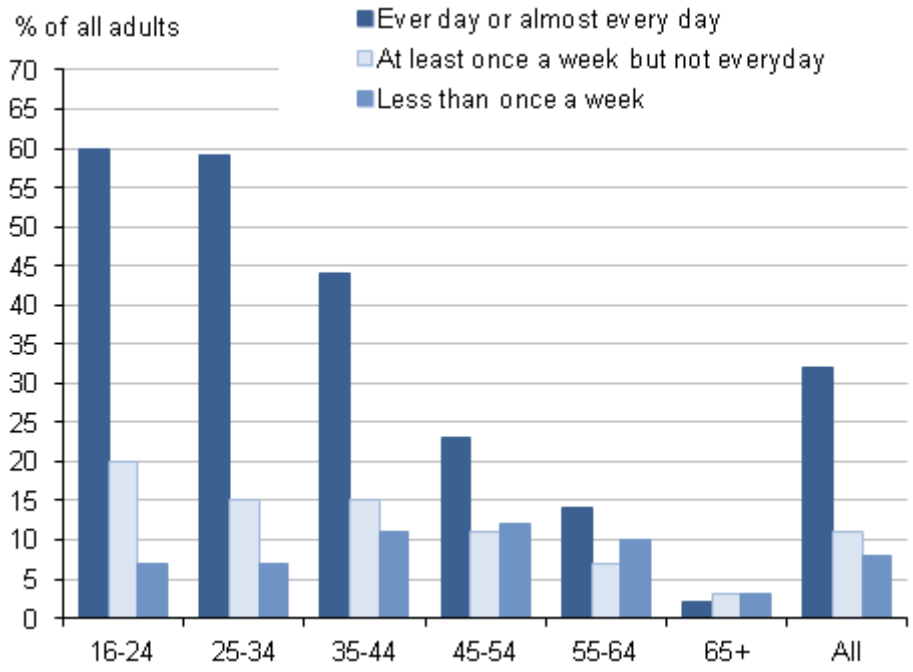


Abbildung 10: Internetnutzungsstatistik nach Alter [ONS.GOV.UK]

Die Hauptbenutzergruppe, die unser Produkt anvisiert, ist vor allem die jüngere Generation. Anhand der beigefügten Statistik ist klar zu erkennen, dass diese maßgebend das Internet benutzt und somit auch als Schlussfolgerung potentiell die höchste Aktivität aufweisen wird.
Der Grund, wieso unsere Diplomarbeit als Cross-Platform-Spiel finalisiert wird, ist, weil die Gerätezahlen zum Großteil für die mobilen Plattformen mit den Betriebssystemen Android (z.B. Samsung-Geräte) sowie iOS (Apple-Geräte) stetig am Steigen sind. Um aber auch lokal am Rechner spielen zu können, wird mittels der Unity Engine eine Version für Windows-Desktopgeräte entworfen.

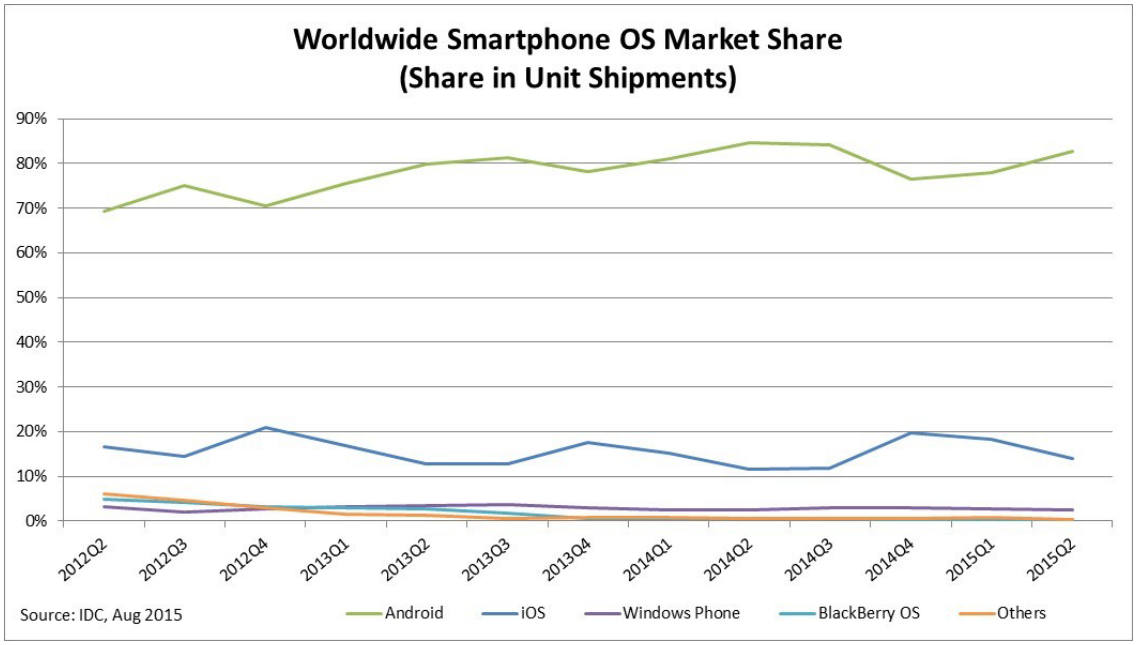


Abbildung 11: Marktanteilstatistik zu mobilen Betriebssystemen [IDC]

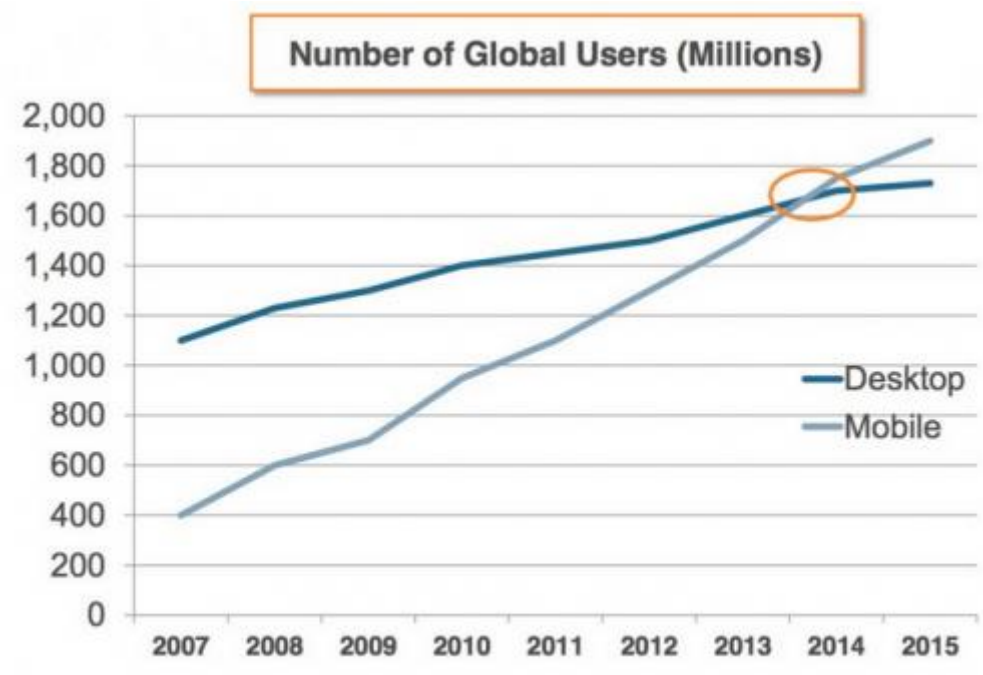


Abbildung 12: Mobil-und Desktopgerätestatistik [SMARTINSIGHTS.COM]

2.2 Zielsetzung

Das Ziel unserer Diplomarbeit ist es, eine spielbare Version eines 2D Endless-Runner Spiels mit der Unity Game Engine zu entwickeln.
Dabei stehen der Unterhaltungswert des Spiels sowie die Kommunikation der Spieler mithilfe der Facebook-Anbindung im Vordergrund.

Dem Spieler ist es möglich, alleine zu spielen oder im Multiplayer-Modus gegen andere Spieler anzutreten.

2.2.1 Ziel für den Auftraggeber

Ziel für das Institute of Telecooperation ist es ein weiteres Referenzprojekt zu den verschiedensten Plattformen für Cross-Platform-Entwicklung zu erhalten. Es wurden bereits einige Projekte realisiert, allerdings noch keines mit der Unity Game Engine. Das Projekt bietet daher die Möglichkeit Erfahrungen mit der entsprechenden Plattform zu sammeln und einen Prototyp zu implementieren.

2.2.2 Geschäftsziele

Man könnte das Geschäftsziel unserer Diplomarbeit als eine Art Vision ansehen. Das Ziel aus der Anwendersicht ist, mithilfe des entwickelten Spieles die Langeweile zu überbrücken.

Der Benutzer sollte das Gefühl vermittelt bekommen, beim Spielen des Spieles „Hunt and Run“ die Zeit verstreichen zu lassen. Durch eine intuitive Oberfläche soll vor allem die Einfachheit der Bedienung hervorgehoben werden, um frustrierte Anwender zu vermeiden. Auch soll das Spielerlebnis durch die Verknüpfung mit der „Social Media Platform“ angeregt werden. mithilfe des Facebook API soll der User seine Freunde herausfordern und sich mit diesen im direkten Vergleich messen können.

2.2.3 Produktziele

- Ziele für das Back-End
    - MySQL
        - Datenbank zur Speicherung von Daten
    - WildFly
        - Schnittstelle zur Datenbank
        - Servlets zum Datenaustausch
    - Facebook API
        - Spielerdatenerfassung

- Ziele für das Front-End
    - Verwendung der Game Engine Unity
    - Intuitive Bedienung des Menüs
    - Einfache Spielgestaltung
    - Schnelle Reaktionszeiten auf Benutzereingaben
    - Grundsätzliche Unterstützung der Windows-sowie Android-Plattform

2.3 Ausgangssituation

Für die Spieleentwicklung gäbe es einige Kandidaten an Plattformen bzw. Technologien, mit denen das Spiel implementiert werden könnte. Sei es Marmelade SDK, welches auf Lua beruht, Apache Cordova, welches auf HTML bzw. Javascript beruht oder zahlreiche andere.
Dann gibt es aber auch noch Unity. Unity wurde vom Projektteam als Game-Engine und Entwicklungsumgebung gewählt, da es auf C# beruht und das Projektteam so die bereits vorhandenen Kenntnisse in den Entwicklungsprozess einbauen kann und ein besseres Endprodukt erreichen kann.
Auch die Plattform für Unity ist überzeugend. Gute Dokumentation, starke Community und verständliche Tutorials.
Außerdem existiert auf dem Institute of Telecooperation noch kein Referenzprojekt zu Unity. So sollen für den Auftraggeber Erfahrungen mit der Game-Engine und Entwicklungsumgebung Unity gesammelt werden.

3 Planung

Das Kapitel *Planung* umfasst die konkrete Umsetzung des Pflichtenheftes, sowie den Struktur-, Ablauf-und Ressourcenplan. Auch wird eine grobe Aufwandsschätzung angefügt.

3.1 Pflichtenheft

In diesem Kapitel werden die zentralen Punkte aus dem Pflichtenheft aufgezeigt.
Grundsätzlich soll dieses Kapitel den Ausmaß unserer Lösung zeigen, aber auch die Abgrenzungskriterien festlegen, die klar von der eigentlichen Lösung getrennt sind.

3.1.1 Grenzkriterien

Folgender Abschnitt beschreibt die festgelegten Grenzkriterien dieser Arbeit.

3.1.1.1 Wunschkriterien

Als klares Wunschkriterium unserer Diplomarbeit ist das lauffähige Endless-Runner-Spiel deklariert.
Es soll für den Benutzer des Spieles möglich sein,

- den Facebook-Login zu benutzen
- Offline-Spiele betreten zu können
- Online-Spiele gegen Freunde betreten zu können
- Spieldaten einzusehen
- neue Highscores an den Sever zu übertragen

3.1.1.2 Abgrenzungskriterien

Da sich unsere Diplomarbeit mit der Programmierung eines Spieles auseinandersetzt, nicht aber mit der Veröffentlichung im App Store, ist das als klarer Abgrenzungspunkt in unserem Pflichtenheft festgehalten.
Außerdem ist

- kein Support der Anwendung vorgesehen;
- keine Unterstützung für nicht aufgelistete Plattformen vorgesehen;
- es vorgesehen, nur Geräte mit einer Android-Version >4.2 zu unterstützen;
- die Unterstützung der Geräte, die <1GB RAM besitzen, nicht vorgesehen;

3.1.2 Produkteinsatz

Das Kapitel *Produkteinsatz* behandelt die Anwendungsbereiche sowie die Zielgruppen des Produktes der Arbeit.

3.1.2.1 Anwendungsbereiche

Die Anwendung dient als Gelegenheitsspiel für Anwender und soll Zeiten der Langeweile in Alltagssituationen überbrücken.

3.1.2.2 Zielgruppen

Die Anwender müssen ein Android Smartphone oder einen Windows Desktop/Tablet besitzen. Des Weiteren ist eine gewisse Geschicklichkeit im Umgang mit Gelegenheitsspielen von Vorteil. Generell sind alle Personen, die diese Kriterien erfüllen, Zielgruppe unserer App. Jedoch sind es vor allem die jüngeren Generationen, denen unser Angebot entsprechen wird. Die Internetnutzung der entsprechenden Generationen sieht man auch an der zuvor bereits gezeigten Statistik: Vgl. Abbildung 10: Internetnutzungsstatistik nach Alter

3.1.3 Produktfunktionen mit GUI-Abbildung

In diesem Kapitel werden die festgelegten Produktfunktionen mithilfe von GUI-Abbildungen festgehalten.

3.1.3.1 Hauptmenü

Abbildung 13: Hauptmenü

Beim Start der Applikation wird das Hauptmenü angezeigt. Von diesem Menü aus kann der Benutzer folgende Aktionen tätigen

- Grüner „Play“ Button: Ein Einzelspieler-Spiel wird gestartet
- Oranger „Multiplayer“ Button: Ein weiteres Menü wird angezeigt, welches die offenen und bereits absolvierten Herausforderungen gegen Freunde beinhaltet
- Blauer Facebook Button: Es wird ein Login über Facebook durchgeführt

3.1.3.2 Login

Abbildung 14: Hauptmenü mit angemeldetem Benutzer

Wenn der Benutzer im Hauptmenü auf den blauen Facebook Button geklickt hat wird er über Facebook angemeldet und sein Facebook-Name und Facebook-Foto werden links in einem Dialog angezeigt.
Der Benutzer kann nun, da er eingeloggt ist auf die Multiplayer-Funktionalität zugreifen.

3.1.3.3 Multiplayer-Ansicht

Abbildung 15: Multiplayer-Menü

In diese Ansicht sieht der Benutzer seine offenen und abgeschlossenen Herausforderungen mit Freunden.
Er kann mithilfe der Buttons, die sich neben den Facebook-Namen und Facebook-Fotos befinden die Herausforderung annehmen bzw. die Herausforderung noch einmal wiederholen.
Mittels des „Plus“ Buttons wird die Herausforderungsansicht aufgerufen, auf der neue Herausforderungen gestartet werden können

3.1.3.4 Herausforderungsansicht

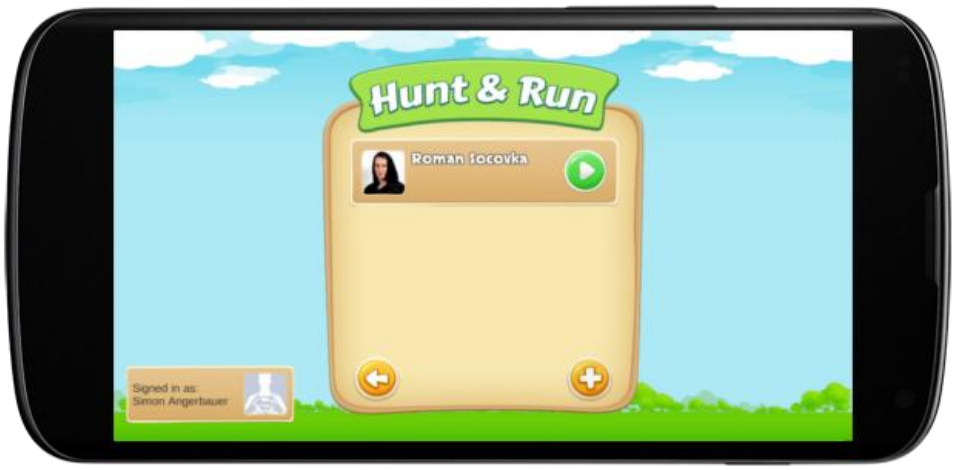


Abbildung 16: Herausforderungsansicht

Es werden alle Facebook-Freunde angezeigt, die diese Applikation spielen.
Diese können mittels des grünen „Play“ Buttons herausgefordert werden.
Über den orangenen „Plus“ Button können über Facebook Freunde eingeladen werde, die dieses Spiel noch nicht spielen.

3.1.3.5 Spielaktivität

Abbildung 17: Spiel

Hat der Benutzer ein Einzelspieler-Spiel vom Hauptmenü aus gestartet, oder hat er eine Herausforderung vom Multiplayer-Menü gestartet, so wird die Spielaktivität ausgeführt.

Der Benutzer findet folgende Interface-Komponenten vor:

- Oranger Pause Button: Das Spielgeschehen wird pausiert und es wird das Pause-Menü angezeigt
- Roter Lebenspunktebalken: Dieser Balken zeigt, wieviel Lebenspunkte der Spieler noch hat. Sind die Lebenspunkte auf 0 ist das Spiel zu Ende und es wird das GameOver-Menü angezeigt.
  Der Spieler verliert Lebenspunkte, wenn er die Hindernisse (Büsche, Steine) berührt.
- Münzenanzeige: Zeigt die Anzahl an Münzen, die der Benutzer hat. Die Münzen erhöhen sich, wenn der Spieler Tiere abschießt.
- Score: Die Anzahl der Punkte, die der Benutzer dieses Spiel erreicht hat. Sie berechnet sich aus der gelaufenen Distanz und den Münzen dieser Runde.
- Sprung-Button: Der Benutzer kann durch diesen Button mit der Spielfigur springen und so den Hindernissen ausweichen.
- Restlicher Bildschirm: Mit Drücken auf den Hintergrund schießt der Benutzer eine Kugel in diese Richtung und kann so die Tiere abschießen.

3.1.3.6 Game-Over-Menü

Abbildung 18:Game-Over-Menü

Wenn die Lebenspunkte des Spielers in der Spielaktivität auf 0 gefallen sind wird dieses Menü angezeigt.
Der Benutzer sieht seine gelaufene Distanz, seine gesammelten Münzen und den Score dieses Spiels.
Er kann ein neues Spiel starten oder zum Hauptmenü zurückkehren.

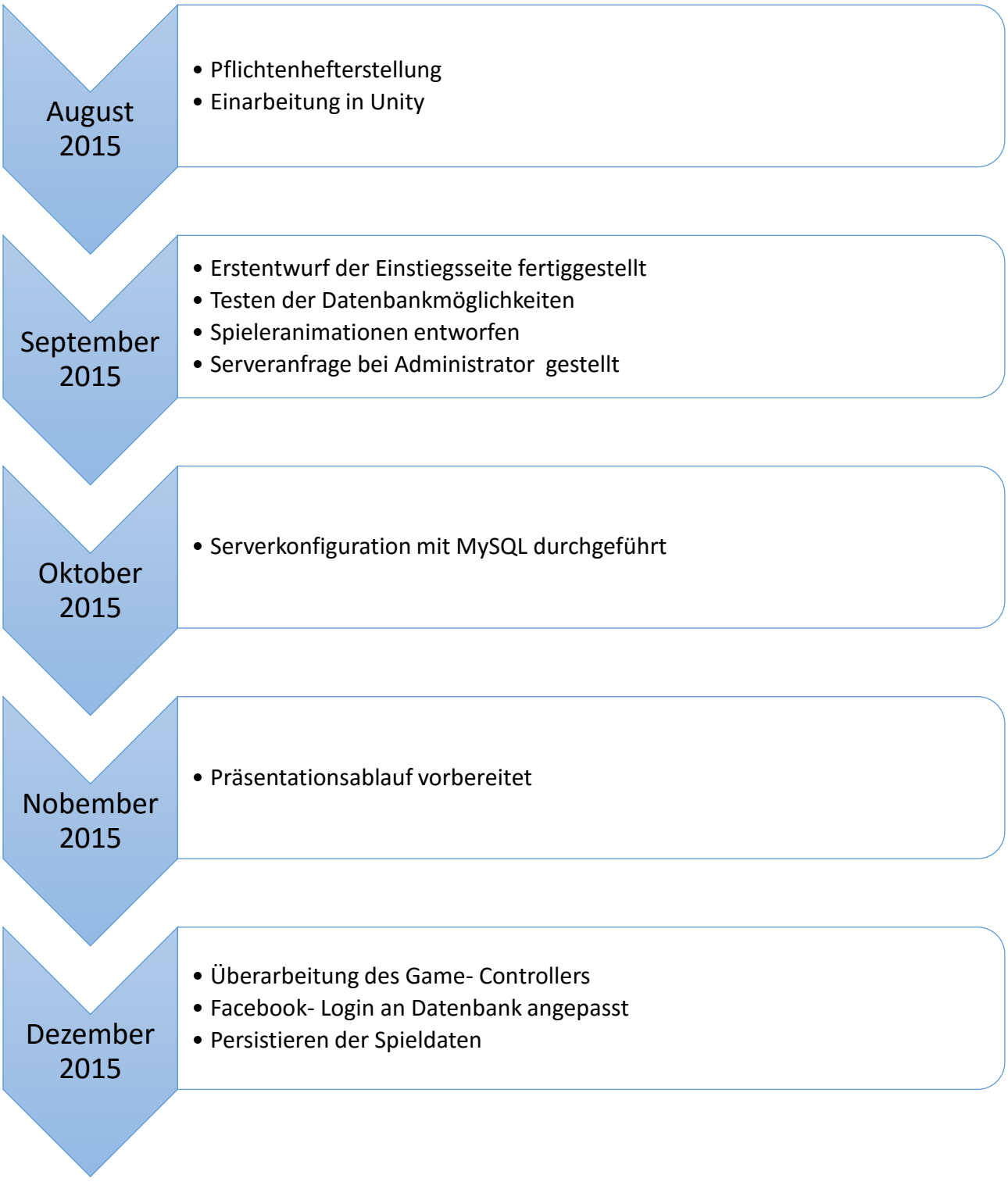
3.1.3.7 Pause-Menü

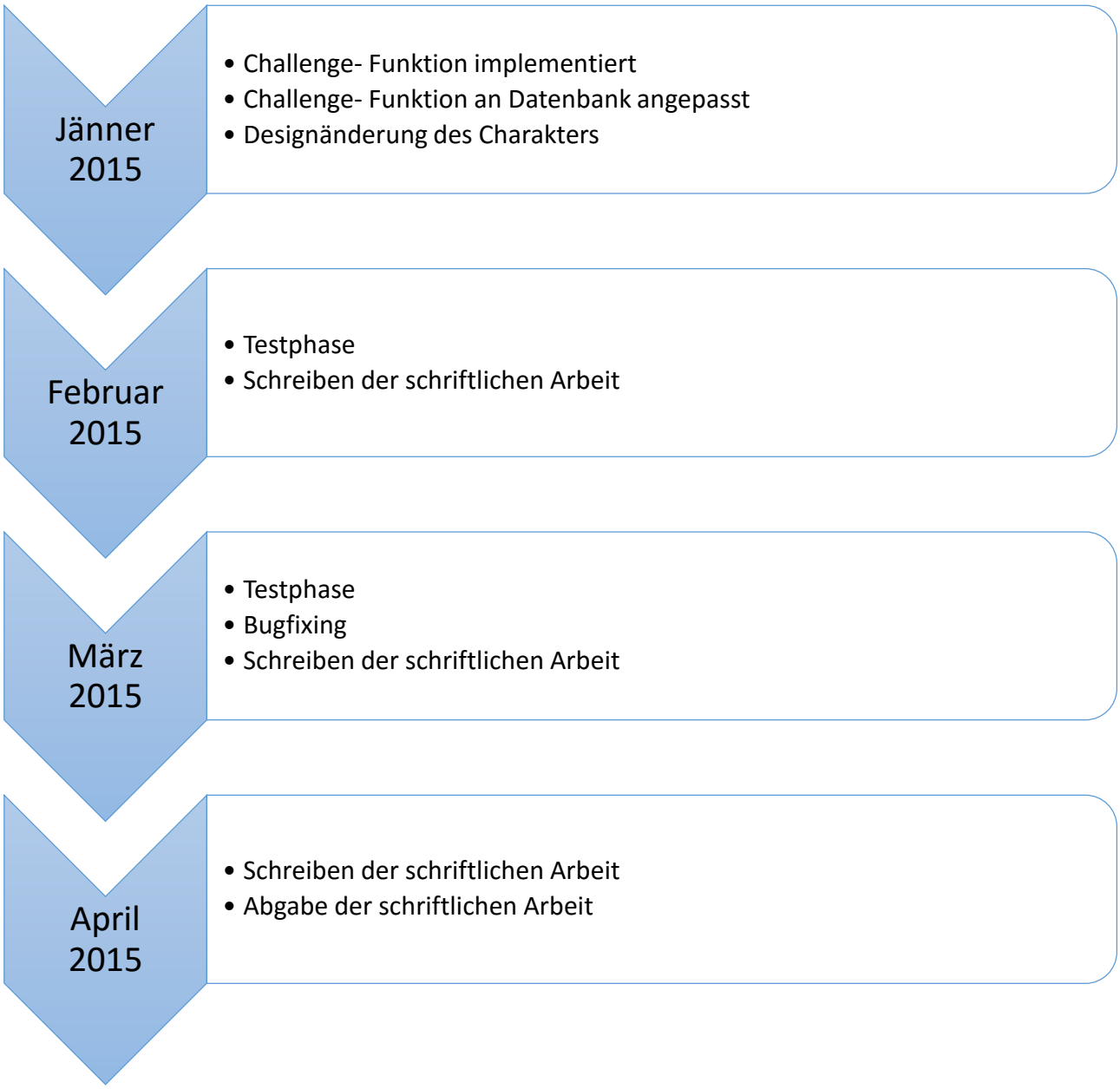
Abbildung 19: Pause-Menü

Wenn der Benutzer während eines Spiels auf den Pause-Button gedrückt hat wird ihm diese Ansicht angezeigt.
Er kann nun die Pause wieder aufheben, ein neues Spiel starten, oder zum Hauptmenü zurückkehren.

3.2 Projektablaufplan

Der Projektablaufplan stellt den gesamten Ablauf des Projektes von August 2015 bis April 2016 graphisch dar. Dabei sind jedem Monat in diesem Zeitraum gewisse Meilensteine zugewiesen.




Jänner 2015
• Challenge- Funktion implementiert
• Challenge- Funktion an Datenbank angepasst
• Designänderung des Charakters
Februar 2015
• Testphase
• Schreiben der schriftlichen Arbeit
März 2015
• Testphase
• Bugfixing
• Schreiben der schriftlichen Arbeit
April 2015
• Schreiben der schriftlichen Arbeit
• Abgabe der schriftlichen Arbeit

3.3 Projektstrukturplan

Der Projektstrukturplan zeigt graphisch die grobe Unterteilung der funktionalen Leistungen unseres Produktes.

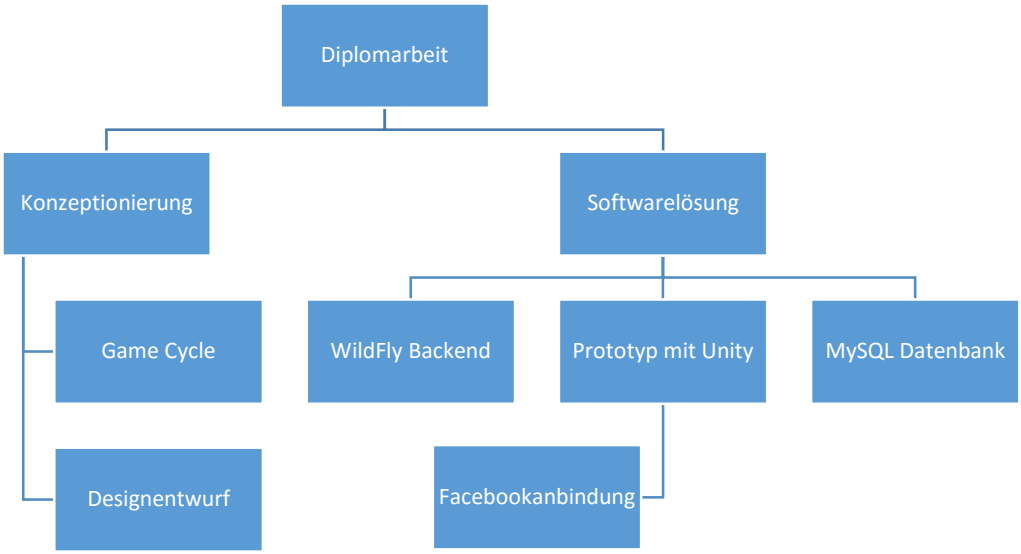


Abbildung 20: Projektstrukturplan

3.4 Ressourcenplan

Der Ressourcenplan beinhaltet die geplanten Ressourcen, die im Laufe unserer Diplomarbeit eingesetzt werden.

3.4.1 Personalressourcen

Die Personalressourcen unserer Diplomarbeit beschränken sich jeweils auf die unten aufgeführten Projektmitglieder.

Ressource	Rolle
Roman Socovka	Projektleiter • Serverarchitektur (main focus) • Game Cycle Programmierung • Marketing • Projektflusskontrolle
Simon Angerbauer	Chief Programmer • Social-Media Anbindung (main focus) • Game Cycle Programmierung • Qualitätskontrolle

Tabelle 2: Personalressourcen

3.4.2 Sach- und Anlageressourcen

Das Kapitel *Sach- und Anlageressourcen* enthält die verwendeten Ressourcen während der Entwicklung und der Testung der Applikation.

3.4.2.1 Projektentwicklung

- Desktop-Geräte
  - Die Entwicklung des Projektes erfolgt hauptsächlich am Privatgerät der Projektmitglieder.
- Server
  - Um die Spieldaten persistent zu halten wird im September 2015 ein Projektserver beantragt und entsprechend konfiguriert

3.4.2.2 Projekttestung

- Mobile Geräte
  - Android-Geräte
    Um eine Variation an Testgeräten zu besitzen, wird auf beiden privaten Testgeräten der Projektmitglieder geprüft.
    - Samsung Galaxy A5, Android 5.0.1
    - Sony Xperia Z1 Compact, Android 5.1.1
- Desktopgeräte
  - Testen erfolgt auch auf Windows-Desktopgeräten

3.5 Aufwandsschätzung

Bereits zum Zeitpunkt des Projektstarts wurde der Projektaufwand grob von den Projektpartnern in Stunden geschätzt.

Bereich	Geschätzte Stundenanzahl
Pflichtenheft	*10*
Einrichten der Umgebung	*10*
Serverkonfiguration	*35*
Designentwurf	*30*
Implementierung des Prototypen	*50*
Anbindung an die Datenbank	*25*
Schnittstellenprogrammierung Client-Server	*20*
Anbindung an soziale Medien	*50*
Testen	*20*
Bugfixing	*30*
Diplomschrift	*80*
Gesamtstundenanzahl	360

Tabelle 3: Grobschätzung des Aufwandes

4 Konzeptentwicklung

In diesem Kapitel befassen wir uns mit der Entwicklung des Konzeptes für den Ablauf des Spiels und für den Entwurf der Spielszenen.

4.1 Game-Cycle

Um den Ablauf mit allen Möglichkeiten des Spiels festzulegen wurde der sogenannte „Game-Cycle“, also der Spielzyklus, in Microsoft Visio, einer Modellierungssoftware, dargestellt. Der erste Teil (Grafik unten) beschreibt die Funktionalitäten des Hauptmenüs.

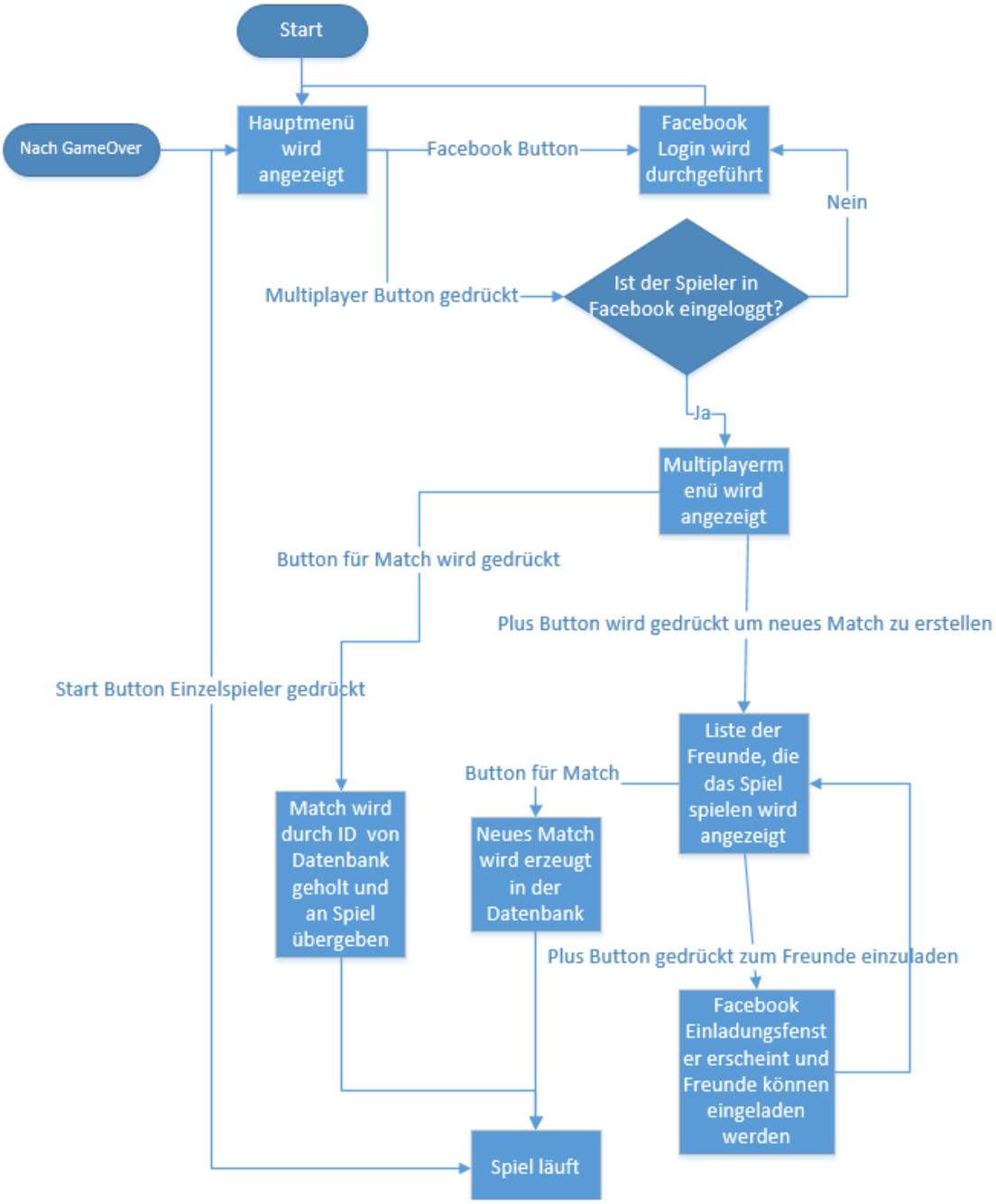


Abbildung 21: Game-Cycle Hauptmenü

Im zweiten Teil des Game-Cycles wird der Ablauf des Spiels dargestellt:

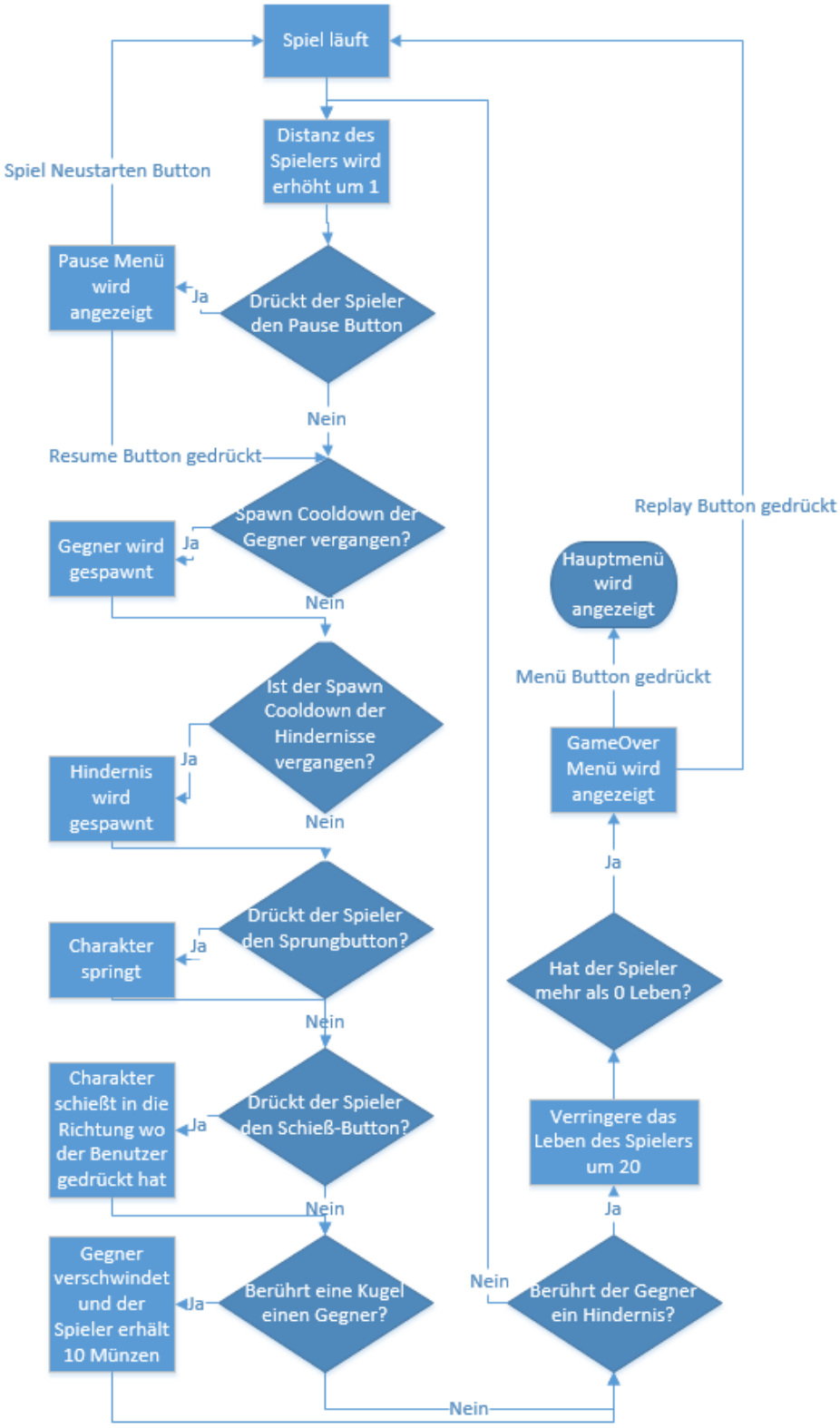


Abbildung 22: Game-Cycle vom Spiel

4.2 Designentwurf

Die Designentwürfe wurden anfangs grob mithilfe des Grafikprogrammes Paint grob skizziert. Die skizzierten Elemente wurden im späteren Verlauf der Diplomarbeit durch freie Grafiken und Vektoren aus dem Internet ersetzt.

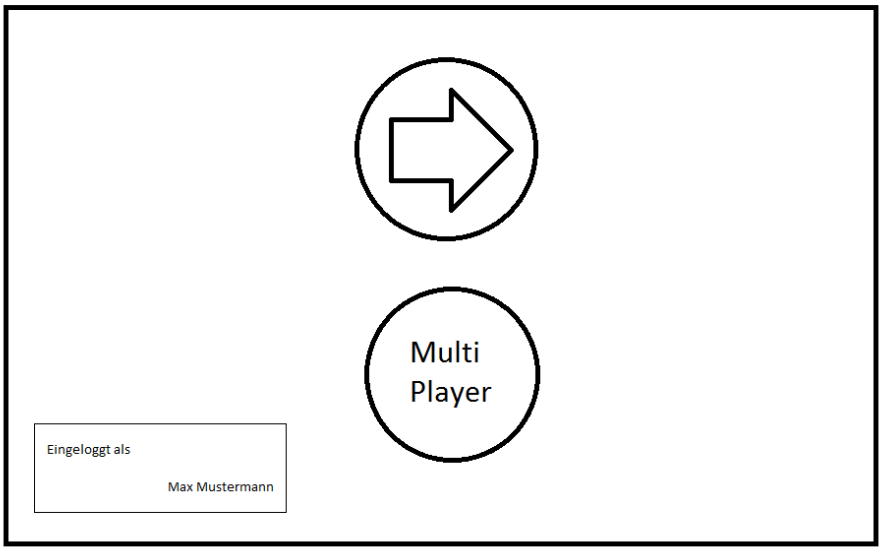


Abbildung 23: Menü-Szene

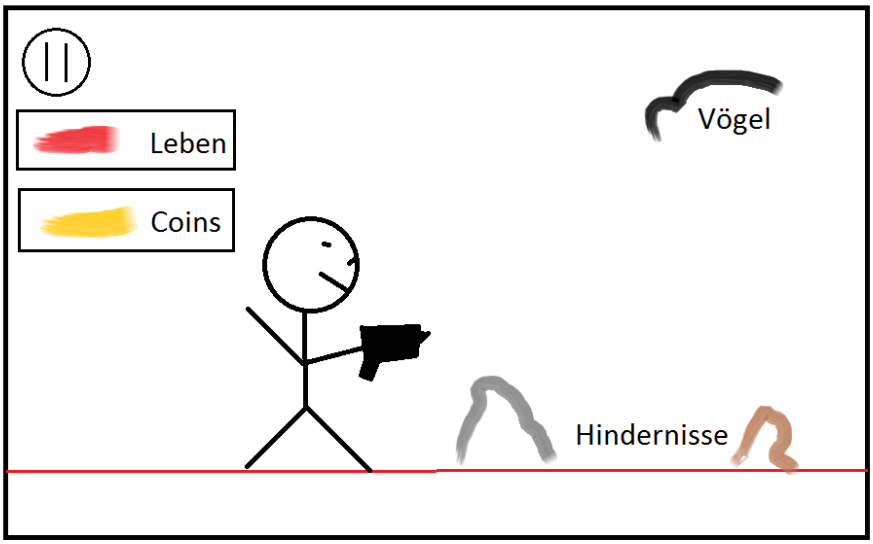


Abbildung 24: Spiel-Szene

5 Entwicklungssysteme

Das Kapitel *Entwicklungssysteme* beinhaltet die verwendeten Technologien, die bei der Umsetzung der Diplomarbeit benutzt wurden. Zusätzlich dazu werden auch die Schnittstellen beschrieben, die das Produkt unserer Diplomarbeit nutzt.

5.1 Verwendete Technologien

Das Kapitel *Verwendete Technologie*n umfasst alle Technologien, mit denen die Diplomarbeit realisiert wurde. Zusätzlich dazu wird jede Technologie mit einem kurzen Steckbrief beschrieben.

5.1.1 Unity

Unity ist eine Laufzeitumgebung und Entwicklungsumgebung für Spiele und andere Grafikanwendungen in 2D und 3D Grafik.

Abbildung 25: Unity Logo [UNITY]

Entwickler	Unity Technologies
Aktuelle Version	5.3.1
Lizenz	EULA

Tabelle 4: Unity Informationen [WIKIPEDIA]

5.1.1.1 Entwicklungsumgebung

Die Entwicklungsumgebung von Unity ist der Unity Editor. Dieser kann im 2D und 3D Modus betrieben werden. Er stellt im Hauptfenster die geöffnete Szene bestehend aus Game-Objekten mithilfe eines Szenengraphen, also einer Datenstruktur, mit dieser die logische und räumliche Anordnung der Szene beschrieben werden, dar.
Eine Applikation kann aus verschiedenen Szenen bestehen und diese entsprechend während der Laufzeit wechseln. Zum Beispiel eine für das Hauptmenü und eine andere für das laufende Spiel.

Game-Objekte sind alle Objekte, die in einer Szene dargestellt werden. Also zum Beispiel Charaktere und Hintergrundobjekte. Sie sind Container und enthalten verschiedenste Komponenten, die die Funktionalität und das Aussehen des Game-Objekts festlegen.
Das Game-Objekt muss zumindest eine Transform-Komponente haben, welche die Position und Orientierung festlegt. Die anderen Komponenten können im Inspector des Unity Editors angepasst und konfiguriert werden. Zum Beispiel können im 2D Modus den Game-Objekten Sprites, also 2D Bilder, angehängt werden und es kann so das Aussehen der Objekte definiert werden.

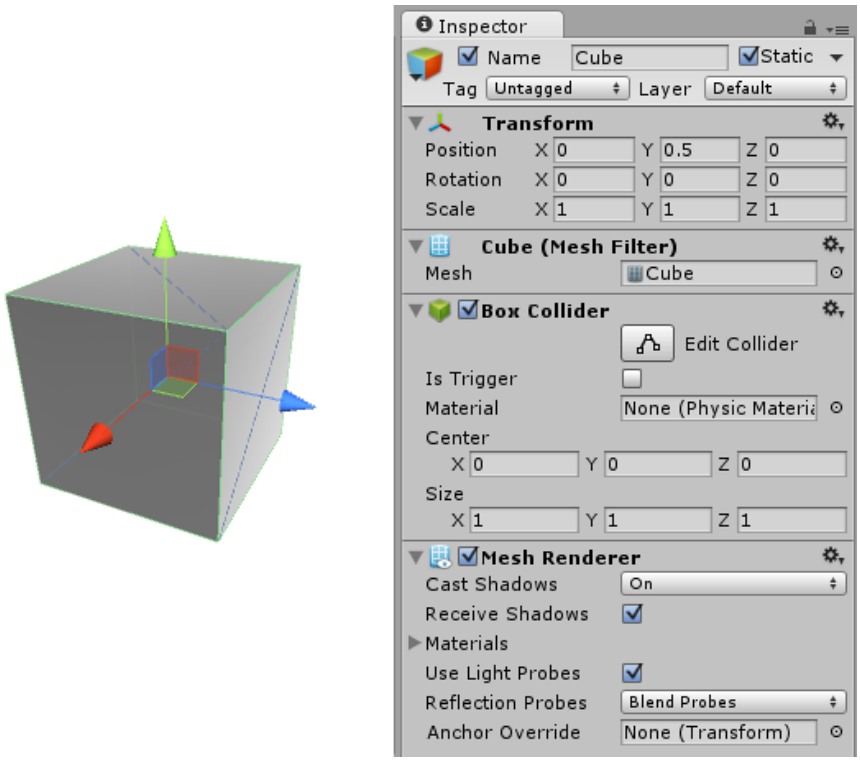


Abbildung 26: Unity Inspector

Auch während das Spiel läuft können im Inspector die Parameter am Game-Objekt geändert werden und so das Verhalten direkt getestet werden.

Diese Komponenten können auch selbst erstellte Skripte sein. Skripte werden in allen möglichen Situationen benötigt. Sei es um Events abzufangen, auf Eingabe des Spielers zu reagieren, oder die Bewegung eines Charakters zu steuern. Diese Skripte können in C# oder JavaScript geschrieben werden.
Die Verbindung mit der Unity Game-Engine wird durch die Vererbung von der Klasse *MonoBehaviour* durchgeführt. Durch diese Vererbung können die Skripte als Komponenten an die Game-Objekte angehängt werden. Das *MonoBehaviour* legt zwei Funktionen fest. Update und Start. Die Update Funktion wird beim Frame-Update aufgerufen und sie enthält alles, was über Zeit behandelt werden muss. Zum Beispiel Bewegungen und Behandeln von Benutzerinteraktionen. Die Start Funktion wird aufgerufen, bevor die Szene geladen wird und wird so für den Initialisierungscode verwendet.

Als Umgebung zur Erstellung der C# Skripte kann entweder das mitgelieferte MonoDevelop verwendet werden, oder auch Visual Studio, welches eine komfortablere Entwicklung ermöglicht als MonoDevelop.

Im Editor kann mithilfe der Build-Einstellungen die Zielplattform aus vielen möglichen Zielplattformen festgelegt werden und so Cross-Platform Unterstützung gewährleistet werden.

Abbildung 27: Unity Zielplattformen [UNITY]

5.1.1.2 Testen

Durch die Verwendung von Visual Studio/MonoDevelop bietet sich die Möglichkeit, den Debugger dieser IDEs an den Prozess von Unity anzuhängen und während der Benutzung des Unity Editors zu verwenden.

5.1.2 WildFly Application Server

Der *Wildfly Application Server* kommt bei der Diplomarbeit zum Einsatz, um eine Schnittstelle zwischen Client und Datenbank zur Verfügung zu stellen.

5.1.2.1 Einleitung

Der WildFly Application Server ist ein Anwendungsserver nach dem Java Enterprise Edition-Standard. Der Server ist in der Programmiersprache Java geschrieben, weshalb bei der Konfiguration und dem Management des Servers auf erlerntes Wissen aus dem Java-Unterricht zurückgegriffen werden kann.

Abbildung 28: WildFly- Logo [WILDFLY]

Entwickler	Red Hat
Aktuelle Version	9.0.2 Final
Lizenz	LGPL

Tabelle 5: WildFly-Informationen [WIKIPEDIA]

Außerdem bietet der WildFly-Server etliche Eigenschaften an, von denen aber nur einige in unserer Diplomarbeitslösung zum Einsatz kommen:

- JPA
- JTA
- Servlets

Mithilfe dieser Technologien ist es möglich, den Datenbankzugriff sowie den Datentransfer vom Server auf den Client so einfach wie möglich zu gestalten.

5.1.2.1.1.1 JPA

Die JPA ist eine Schnittstelle für Java-Anwendungen, die die Zuordnung und die Übertragung von Objekten zu Einträgen der Datenbank vereinfacht.

Die Architektur von JPA wird in folgendem Beispiel verdeutlicht:

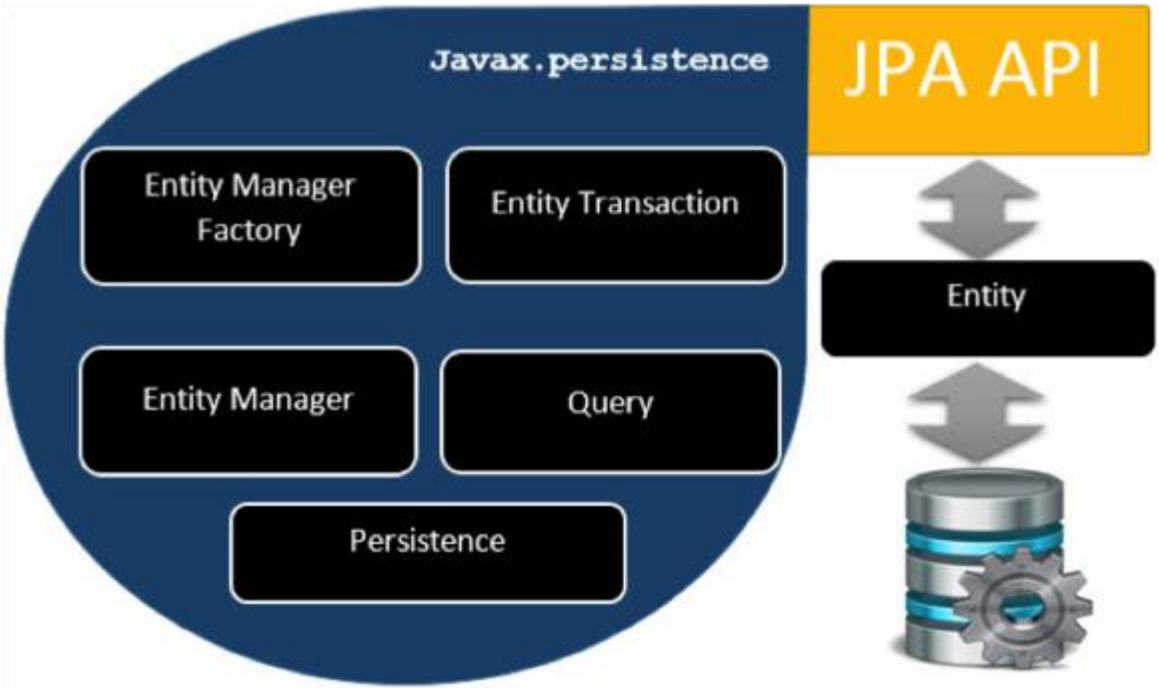


Abbildung 29: JPA-Architektur [TUTORIALSPOINT.COM]

Entity Manager Factory	Erstellt und verwaltet Instanzen von *EntityManager*
Entity Manager	Ein Interface, welches die Schritte zur Persistierung an Datenbankobjekten durchführt.
Entity	Sind die persistierten Objekte, gespeichert als Eintrag in der Tabelle
Entity Transaction	Hat eine *1 zu 1* – Verbindung mit dem EntityManager. Jeder EntityManager verwaltet seine Transaktionen mithilfe von *EntityTransaction.*
Persistence	Die Klasse beinhaltet statische Methoden, um Zugriff auf die EntityManagerFactory-Instanz zu bekommen.
Query	Wird verwendet, um die Abfragemenge mithilfe von Suchkriterien einzuschränken.

Tabelle 6: Übersicht JPA-Architektur

5.1.2.1.1.2 JTA

Die JTA (Java Transaction API) ist eine spezifizierte Programmierschnittstelle, welche den Einsatz von Transaktionen ermöglicht. Eine Transaktion kann zum Beispiel eine zusammenhängende Datenabspeicherung von mehreren Datensätzen sein. Die Transaktionen sorgen dafür, dass die Daten nur dann dauerhaft in der Datenbank gespeichert werden, wenn diese vollkommen fehlerfrei abgeschlossen werden können.
Üblicherweise deklaliert man sich dafür eine Data Source am WildFly-Server, die auf eine reale Datenbank, zum Beispiel auf eine MySQL Datenbank, zugreift.

5.1.2.1.1.3 Servlets

Servlets benutzt man, um Anfragen von Clients am Webserver entgegennehmen und beantworten zu können. Der Inhalt der Antworten vom Server kann dynamisch, also im Moment der Anfrage, erstellt werden. Üblicherweise reagiert man dabei auf GET- bzw. POST-Anfragen, die vom Client gestellt werden, und agiert dann entsprechend der Anfrage.
Die Anfrage kann entweder als Lesezugriff(„Was ist der momentane High Score?“), oder als Schreibzugriff („Ich habe den High Score übertroffen, schreibe es bitte in die Datenbank!“) formuliert werden.
Die Servlets eignen sich in unserer Diplomarbeitslösung deshalb besonders als Schnittstelle für den Datenaustausch zwischen Datenbank und Client, wobei der Datenaustausch im Datenformat JSON erfolgt.

5.1.2.1.1.4 JSON

JSON ist ein kompaktes, einfach lesbares Datenformat, welches zum Zweck des Datenaustausches verwendet wird.
Besonders eignet es sich, weil es praktisch für fast alle Programmiersprachen einen Parser gibt, der die JSON-Daten bzw. den JSON-String in Variablen umwandeln kann.

5.1.2.2 Arbeitsumgebung

Um das am Server eingesetzte Projekt zu erstellen, wird Eclipse als Entwicklungsumgebung eingesetzt.
Konkret wird in unserem Fall als Umgebung die *Eclipse IDE for Java EE Developers* [ECLIPSE.ORG] eingesetzt, da uns diese den Umgang mit WildFly, JPA, JTA sowie der Servlets erleichtert.

Abbildung 30: Eclipse-Logo [ECLIPSE.ORG]

Entwickler	Eclipse Foundation
Aktuelle Version	4.5.1
Lizenz	EPL

Tabelle 7: Eclipse-Informationen [WIKIPEDIA]

Der WildFly-Server besitzt allerdings auch eine eigene Konfigurationsseite, bei der Daten wie die zu veröffentlichenden Projekte, Zugriffsrechte, Datenbankzugriffsdaten sowie etliche andere Informationen gespeichert werden.
Diese ist üblicherweise, falls man nichts an der Serverkonfiguration ändert, standardmäßig über http://localhost:9990/ erreichbar.

5.1.2.3 Testen

Durch Verwendung von Eclipse als Entwicklungsumgebung bieten sich verschiedene Möglichkeiten zur Testung des Projektes.

- Unit-Test
  - Mithilfe eines Unit-Tests wird ein Modul der Serverlösung auf die korrekte Funktionalität getestet. Diesen Modul-Test kann man sich problemlos in der Eclipse-Umgebung einrichten.

- Localhost-Test
  - Mit laufendem WildFly-Server und der http-Applikation Fiddler kann man Netzwerkzugriffe auf den Server simulieren und so Testanfragen am Server bearbeiten.

- Client-Server-Tests
  - Um den gesamten Testrahmen abzudecken, ist es möglich, die Tests mithilfe der Unity-Lösung durchzuführen. Man simuliert hierbei die Zugriffe der Kunden auf die Datenbank sowie den Server. Dabei wird nicht nur der Server auf Fehlermöglichkeiten getestet, sondern auch die Kundengeräte auf Ihre Performance und den Datendurchsatz.

5.1.3 MySQL

MySQL kommt bei der Diplomarbeit zum Einsatz, um eine dauerhafte Datenspeicherung der Benutzer-sowie Spieldaten möglich zu machen.

5.1.3.1 Einleitung

Bei MySQL handelt es sich um einen Server, auf dem Daten persistent gespeichert werden. Auf dem DBMS (Datenbankmanagementsystem) können mehrere Datenbanken erstellt werden, diese wiederum können mehrere Tabellen enthalten. Die Tabellen sind dann wiederum in Spalten, bzw. wenn man es aus der relationalen Sicht betrachtet, in Attribute unterteilt.

Abbildung 31: MySQL- Logo [OTREVA.COM]

Entwickler	Oracle Corporation
Aktuelle Version	5.7.10
Lizenz	GPL

Tabelle 8: MySQL-Informationen [WIKIPEDIA]

Wie der Name MySQL bereits sagt, benutzt man zur Kommunikation mit der Datenbank die Datenbanksprache SQL. Diese wird verwendet, um Datenstrukturen zu definieren, sowie zum Bearbeiten (Einfügen, Verändern, Löschen) und Abfragen von darauf bereits gespeicherten Daten.
Konkret in unserem Fall nimmt allerdings WildFly mithilfe JPA u. JTA das Schreiben des SQL-Code vorweg, da dieser den von uns geschriebenen Inhalt automatisch in SQL-Code übersetzt.

5.1.3.2 Modellierung

Um die Modellierung der Daten der MySQL-Datenbank so einfach wie möglich zu gestalten, benutzt man MySQL Workbench, ein Datenbank-Modellierungswerkzeug. Mit diesem Tool ist es möglich, Modellierung, Erstellung und Instandhaltung von MySQL-Datenbanken zu betreiben.

Abbildung 32: MySQL Workbench- Logo [MICHAELSTULTS.COM]

Entwickler	MySQL Developer Tools Team
Aktuelle Version	6.3.5
Lizenz	GPL

Tabelle 9: MySQL Workbench-Informationen [WIKIPEDIA]

Mithilfe von Drag & Drop ist es möglich, Tabellen in eine Lösung einzubauen. Diese kann man dann, falls erwünscht, mit beliebig vielen Spalten versehen. Primary Keys, Auto-Increment-, Unique-, Binary-sowie Not Null-Werte kann man leicht über das Klicken einer Checkbox generieren lassen.

Falls man Testdaten einfügen möchte, ist dies ebenfalls über die MySQL Workbench möglich.

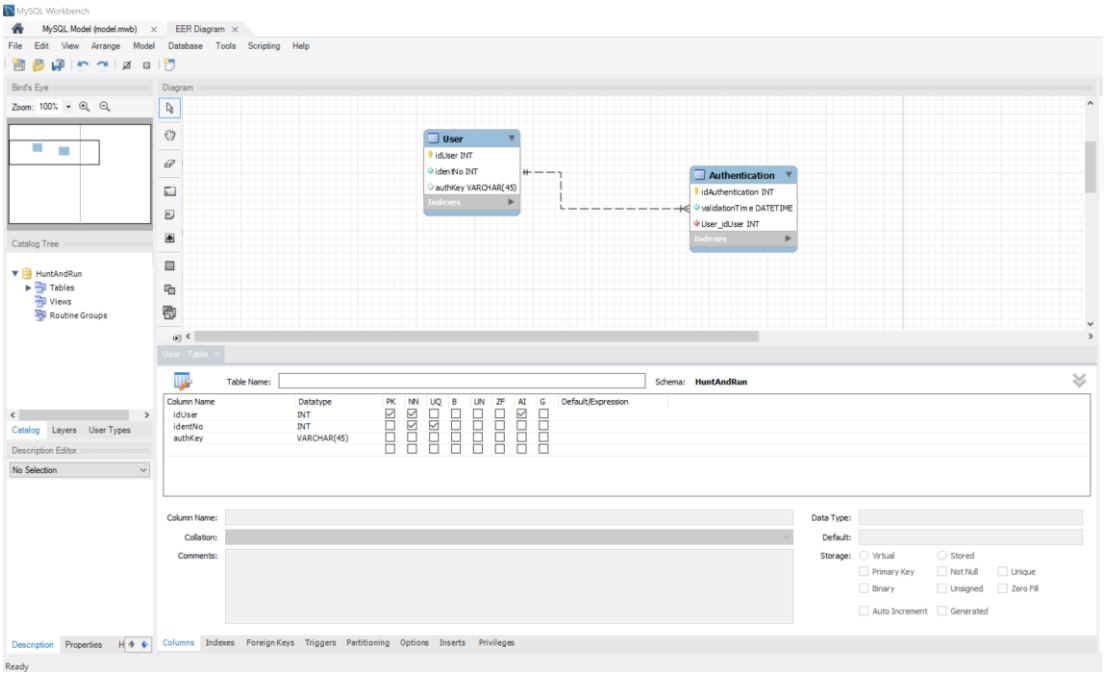


Abbildung 33: Modellierung in MySQL Workbench

Die Funktion *Forward Engineering* wird benötigt, damit man das erstellte Datenbankdesign zu SQL-Statements überführen und es in eine Datenbank laden kann. Dabei werden alle angepassten Einstellungen wie Primary Keys, Inserts etc. mit exportiert.

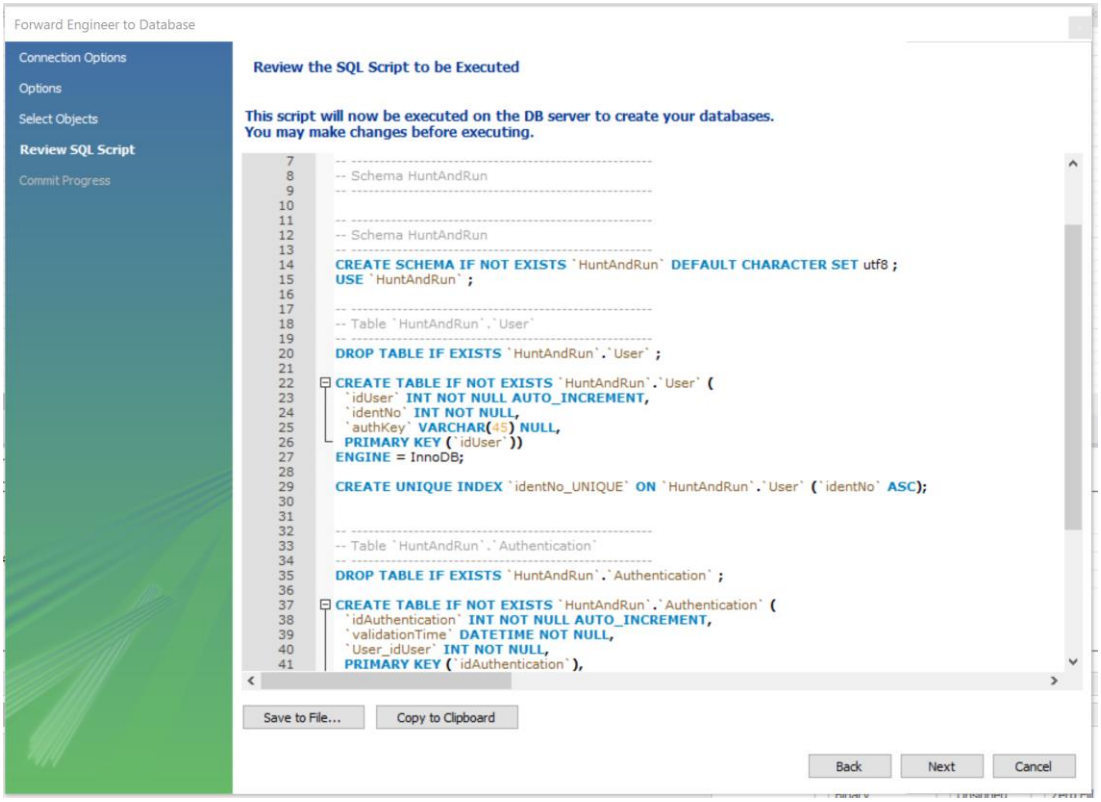


Abbildung 34: Beispielhaftes Forward Engineering

5.1.4 Facebook Graph API

Das Facebook Graph API dient zum Austausch von Daten mit der Facebook Plattform. Es kann zum Beispiel auf Benutzer, Seiten, Veranstaltungen oder Fotos zugegriffen werden. Die Schnittstelle basiert auf HTTP und liefert JSON Ergebnisse zurück.

Der Zugriff auf zum Beispiel Benutzerdaten funktioniert über die Facebook-ID des Benutzers. Dadurch können alle Informationen, wie Name, Vorname, Freunde oder Foto, angefordert werden.

Abbildung 35: Facebook Logo [FACEBOOK]

Eigentümer	Facebook Inc.
Benutzer	1,44 Mrd. monatlich

Tabelle 10: Facebook-Informationen [WIKIPEDIA]

5.1.4.1 SDK

Das Facebook SDK gibt es auf unterschiedlichen Plattformen, so auch auf Unity. Es bietet die Möglichkeit von einfacher Implementierung von Facebook Login, Einladungen, Teilen und es bringt die Schnittstelle für das Facebook Graph API mit.

5.1.4.2 Testen

Die Facebook Developer Plattform bietet die Möglichkeit für eine Applikation, die in Facebook erstellt wurde, automatisch Testbenutzer zu generieren. So kann zum Beispiel die Funktionalität des Logins oder von Einladungen getestet werden. Die Authentifizierung dieser Benutzer findet über die Access Tokens statt, die man der Developer Plattform entnehmen kann.

5.2 Schnittstellen

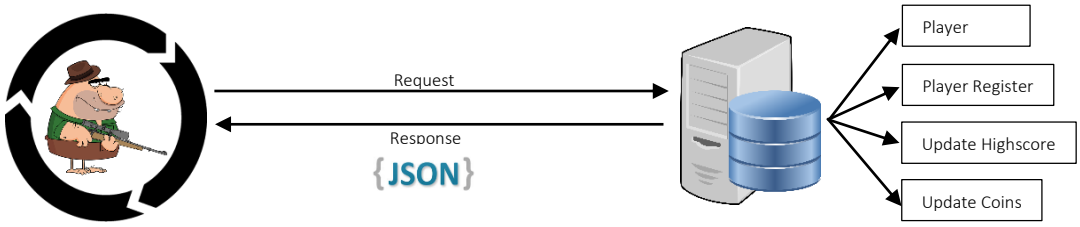
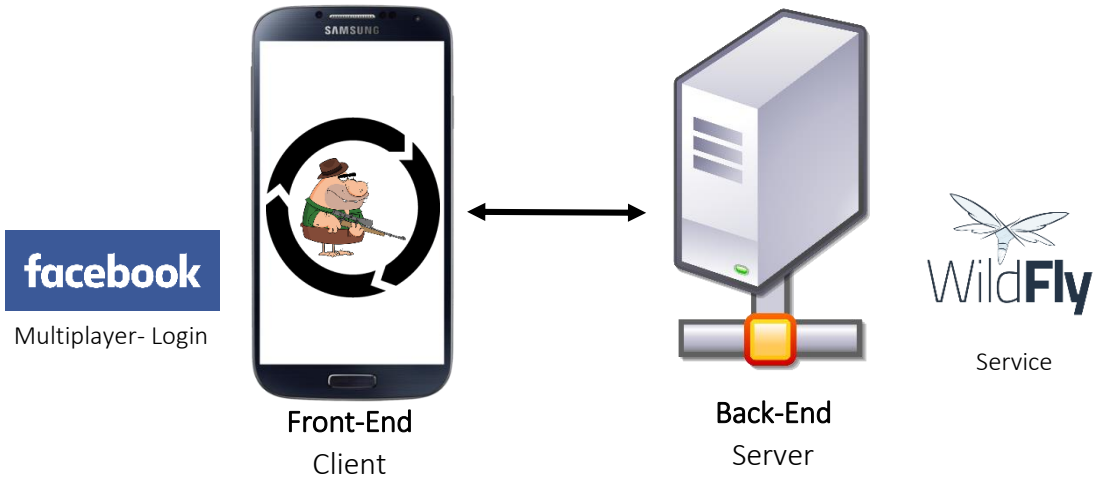


Abbildung 36: Client- Server- Schema

Die Schnittstelle unserer Diplomarbeit zwischen Server und Client ist der Netzwerkzugriff auf die Servlets des WildFly-Servers. Diese Servlets haben die Möglichkeit, Daten abzuprüfen sowie neue Daten in die Datenbank zu schreiben. Der Datenaustausch zwischen Server und Client erfolgt, wie oben bereits erwähnt, im Format JSON.

6 Implementierung

Dieses Kapitel bezieht sich auf die konkrete Umsetzung des Produkts, bestehend aus dem Back-End, also dem Server, und dem Front-End, also dem Client.



6.1 Front-End

Das Front-End besteht zum einen aus der Cross-Platform Unity-Anwendung und zum anderen aus der Anbindung an Facebook. Es wird nun die Implementierung dieser Komponenten näher beschrieben.

6.1.1 Anwendung

Die Anwendung wurde mit der zuvor beschriebenen Technologie Unity implementiert. Im Folgenden werden die implementierten Komponenten und Funktionalitäten näher beschrieben.

6.1.1.1 Szenen

Die Anwendung besteht aus zwei Szenen. Szenen sind Container für die Game-Objekte.
Zum Beispiel:

- UI Controls
- Skripte
- Charaktere

6.1.1.1.1 Hauptmenüszene

Die Hauptmenüszene wird beim Start der Anwendung aufgerufen. Sie besteht aus den Komponenten, welche im Bild rechts abgebildet sind und im folgenden Abschnitt weiter beschrieben werden:

Abbildung 37: Aufbau Hauptmenüszene

Canvas

Das Canvas dient als Container der UI Komponenten.

Folgende Menüs sind als Kind-Elemente dem Canvas angehängt:

- Einstiegsmenü mit Buttons zum
  - Navigieren zum Einzelspielermodus. Dazu wird die Szene auf die Spielszene gewechselt.
  - Navigieren in das Multiplayermenü. Das Einstiegsmenü wird hierfür ausgeblendet und das Multiplayermenü eingeblendet.
  - Einloggen in Facebook. Hier wird der Login-Controller aufgerufen um den Login durchzuführen

- Multiplayermenü:
  Das Multiplayermenü zeigt alle Spiele des Benutzers gegen seine Facebook-Freunde an. Dazu ruft der Hauptmenü-Controller, das Skript zur Bereitstellung des Hauptmenüs, vom Data-Service, dem Skript, dass den Zugriff auf das Service bereitstellt, die entsprechenden Spieldaten ab und baut die Liste dynamisch nach unterschiedlichen Prefabs, das sind vor Laufzeit erstellte Game-Objekte, die nur noch initialisiert und mit Daten befüllt werden müssen, auf, je nachdem, ob der Benutzer gewonnen, verloren oder noch nicht gespielt hat. Für die einzelnen Prefabs werden die Informationen der Gegner vom Facebook API abgerufen und angezeigt.

  Die Prefabs enthalten einen Button zum Starten eines Spiels gegen den entsprechenden Benutzer. Hierfür wird hinter den Buttons die jeweilige eindeutige Facebook-ID hinterlegt und aufgrund dieser kann der Eintrag in der Datenbank über das Data-Service identifiziert werden und das Spiel über das Wechseln in die Spielszene gestartet werden.

- Freundeslistenmenü:
  Hier werden mittels Facebook API alle Facebook-Freunde ermittelt, die auch die Anwendung installiert haben. Für jeden Freund wird mittels Prefab ein Eintrag in der Liste initialisiert und die Freunde können dann herausgefordert werden. Für die Herausforderung wird über das Data-Service ein Eintrag in der Datenbank erstellt.

- Facebook-Menü:
  Das Facebook Menü („SignedInMenu“ in Abbildung 37) dient zur Anzeige des jeweils eingeloggten Benutzers. Es wird nach dem erfolgreichen Login aufgerufen und mit den Daten vom Facebook API befüllt.

Event-System

Zur Bereitstellung der Button-Funktionalität im Canvas wird das Event-System benötigt. Dieses wird automatisch beim Erstellen eines Canvas erzeugt.

Facebook-Manager, Login-Controller, Hauptmenü-Controlle & Data-Service

Diese Komponenten sind leere Game-Objekte in der Szene, sie sind also nicht sichtbar. Sie enthalten nur jeweils eine Skript-Komponente, welche beim Start der Szene ein Objekt der Klasse initialisiert. Für die Aufgaben der Skripte siehe 6.1.1.3

6.1.1.1.2 Spielszene

Die Spielszene dient zur Umsetzung des Spiels und wird von der Hauptmenüszene entweder im Einzelspielermodus oder im Multiplayermodus aufgerufen. Der Modus hängt ab vom übergebenen Spiel-objekt. Wird ein Spiel-Objekt übergeben, dann ist es ein Multiplayer-Spiel und die Scores werden dann in dem Spiel eingetragen. Wenn keines übergeben wird ist es ein Einzelspieler-Spiel.

Die Szene besteht aus folgenden Komponenten:

Kamera

Die Kamera zeigt den Bereich an, der am Bildschirm des Benutzers angezeigt wird. Das soll also immer der Bereich sein, in dem sich der Charakter gerade befindet. Um das umzusetzen wurde ein Skript implementiert, welches die Kamera bei jeder Bewegung des Charakters an dessen Position angleicht.

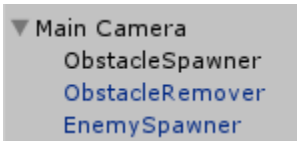


Abbildung 38: Kamera in der Szenenhierarchie

Die Kamera hat dann noch drei Komponenten, welche für die Initialisierung der Hindernisse & Gegner verantwortlich ist. Die zwei „Spawner“, sind vor der Kamera platziert. Sie erstellen also die

Gegner und Hindernisse im nicht sichtbaren Bereich. Erst wenn sich der Spieler weiter bewegt werden sie sichtbar.

Der „Remover“ ist wichtig für die Performance der Anwendung. Er ist nämlich hinter der Kamera positioniert und setzt alle Hindernisse und Gegner, die die in der Kamera nicht mehr sichtbar sind auf inaktiv. Die „Spawner“ sind so implementiert, dass sie die inaktiven Objekte dann wieder auf aktiv setzen und der Position an die Position des „Spawners“ angleichen. Dadurch wird ein sogenanntes „Object-Pooling“ bereitgestellt, welches dafür sorgt, dass nicht laufend neue Objekte erstellt werden müssen, sondern immer nur so viele, wie gerade in der Kamera sichtbar sind. Nur wenn kein inaktives Objekt mehr vorhanden ist wird ein neues instanziiert.

Die „Spawner“ basieren außerdem auf einem Zufallsprinzip. Sie erstellen immer in zufälligen Zeitabständen zufällige mögliche Hindernisse und Gegner.

Dieses Zufallsprinzip wird beim Multiplayermodus genutzt, um durch die Übernahme des Seeds, also der Ausgangszahl, von der die zufälligen Ergebnisse generiert werden, identische Gegebenheiten für beide Spieler zu gewährleisten.

Hier sieht man die Umsetzung der „Spawn“ Methode für Gegner und Hindernisse:

```
void Spawn()
{
     //Wenn es ein Multiplayer-Spiel ist wird der Seed übernommen
     if (Seed != null)
         Random.seed = Seed;
     else
         Random.seed = System.Environment.TickCount;
     //Liste der Objekte durchmischen um zufälliges Objekt zu erhalten.
     for (int i = 0; i < objects.Count; i++)
     {
         GameObject temp = objects [i];
         int randomIndex = Random.Range (i, objects.Count);
         objects [i] = objects [randomIndex];
         objects [randomIndex] = temp;
     }
     //Finden eines inaktiven Game-Objektes im Objekt-Pool
     GameObject obj = objects.FirstOrDefault (x => !x.activeInHierarchy);
     if (obj == null)
     {
         //Initialisierung von jeweils einem Objekt von allen möglichen Objekten,
         //um gleiches Verhältnis von unterschiedlichen Objekten in der Szene zu
         //gewährleisten
         for (int i = 0; i < possibleOjects.GetLength(0); i++){
              GameObject tmp = (GameObject)Instantiate(possibleOjects[i]);
              tmp.SetActive (false);
              objects.Add (tmp);
              obj = tmp;
          }
      }
      //Ändern der Position auf die des Spawners
      obj.transform.position = new Vector3 (gameObject.transform.position.x,
      obj.transform.position.y, 0);
      //Aktivierung des Hindernisses oder Gegners
      obj.SetActive (true);
      //Erneuter Aufruf der Spawn Methode nach einer zufälligen Zeit im Bereich
      //zwischen minimaler und maximaler Spawnzeit
      Invoke ("Spawn", Random.Range(spawnMin, spawnMax));
}
```

Canvas

Dieses beinhaltet wieder die Menüs des Spiels.

Zur Hauptmenüszene ist hier aber der Unterschied, dass das Canvas konfiguriert werden musste, so dass es der Kamera folgt, um das Menü immer am Bildschirm des Benutzers anzuzeigen.

Abbildung 39: Canvas in der Szenenhierarchie

Das Canvas besteht in der Spielszene aus folgenden Komponenten:

- Score-Anzeige, die bei jedem Update des Charakters aktualisiert wird.
- Button zum Springen, der den Charakter in die Springanimation versetzt.
- Pause Button, mit dem das Pause Menü geöffnet wird. Im Pause-Menü wird die Zeit des Spiels auf 0 gesetzt und der Fortschritt somit blockiert. Währenddessen wird der Hintergrund noch gedimmt, um die Pause besser darzustellen.
- Anzeige für die Münzen des Benutzers -Diese werden vom Data-Service in das Spieler-Objekt geladen und beim Abschuss von Gegner aktualisiert.
- Eine Lebensanzeige, die immer, wenn der Charakter mit einem Hindernis kollidiert, um 20% verringert wird. Wenn sie Null erreicht wird im Spiel-Controller die „GameOver“ Methode aufgerufen und das „GameOver“ Menü angezeigt.

Charakter

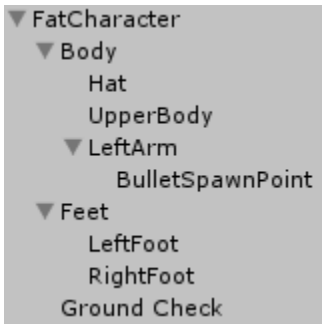


Abbildung 40: Charakter in der Szenenhierarchie

Der Charakter ist das Game-Objekt, um den sich das Spiel dreht. Er besteht aus einzelnen Körperteilen (Füße, Körper, Arm), die animiert worden sind. Für die Animation siehe 6.1.1.2.2

Um mit der Umgebung korrekt zu interagieren benötigt der Charakter sogenannte „Collider“ (siehe Abbildung 41: Collider des Charakters). Diese werden in Formen von Quadraten, Kreisen oder den weniger effizienten Polygonen an die Grafik des Körpers angepasst und ermöglichen so, dass der Charakter auf dem Boden laufen kann bzw. er mit den Hindernissen kollidieren kann. Dazu müssen natürlich der Boden und die Hindernisse auch entsprechende „Collider“ besitzen. Die Hindernisse verursachen durch Events, die bei der Kollision auftreten, den Abzug von Lebenspunkten vom Benutzer.

Abbildung 41: Collider des Charakters

Um die Kugeln der Waffe zu feuern wird eine Startposition benötigt. Um die Position der Waffenspitze immer zu erkennen, auch wenn sich die Rotation des Armes verändert hat wurde ein leeres Game-Objekt als Startposition erstellt, welches immer an der Spitze des Gewehres fixiert ist.

Wenn der Benutzer nun schießen möchte, so wird im Charakter-Controller, dem Skript zur Bereitstellung der Interaktionen mit dem Charakter, die Kugel an der Position erstellt und ein Vektor errechnet, um die Kugel genau in die Richtung des gedrückten Punktes schießen zu lassen. Auch hier wird wieder das zuvor erklärte „Object-Pooling“ verwendet, um die Performance zu verbessern.

Untenstehend sieht man die implementierte Schuss-Funktion des Charakters, die aufgerufen wird, sobald der Benutzer versucht einen Gegner abzuschießen.

```
public void Shoot()
{
     //Zielvektor erhalten
     Vector3 target = new Vector3(Input.mousePosition.x, Input.mousePosition.y, 0);

     //Vektor umwandeln, so dass der Vektor genau von der Kamera ausgeht
     target = Camera.main.ScreenToWorldPoint(target);

     //z-Achse auf Position des Charakters setzen, um Kugel nicht auf z-Achse zu
     //verschießen
     target.Set(target.x, target.y, gameObject.transform.position.z);

     //Herausfinden der Rotation, die benötigt wird um das Gewehr
     //zum Ziel auszurichten
     Quaternion rotation = Quaternion.LookRotation(target - transform.position,
           transform.TransformDirection(Vector3.up));

     //Setzen der Rotation auf das Gewehr
     rifle.transform.rotation = new Quaternion(0, 0, rotation.z, rotation.w);

     //Inaktive Kugeln der Szene finden
     GameObject obj = bullets.FirstOrDefault(x => !x.activeInHierarchy);
     if (obj == null)
     {
         //Kugel erstellen, wenn keine inaktive vorhanden ist
         obj = (GameObject)Instantiate(bullet);
         //Kugel zur Liste von vorhandenen möglichen Kugeln hinzufügen
         bullets.Add(obj);
     }

     //Kugel sichtbar machen
     obj.SetActive(true);

     //Position der Kugel auf die x und y Position der Gewehrspitze setzen
     obj.transform.position = new Vector3(bulletSpawn.transform.position.x,
           bulletSpawn.transform.position.y,0);

     //Rotation der Gewehrspitze übernehmen
     obj.transform.rotation = bulletSpawn.transform.rotation;

     //Normalvektor zum Ziel mit der Geschwindigkeit multiplizieren und als
     //Geschwindigkeit der Kugel setzen
     obj.GetComponent<Rigidbody2D>().velocity = (target -
           bulletSpawn.transform.position).normalized * speed;

     //Schießen für eine Sekunde unterdrücken danach durch Aufruf der
     //"CanShootAgain" Methode wieder aktivieren
     shoot = false;
     Invoke("CanShootAgain", 1f);
}
```

6.1.1.2 Charakter Animation

Die Animation des Charakters besteht zu einem Teil aus dem Animation-Controller und zum anderen Teil aus der Animation selbst.

6.1.1.2.1.1 Animation-Controller

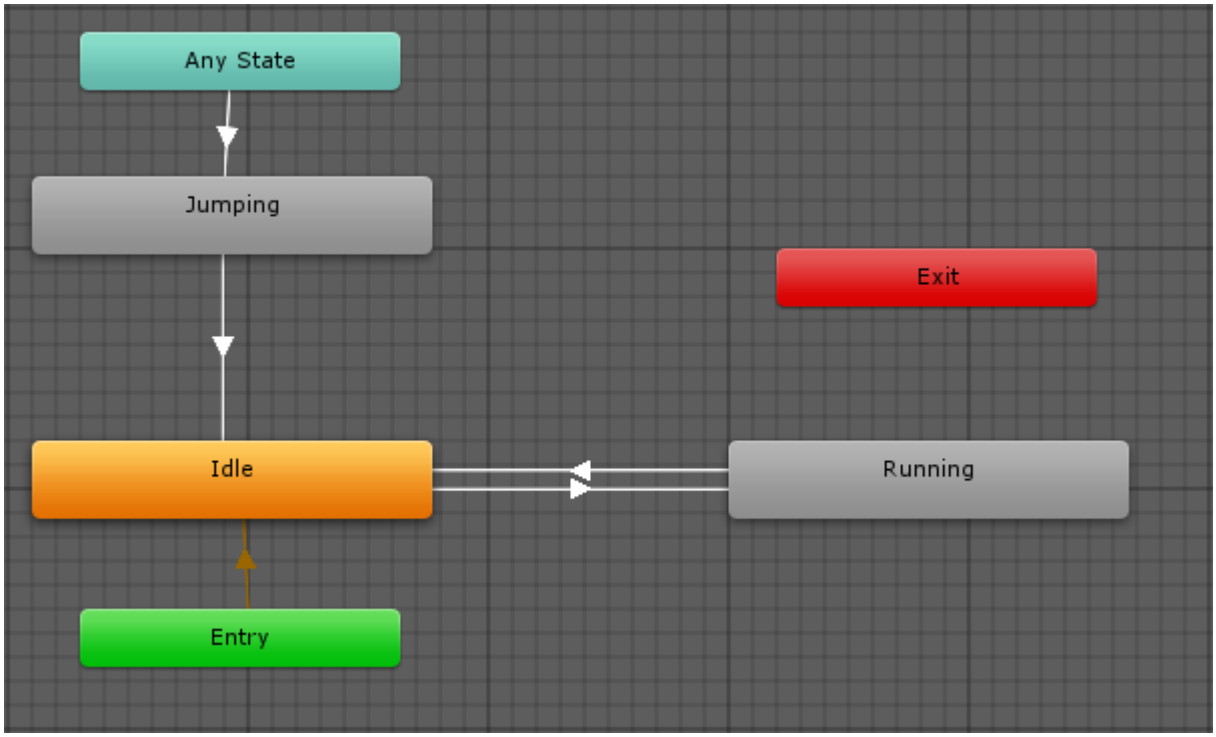


Abbildung 42: Animation-Controller

Der Animation-Controller legt die Status der Animation fest. Diese Status haben dann je eine Animation zugeordnet, die vom Character-Controller bei Benutzerinteraktionen aufgerufen wird.
Um diese Funktionalität zu erhalten muss der Animation-Controller in der Szene dem Charakter als Komponente angehängt werden.

6.1.1.2.2 Animation

In der untenstehenden Grafik sieht man die Kurven der einzelnen Positions- und Rotationsparameter der Lauf-Animation. Es wird hierbei je die Position und Rotation der beiden Füße und die Position des Körpers angepasst. Die Kurven wiederholen sich dann immer wieder. In der Grafik Rechts sieht man, dass die in der Animation angepassten Parameter von den Game-Objekten stammen, die in der Szene dem Charakter als Kindelemente untergeordnet sind.

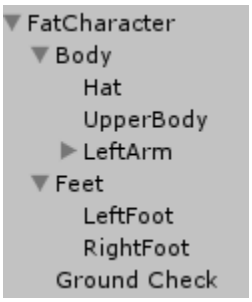


Abbildung 43: Charakter in der Szenenhierarchie

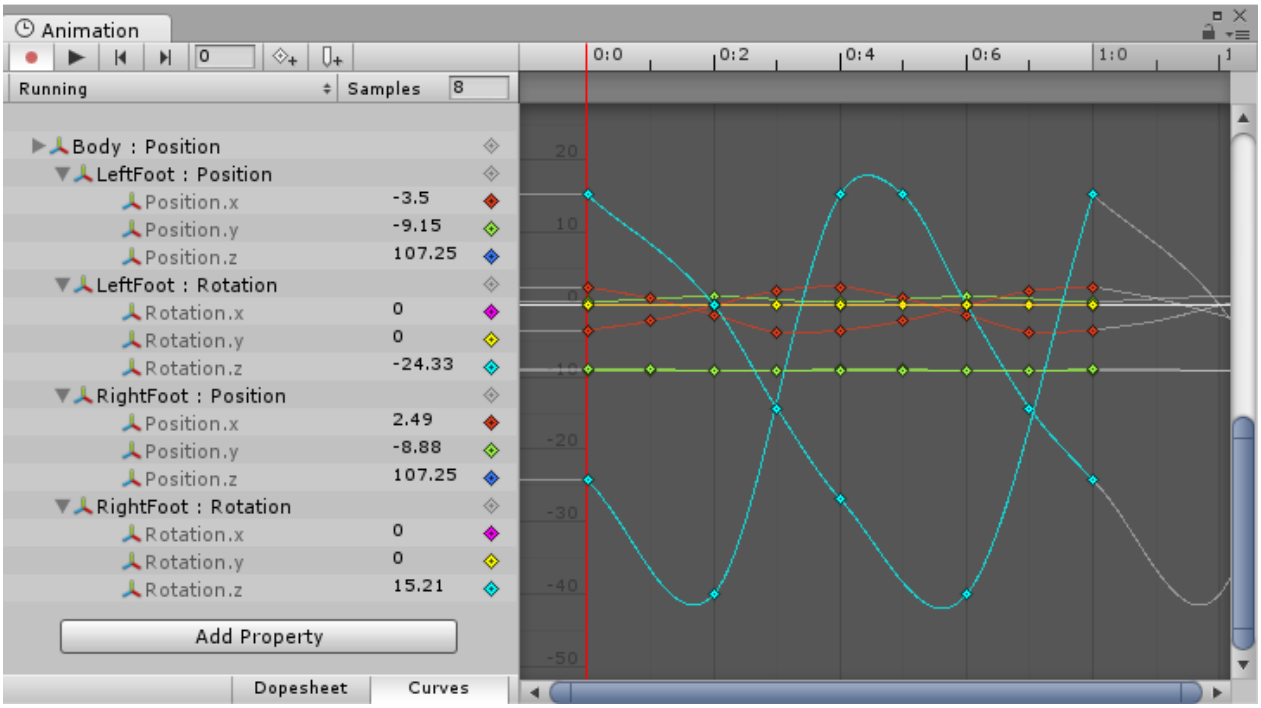


Abbildung 44: Animationskurven

6.1.1.3 Controller

Die Controller sind für die Bereitstellung der Funktionalität der einzelnen Szenen bzw. der Interaktionen mit Schnittstellen implementiert worden.

6.1.1.3.1 Hauptmenü-Controller

Der Hauptmenü-Controller stellt die Funktionalität des Hauptmenüs zur Verfügung.
Er öffnet und schließt Menüs, lädt Daten über den Facebook-Manager vom Facebook API und startet den Facebook-Login über den Login-Controller. Der Hauptmenü-Controller wird in der Hauptmenüszene als leeres Game-Objekt initialisiert und wird nicht in die Spielszene übernommen.

6.1.1.3.2 Login-Controller

Der Login-Controller ruft den Facebook Login des Facebook-Managers auf um den Benutzer bei Facebook zu authentifizieren und ruft anschließend den Login des Data-Service auf um den Facebook-Benutzer am Back-End anzumelden, oder wenn er nicht vorhanden ist, zu registrieren.
Nach Beendigung der Anmeldung wird mittels delegate die View zur Anzeige des Benutzers im Menü benachrichtigt.

6.1.1.3.3 Spiel-Controller

Um den Ablauf des Spiels zu kontrollieren wurde der Spiel-Controller implementiert. Er ist für die Verwaltung des Punktestandes, der Distanz, der Münzen und des Spielers während des Spielablaufs zuständig. Außerdem stellt er die Pause und GameOver Funktionalität zur Verfügung.
Eine weitere wichtige Aufgabe ist das erhöhen der Geschwindigkeit des Spiels, so dass es bei längerer Spieldauer nicht langweilig wird, sondern es immer schwieriger wird den Hindernissen auszuweichen.
Der Spiel-Controller wird bei der Navigation zur Spielszene initialisiert und existiert nur in dieser Szene. Der Spieler wird bei der Initialisierung vom Data-Service angefordert und wenn das Spiel ein Mehrspieler-Spiel ist, so wird auch die bereits vorhandene Herausforderung mit den beiden betroffenen Benutzern geholt.

6.1.1.3.4 Spielmenü-Controller

Um die Funktionalität der Menüs der Spielszene bereitzustellen wurde der Spielmenü-Controller implementiert. Er zeigt bei Pause & GameOver die entsprechenden Menüs an und füllt sie mit den Daten, die der Spiel-Controller beim Aufruf der Funktionalitäten des Spielmenü-Controllers übergibt.

6.1.1.3.5 Character-Controller

Um die Bewegungen, Animationen und Aktionen des Charakters zu kontrollieren wurde der Character-Controller erstellt.
Das Skript ist als Komponente an den Charakter in der Szene angehängt. Der Controller reagiert dann in der Update Funktion auf die Interaktionen des Benutzers zum Springen oder Schießen. Es werden dann die entsprechenden Animationen über den Animation-Controller gestartet bzw. beim Springen die Kraft für den Sprung auf den Charakter angewendet und beim Schießen die Kugeln initialisiert und abgeschossen.

6.1.1.3.6 Data-Service

Um Serveranfragen erledigen zu können, wurde der Data-Service entwickelt.

Der Spieler, sobald er ein Spiel im Singleplayer- oder auch im Mutiplayer-Modus gespielt hat, greift dieser über diesen auf den Server zu. Dieser validiert dabei die Anfrage, und schickt dem Client eine Antwort, die ebenfalls im Data-Service behandelt wird.
Generell kann man sagen, dass jede Client-Server-Anfrage über diesen Data-Service erledigt wird.

6.1.1.4 Parallaxing

Um eine Illusion der Tiefe in einem 2D Spiel zu schaffen, wurde der Parallaxe-Effekt verwendet.

6.1.1.4.1 Definition

Die Definition der Bewegungsparallaxe lautet, laut Wikipedia:
Unter Bewegungsparallaxe versteht man in der Wahrnehmungspsychologie den Effekt, der sich optisch ergibt, wenn verschiedene Objekte unterschiedlich voneinander entfernt in einer Landschaft verteilt sind und sich der Beobachter parallel zu diesen Objekten seitlich fortbewegt und dabei in Richtung Horizont blickt. [WIKIPEDIA.ORG]

6.1.1.4.1.1 Anwendungsfall und Implementierung

Der Anwendungsfall der Parallaxe ergibt sich in der *Spielszene* des 2D Spieles. Die Illusion, die damit geschaffen wird, lässt es so aussehen, als würde sich der Hintergrund langsamer bewegen als der Spieler selbst.

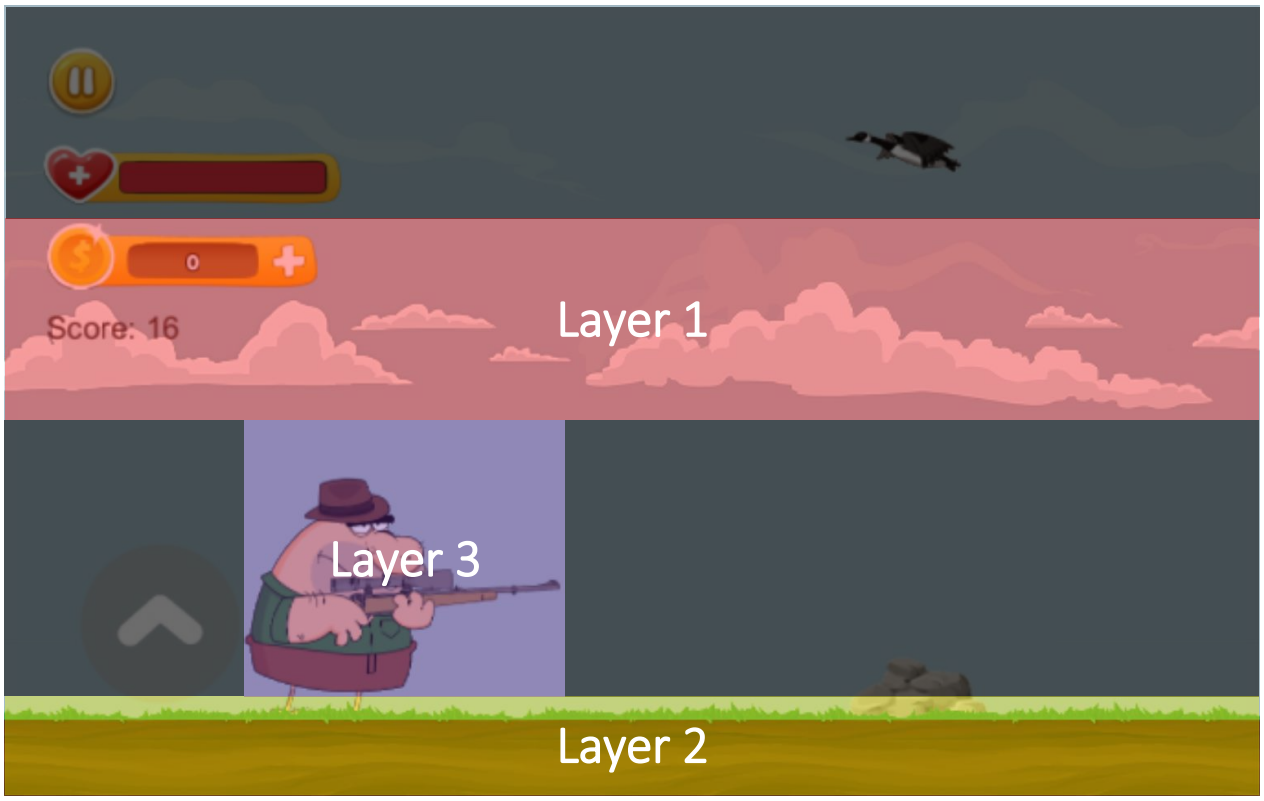


Abbildung 45: Schematische Layer-Unterteilung

Um dies anhand des entwickelten 2D Spieles zu erklären, ist es am leichtesten, wenn man sich die Spielszene in mehreren Schichten, den so sogenannten Layern vorstellt.
Wie oben bereits erwähnt, scheint es dem Spieler so, als würde konkret *Layer 1*, die Wolken langsamer vorbeiziehen. Gleichzeitig scheint es dem Spieler so, als würde Layer 2, der Boden und die Hindernisse, schneller auf diesen zukommen. Der Spieler selbst, Layer 3, wird mit seiner konstanten Geschwindigkeit weiterbewegt.

Um die konkrete Planung dieses Effektes auch in Unity umsetzen zu können, wurde das GameObject ParallaxManager entwickelt. Diesem Objekt werden jeweils von außen die Schichten, also die Layer übergeben. Dieser rechnet sich anhand der Z-Position der Objekte die „Geschwindigkeit" aus, mit der die Objekte bewegt werden sollen.
Am unten angefügten Bild ist zu erkennen, dass für die Backgrounds, also die Layer ein Array zur Verfügung gestellt wird, welches die zu bewegenden Elemente beinhaltet.

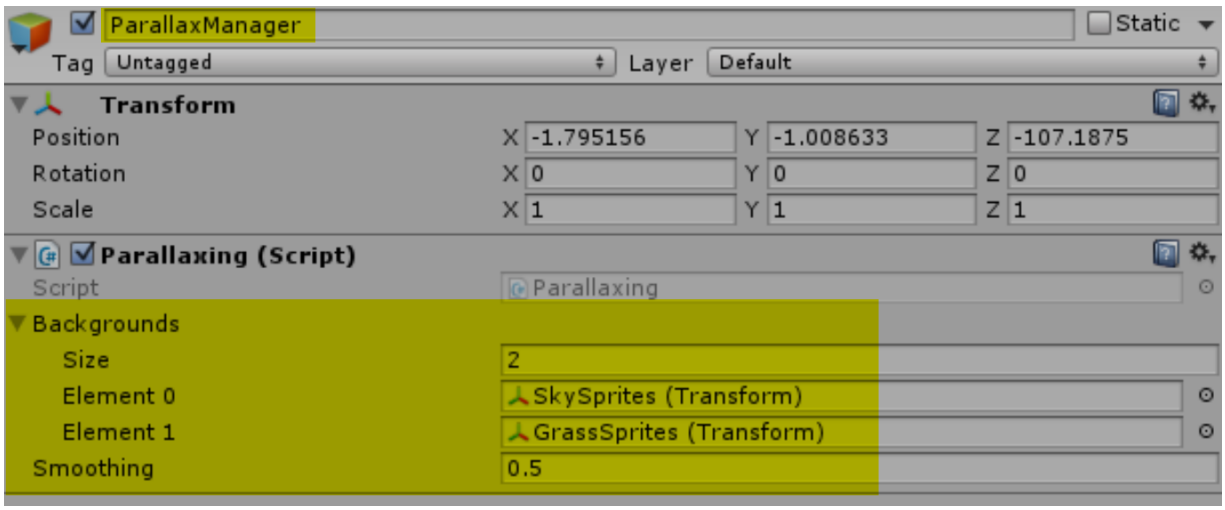


Abbildung 46: GameObject ParallaxManager

So haben zum Beispiel *Layer 1,* die Wolken den Z-Wert von 10, im Gegensatz zum Spieler, *Layer 3,* der den Z-Wert auf 0 gesetzt hat.

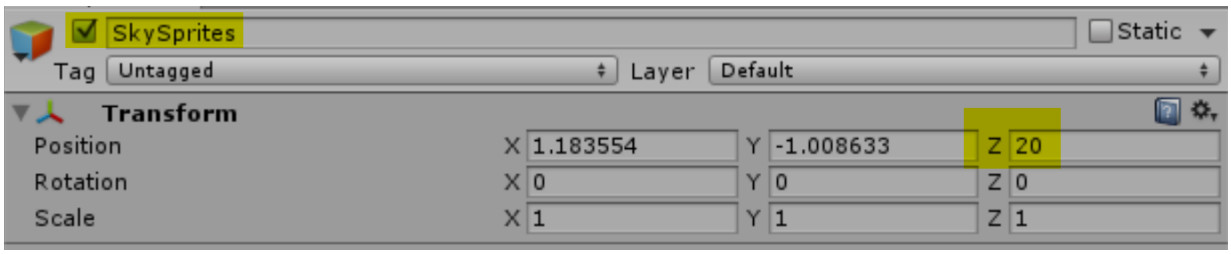


Abbildung 47: Wolken Transform

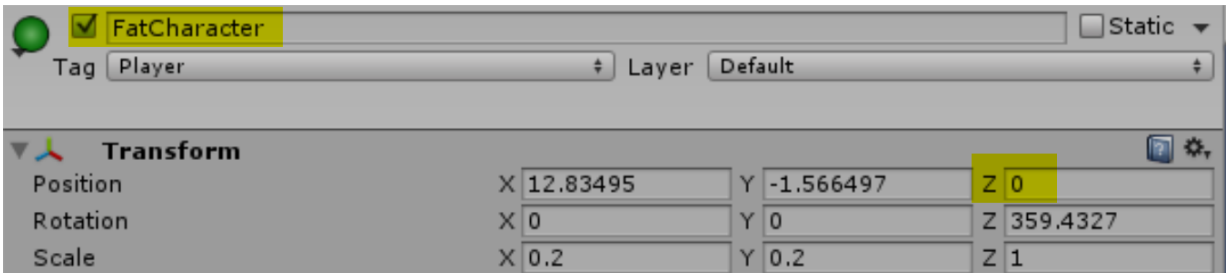


Abbildung 48: Charakter-Transform

Die Implementierung des Parallax erfolgt in der Klasse Parallaxing. Die Hauptumstellung der Elemente erfolgt in der Methode Update(), die periodisch aufgerufen wird.

```
public class Parallaxing : MonoBehaviour {

      public Transform[] backgrounds;   // Liste der zu bewegenden Objekte
      private float[] parallaxScales;   // Die Proportionen der Kamera
      public float smoothing = 1f;      // Einstellung, wie fein der Parallax sein
                                           sollte; Wert > 0 !

      private Transform cam;            // Referenz auf Transform der Kamera
      private Vector3 previousCamPos;   // Position der Kamera im letzten Frame

      void Awake () {
         cam = Camera.main.transform;
      }

      void Start () {
         previousCamPos = cam.position;

         parallaxScales = new float[backgrounds.Length];
         for (int i = 0; i < backgrounds.Length; i++) {
            parallaxScales[i] = backgrounds[i].position.z*-1;
         }
      }

      void Update () {
         //Iteration durch alle übergebenen Layer
         for (int i = 0; i < backgrounds.Length; i++) {
            //Berechnung des Parallaxing
            float parallax = (previousCamPos.x - cam.position.x) * parallaxScales[i];
            float backgroundTargetPosX = backgrounds[i].position.x + parallax;

            //Vektor, der die Position des Layers hat, mit der Ziel- X Position
            Vector3 backgroundTargetPos = new Vector3 (backgroundTargetPosX, back
                  grounds[i].position.y, backgrounds[i].position.z);

            //Zwischen den zwei Positionen mit Lerp wechseln
            backgrounds[i].position = Vector3.Lerp (backgrounds[i].position, back
                  groundTargetPos, smoothing * Time.deltaTime);
             }

            //Die Kameraposition am Ende wieder neu setzen
            previousCamPos = cam.position;
      }
}
```

6.1.2 Facebook-Anbindung

Zur Bereitstellung der Funktionalitäten von sozialen Netzwerken wird Facebook verwendet.
In diesem Kapitel geht es um die Hintergründe und die Umsetzung der Anbindung in der Anwendung.

6.1.2.1 Unity Facebook SDK

Um das Facebook SDK mit dem Unity Projekt zu verwenden, muss das Facebook SDK für Unity in das Projekt eingebunden werden. Das funktioniert über ein von Facebook bereitgestelltes Paket („unitypackage“) welches über den Unity Editor importiert wird und so die Einstellungen bzw. Klassen des Facebook SDKs verfügbar macht.

Damit dieses SDK auch läuft, muss über Facebook die App registriert werden. Dies findet auf der Entwickler Plattform von Facebook statt. Die App erhält nun eine ID und kann nach Festlegung des Packages im Unity Editor und nach Generierung des Key-Hashes mittels OpenSsl, auf der Ziel Plattform ausgeführt werden.

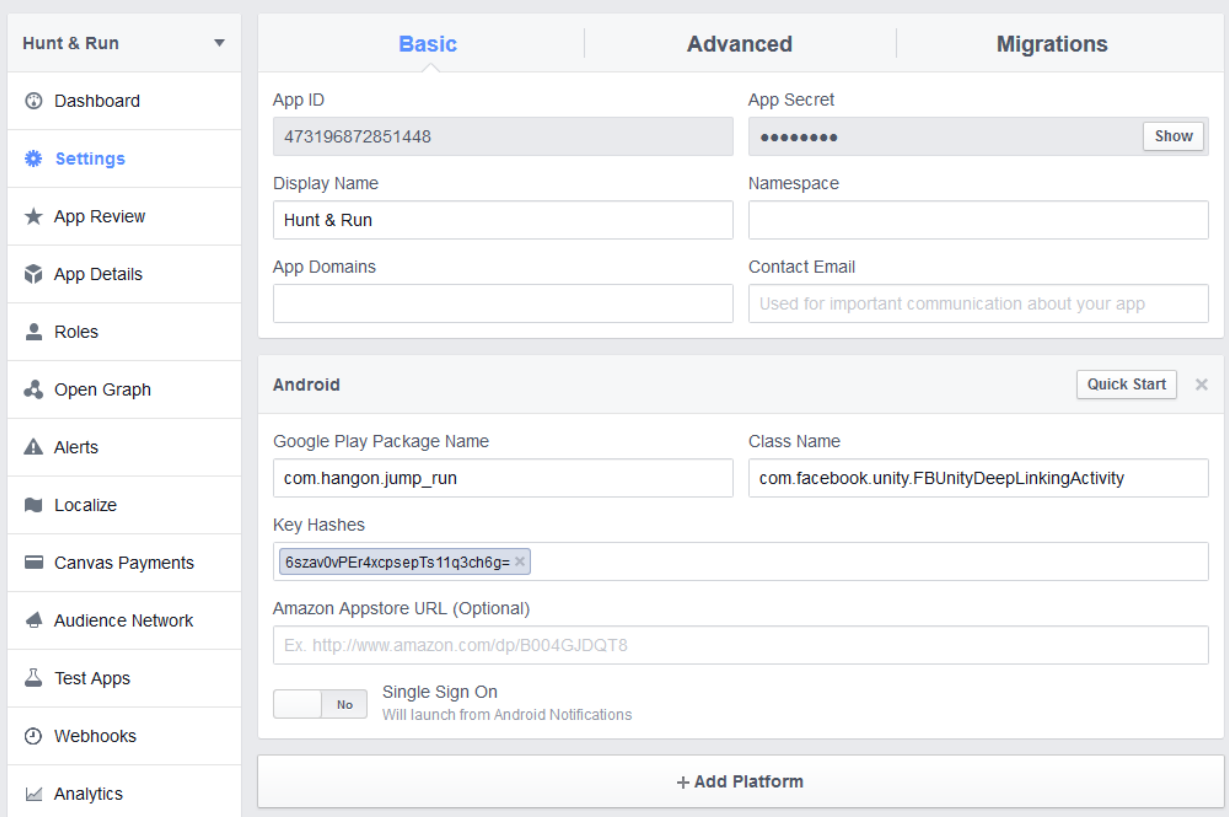


Abbildung 49: Konfiguration der App in Facebook

6.1.2.2 Verwendete Funktionalität des SDKs

Im Folgenden werden die verwendeten Funktionalitäten des SDKs, deren Hintergründe und deren Implementierung beschrieben.

6.1.2.2.1 Login

Die Login Funktionalität wird vom SDK bereitgestellt und kann aufgerufen werden, sobald die Facebook Klasse initialisiert ist. Während der Login durchgeführt wird, wird die Unity Ausführung gestoppt und nach Beendigung des Prozesses mittels Callback der Login Funktion wieder aktiviert. Nach erfolgreichem Login ist der Benutzer mittels seiner eindeutigen Facebook User ID identifiziert und es kann nun auf seine Daten zugegriffen werden nachdem er die entsprechenden Berechtigungen an unsere Applikation übergeben hat.

Nachfolgende Berechtigungen werden direkt erhalten, wenn sich der Benutzer mit der App authentifiziert:

email Bietet Zugang zur primären Emailadresse des Benutzers.
public_profile Bietet Zugang zu den öffentlichen Basisinformationen eines Benutzers (Name, Profilbild, Geschlecht, Alter)
user_friends Bietet Zugang zu der Liste von befreundeten Person des Benutzers, die ebenso die App verwenden.

Tabelle 11: Standardmäßige Login Berechtigungen

Die User ID wird außerdem auf der Datenbank hinterlegt, um die Spieldaten des Benutzers zu verwalten

6.1.2.2.2 Invitations

Um Invitations, also Einladungen zum Spiel in der App umzusetzen, muss ein AppRequest, der vom Facebook SDK implementiert wird, aufgerufen werden.
Dieser AppRequest hat folgende Parameter:

```
public static void AppRequest(
    string message,
    OGActionType actionType,
    string objectId,
    IEnumerable<string> to,
    string data = "",
    string title = "",
    FacebookDelegate<IAppRequestResult> callback = null
)
```

Die Parameter, die benötigt werden, um einen Auswahldialog für die App-Einladung anzuzeigen sind:

- **message:** Nachricht, die dem Empfänger bei der Einladung angezeigt wird
- **to:** Muss *null* sein, um einen Auswahldialog anzuzeigen
- **title:** Titel des Auswahlfensters, welches dem Benutzer erscheint
- **callback:** Delegate zur Methode, die bei Abschluss der Einladung ausgeführt werden soll

6.1.2.2.3 Graph-API

Wie bereits erwähnt dient die Graph-API zum Austausch von Daten mit der Facebook Plattform. Die Informationen im Facebook Graph werden wie folgt strukturiert:

- **Nodes**
  - Objekte wie z.B.: Benutzer, Foto, Seite, Kommentar
- **Edges**
  - Verbindung zwischen den Objekten wie z.B.: Beiträge zu einer Seite, Kommentare zu einem Foto
- **Fields**
  - Informationen über die Nodes. Z.B.: Name einer Person, Name einer Seite

Mit folgenden http-Befehlen können die Informationen abgefragt werden:

Abfrage einer Node mithilfe der eindeutigen ID (z.B.: Benutzer durch Benutzer-ID):

```
GET graph.facebook.com
  /{node-id}
```

Abbildung 50: Graph-API – GET-Befehl einer Node [FACEBOOK]

Abfrage einer Edge mithilfe der eindeutigen ID der verknüpften Node (z.B.: Kommentare zu einem Foto):

```
GET graph.facebook.com
  /{node-id}/{edge-name}
```

Abbildung 51: Graph-API -GET-Befehl einer Edge [FACEBOOK]

Die POST bzw. DELETE Befehle funktionieren nach gleichem Aufbau.

6.1.2.3 Facebook Manager

Um die Aufrufe des Facebook SDKs zu bündeln wurde die Facebook Manager Klasse implementiert. Diese Klasse ist ein Singleton. Das heißt, es existiert nur ein Objekt dieser Klasse. Dieses Objekt wird beim Start der Applikation erzeugt.

Folgende Funktionalitäten wurden implementiert:

- Initialisierung der Facebook Klasse
- Facebook Login und Login am Backend nach erfolgreichem Facebook Login.
- Anforderung von Profilname und Profilbild des Benutzers via Graph-API zur Anzeige des Profils auf der Startseite.
- Erfassen aller Freunde, welche die App benutzen, und Übergabe der Liste an den Menü-Controller zum Aufbau der Liste, so dass diese anschließend zu einem Spiel herausgefordert werden können.
- Umsetzung eines sogenannten AppRequests zur Anzeige der Auswahlliste um App-Einladungen an Freunde zu versenden.

6.2 Back-End

Das Back-End besteht aus dem Server, auf dem der WildFly Service zum Einsatz kommt, sowie der MySQL Datenbank. In diesem Kapitel wird nun die Implementierung dieser beiden Technologien erläutert.

6.2.1 Server

Dieses Unterkapitel behandelt die konkrete Umsetzung der Serverkonfiguration. Es wird die gesamte Implementierung des WildFly-Projektes erklärt, unter anderem auch die Anfragen, die der Client dem Server senden kann.

6.2.1.1 Windows-Server

Um den Clients ein vom Internet aus erreichbares Web-Service zur Verfügung stellen zu können, ist im September 2015 ein Windows-Server beantragt worden. Bei diesem musste der http-Port 80 freigeschaltet werden, damit die Clients den Server vom Internet aus erreichen können.

Außerdem wurde bei der Serveranfrage auch ein DNS-Eintrag erwünscht, weshalb unser Projektserver unter *huntandrun.htl-perg.ac.at* erreichbar ist. Die Clients haben dabei genau diese URL als Server-Eintrag gespeichert.

6.2.1.2 WildFly-Projekt

Das WildFly-Projekt wurde anfangs am Privatrechner entwickelt-am Ende wurde es, ähnlich zum Datenbankentwurf, auf den Windows-Server portiert.

Der Aufbau des Projektes, welches mithilfe der Entwicklungsumgebung Eclipse realisiert wurde, sieht folgendermaßen aus:

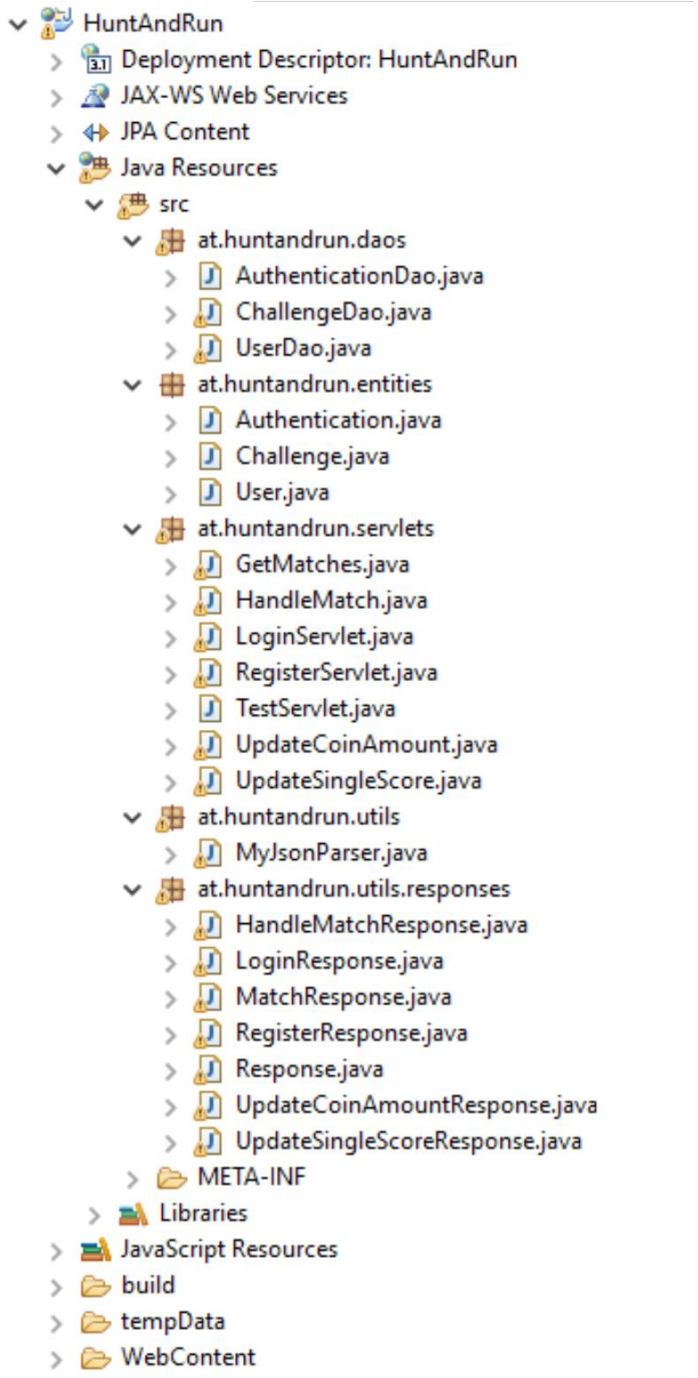


Abbildung 52: Aufbau des WildFly-Projektes

Bei der Entwicklung wurde Wert darauf gelegt, die Strukturierung des Projektes so übersichtlich, aber auch einfach zu halten. Mit den Packages *.daos, *.entities, *.servlets, *.utils sowie *.utils.responses wurde eine klare Verschachtelung der Lösung erstellt.

6.2.1.2.1 Konfiguration

Die Konfiguration des Servers wurde in der *WildFly Management Console* erledigt. Diese ist standardmäßig, solang man die Konfiguration nicht verändert, über die Serveradresse und den Port *9990* erreichbar.
Beim Aufruf dieser *Console* muss man sich mit dem *ManagementRealm* Benutzer anmelden, welchen man bereits vor der Konfiguration erstellt haben soll.

Damit der Server einen Zugriff auf die MySQL-Datenbank erhält, muss man unter dem Konfigurationspunkt *„Configuration: Subsystems -> Subsystem: Datasources -> Type: Non-XA“* einen Verweis auf unsere Datenbank erstellen. Der Verweis, der auch *Datasource* genannt wird, sorgt dafür, dass der Server eine Verbindung zur Datenbank erstellen kann und uns somit den Zugriff auf die gespeicherten Daten garantiert.

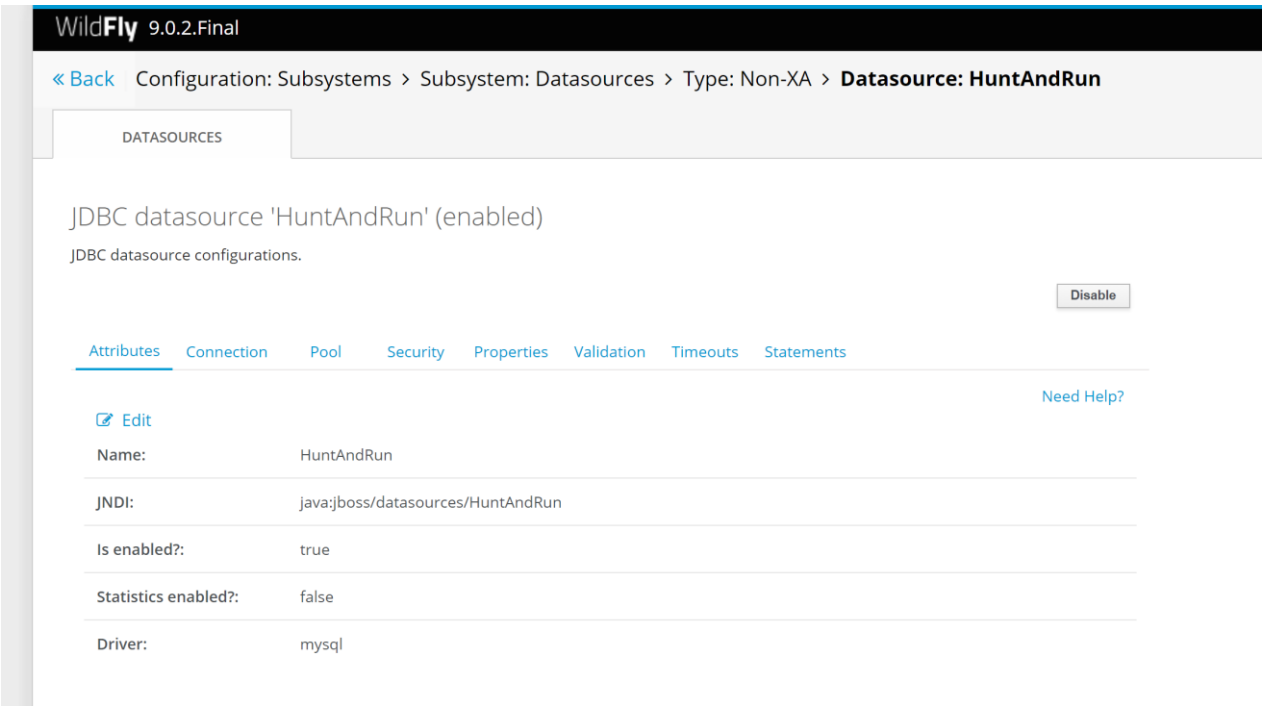


Abbildung 53: Konfiguration der Datasource

Die erstellte Datasource muss auch in dem jeweiligen WildFly-Projekt als Datasource in der Datei *persistence.xml* eingetragen werden, da ansonsten das der Server die Quelle der Daten im Projekt nicht kennt. Nach diesem Schritt sollte es bereits möglich sein, Datenbankabfragen mithilfe der Serverlösung zu tätigen.
Mithilfe des ORM (Object/Relational Mapping) ist es nun auch möglich, diese Datensätze in Objekte überzuführen.

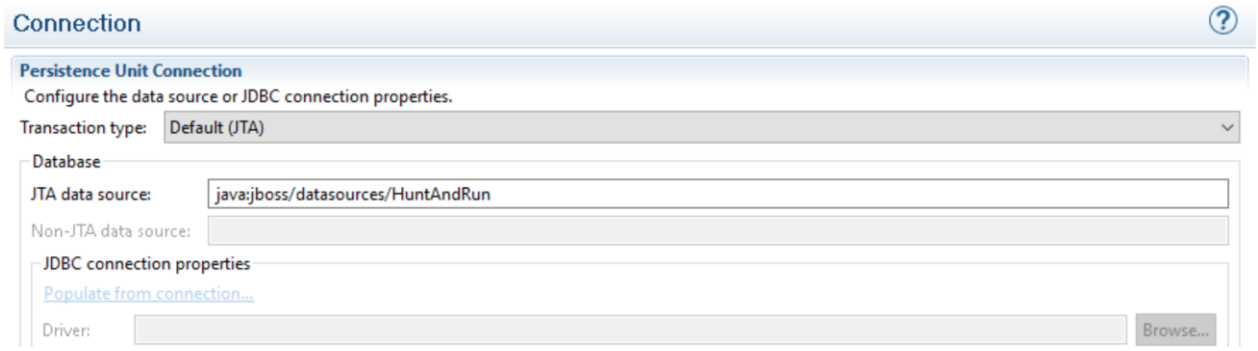


Abbildung 54: Eintrag der Datasource im persistence.xml-File

In der *Management Console* kann man sich aber auch viele, sehr praktische Informationen ansehen. So kann man sich zum Beispiel sehr einfach die derzeitige Serverauslastung ansehen, ohne die Log-Dateien des Servers bzw. den Task-Manager des Servers aufzumachen.

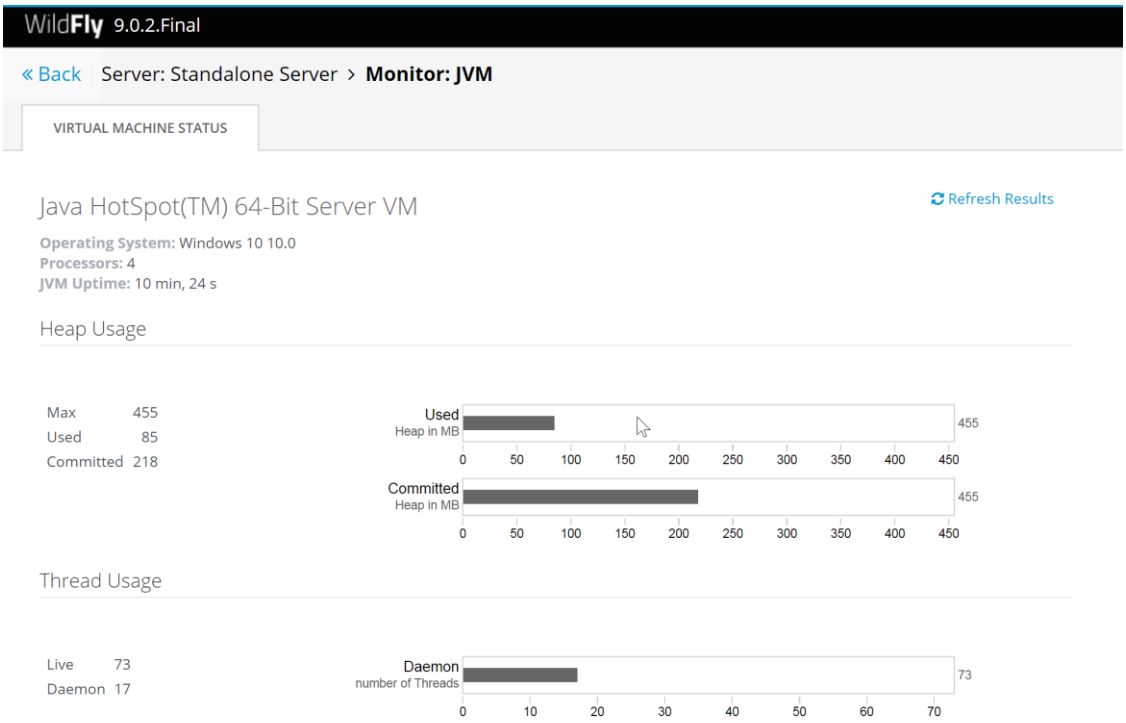


Abbildung 55: Serverauslastung

Grundsätzlich sollte man, wie bei jeder Architektur, darauf achten, dass der Server nicht überlastet wird. Natürlich wird die Geschwindigkeit des Servers gedrosselt, wenn mehrere Clients gleichzeitig auf diesen zugreifen, jedoch kann man mit einfachen Mitteln diese Geschwindigkeitsdrosselungen beheben:

- Keine Anweisungen im Code verwenden, die den Hauptthread sperren
- Keine offenen Datenbankconnections halten
- Heap-Speicherkapazität beachten
    - Keine großen Datenmengen dauerhaft im RAM abspeichern
    - Speicher freigeben

6.2.1.2.2 Client-Anfragen an den Server

Grundsätzlich arbeitet der Server jede Anfrage eines Clients auf dieselbe Weise ab.
Der Benutzer bzw. Client sendet eine Anfrage (z.B. Highscore ändern) an den Server, dieser prüft ob der Benutzer ein „echter Benutzer“ ist, oder sich nur als dieser ausgibt. Falls der Benutzer authentisiert wird, bekommt der Client eine erfolgreiche Nachricht zurückgeschickt.

Die Abarbeitung erfolgt in den sogenannten Servlets. Der Client kann diese aufrufen, indem er eine http-Verbindung zu dem jeweiligen Servlet unter der Serveradresse aufruft. Die Daten, die der Benutzer bei der Anfrage mitsendet, hängen von der jeweiligen Anfrage an das Servlet ab. Jedoch wird bei jeder Anfrage, wie bereits oben erwähnt, die Authentisierung des Benutzers mitgesendet.
Der Zugriff auf die Datenbank in den Servlets erfolgt über sogenannte DAOs (Data Access Objects). Diese beinhalten Methoden, die den Datenzugriff auf bestimmte Elemente der Datenbank erlauben.

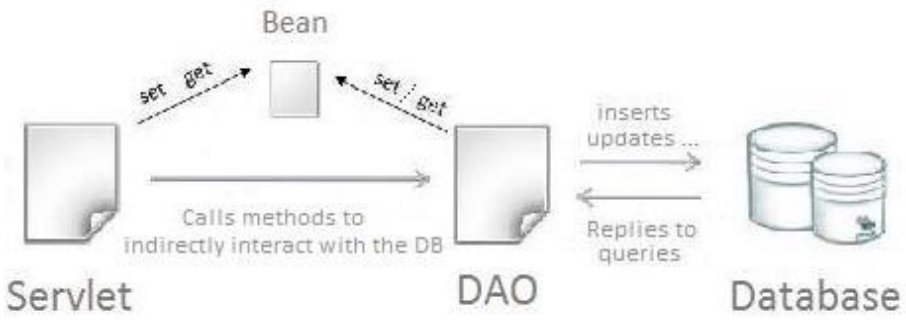


Abbildung 56: schematischer Aufbau Servlet, Dao, Database [MET.GUC.EDU.EG]

Eine beispielhafte Implementierung des Data Access Objects sieht folgendermaßen aus:

```
@Named
@Stateless
public class UserDao {
	@PersistenceContext
	public EntityManager manager;

	public List<User> findAll(){…}

	public User findByFacebookId(String id){…}

	public User findByUniqueId(int id) {…}

	public void save(User user) {…}
	public void updateLastLogin(User user) {…}

	public void updateHighScore(User user, int score) {…}

	public void addAmountOfCoins(User user, int coinsToAdd) {…}
}
```

Abbildung 57: Implementierung Aufbau Dao

Jede Tabelle unseres Datenmodells hat dabei ein eigenes DAO *(AuthenticationDao, ChallengeDao, UserDao),* welches dafür sorgt, dass der Datenzugriff über einfache Methodenaufrufe erfolgen kann.

Die DAOs erleichtern den Umgang mit Daten, weil man sich nicht direkt im Servlet mit der konkreten Datenspeicherung sowie -beschaffung auseinandersetzen muss, sondern einfach, wie bereits oben erwähnt, die Methoden des Data Access Objects im Servlet aufrufen kann.

```
public User findByFacebookId(String id){
	TypedQuery<User> query = manager.createQuery
		("SELECT c FROM User c WHERE c.identNo = :id", User.class);

	query.setParameter("id", id);

	User returnUser = null;
	try{
	    returnUser = query.getSingleResult();
	}catch(NoResultException ex){
	    System.out.println("User-Request with false ID!");
	}

	return returnUser;
}
```

Abbildung 58: Implementierung einer Dao-Methode

Eine beispielhafte Server-Anfrage sieht folgendermaßen aus:

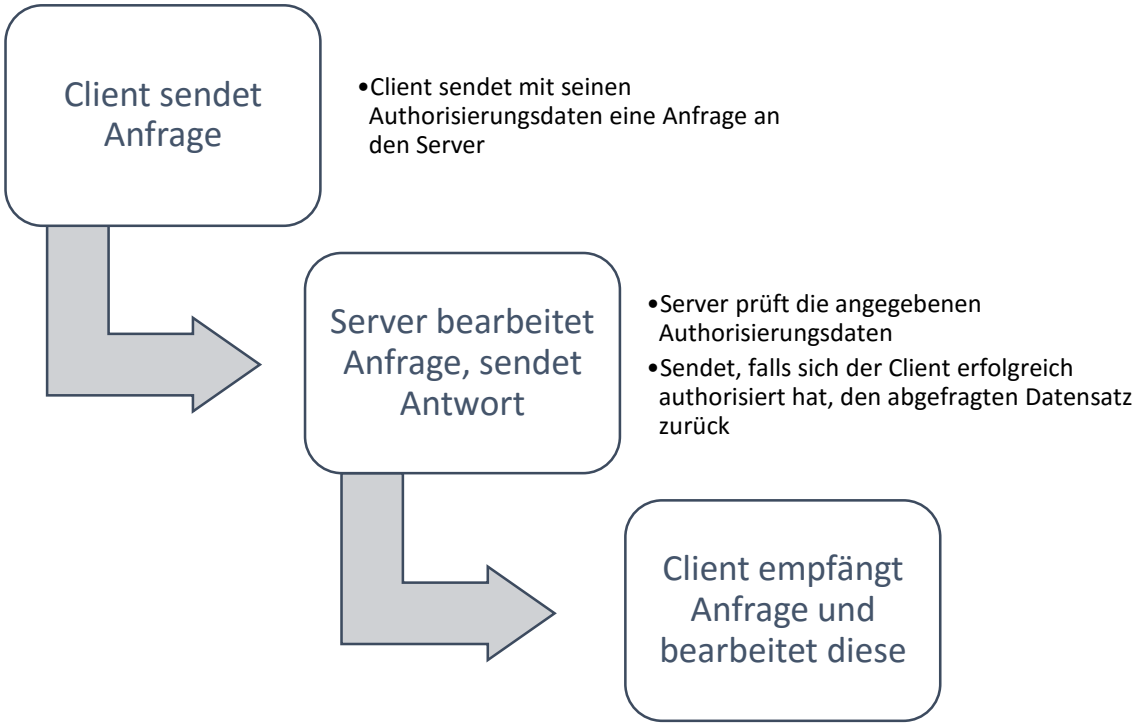


Abbildung 59: Server-Anfrage vom Client

In unserem Fall sind die wichtigsten Client-Anfragen

- Login-sowie Register,
- Update Score sowie Coins und
- Update Match sowie Handle Match.

Diese werden in den Unterpunkten dieses Kapitels nochmals genauer mit Codesequenzen erklärt.

6.2.1.2.2.1 Login und Register

Beim Login-Versuch des Benutzers wird das Servlet *LoginServlet* aufgerufen. Der Benutzer authentifiziert sich, indem er einen POST-Request an den Server absetzt. Falls beim Login kein Benutzer gefunden wird, erfolgt eine automatische Weiterleitung an das *RegisterServlet*, welches dafür sorgt, dass der Benutzer in die Datenbank geschrieben wird.

Bei jeder Kommunikation mit dem Server wird der Aufruf von der *doPost*-Methode verarbeitet.

```
protected void doPost(…) {
      {…}

      //Umwandeln des mitegegebenen Json- Strings in ein Json- Objekt
      MyJsonParser parser = new MyJsonParser(json);
      JSONObject jsonObject = parser.getJsonObject();

      //Parsen der mitgegebenen Authentication
      String userId = (String) jsonObject.get("userid");
      String auth = (String) jsonObject.get("auth");

      //Findes des Benutzers, falls dieser schon existiert
      User user = userDao.findByFacebookId(userId);

      //Login- Response vorbereiten
      LoginResponse loginResponse = new LoginResponse();
      if(user==null){
          //Falls kein Nutzer gefunden wurde, Reponse so konstruieren dass
            Benutzer zum Registrierungsformular gelangt
          System.out.println("User not found :( !");
          loginResponse.setStatus(Response.STATUS_FAILED);
          loginResponse.setAllowedToCreate(LoginResponse.STATUS_YES);
      }else{
         //Falls bereits ein Nutzer gefunden wurde
         System.out.println("User found :) !");

         //Spielerdaten updaten, Response so konstruieren dass Antwort
           Spielerdaten zurückgibt
         loginResponse.setStatus(Response.STATUS_OK);
         loginResponse.setLastLoginOn(user.getLastLogin());
         loginResponse.setPlayerId(user.getIdUser());
         loginResponse.setHighScore(user.getHighscore());
         loginResponse.setCoins(user.getCoins());

         userDao.updateLastLogin(user);

      }

      //Das Ergebnis der Login- Response als Json- String zurückgeben
      response.getWriter().append(loginResponse.getJsonString());
}
```

Abbildung 60: LoginServlet Implementierung

Das Praktische an dem implementierten Login ist, dass man aus der Sicht der Administratoren immer sagen kann, welche Spieler sich zur welcher Zeit angemeldet haben. Somit kann auch eine Statistik ermittelt werden, in welcher Zeitspanne sich die meisten Spieler im Spiel befinden.

6.2.1.2.2.2 Update Score und Update Coins

Auch beim *UpdateScoreServlet* wird, ähnlich zum Login bzw. Register Anfangs geprüft, ob überhaupt ein Benutzer mit der mitgegebenen *userId* existiert, um Inkonsistenzen der Datenbank zu verhindern. Nachdem man allerdings davon ausgeht, dass sich ein authentifizierter Benutzer sowieso in der Datenbank befindet, wird dieser Fall üblicherweise nicht eintreten.
Nach der erfolgreichen Authentifizierung des Benutzers wird der High Score bzw. die Coin-Anzahl des Spielers aktualisiert.

```
protected void doPost(…) {
    {…}

    //Neuen Highscore aus json- String parsen
    Long newScore = (Long) jsonObject.get("new_score");

    //Benutzer finden
    User user = userDao.findByUniqueId(userId.intValue());

    //Response vorbereiten
    UpdateSingleScoreResponse updateScoreResponse =
                            new UpdateSingleScoreResponse();
    if(user==null){
         //Wenn kein Spieler gefunden wird
         System.out.println("Score not updated because no player found!");
         updateScoreResponse.setStatus(Response.STATUS_FAILED);
         updateScoreResponse.setSuccessfullyUpdated(
              UpdateSingleScoreResponse.STATUS_NO);
    }else{
         //Wenn der Spieler gefunden wird und der Score geupdated werden
           kann
         System.out.println("Score successfully updated!");
         updateScoreResponse.setStatus(Response.STATUS_OK);
         updateScoreResponse.setSuccessfullyUpdated(
              UpdateSingleScoreResponse.STATUS_YES);

         //Spielerscore updaten
         userDao.updateHighScore(user, newScore.intValue());

    }

    //Die Response dem Spieler als Json- String ausgeben
    response.getWriter().append(updateScoreResponse.getJsonString());
}
```

Abbildung 61: UpdateCoins Implementierung

6.2.1.2.2.3 Update Match und Handle Match

Das *HandleMatchServlet* kümmert sich in dieser Lösung darum, dass sich die Spieler gegenseitig herausfordern können-und aber auch die Herausforderungen annehmen können.

```
protected void doPost(…) {
    //Die zum Match dazugehörigen Informationen aus dem json- String parsen
    JSONObject match = (…);
    Long challengerId = (…);
    String challengedFacebookId = (…);
    Long challengerScore = (…);
    Long challengedScore = (…);
    Long matchId = (…);

    //Challenger sowie Challenged Spieler aus der Datenbank laden
    User challenger = userDao.findByUniqueId(….);
    User challenged = userDao.findByFacebookId(…);
    //Response vorbereiten
    HandleMatchResponse handleMatchResponse =
                                        new HandleMatchResponse();
    //Wenn einer der beiden Spieler sich nicht in der Datenbank befindet
    if(challenger==null || challenged==null){
      handleMatchResponse.setStatus(Response.STATUS_FAILED);
    }else{
        //Diverse Prüfungen, um es möglich zu machen, mittels diesem
          Servlet auch auf Herausforderungen zu reagieren, anstatt nur
          herausfordern zu können
        if(matchId == null || matchId == 0 || matchId.intValue()==0){
            Challenge challenge = new Challenge();
            challenge.setChallengerScore(challengerScore.intValue());
            challenge.setChallengedScore(challengedScore.intValue());
            challenge.setUser1(challenger);
            challenge.setUser2(challenged);

            challengeDao.save(challenge);

            handleMatchResponse.setStatus(Response.STATUS_OK);
        }else{
            Challenge challenge = challengeDao.
                                findChallengeById(matchId.intValue());

            challenge.setChallengerScore(challengerScore.intValue());
            challenge.setChallengedScore(challengedScore.intValue());

            challengeDao.update(challenge);

            handleMatchResponse.setStatus(Response.STATUS_OK);
        }
    }
    //Die Antwort als json- String zurückgeben
    response.getWriter().append(handleMatchResponse.getJsonString());
}
```

Abbildung 62: UpdateMatch Implementierung

6.2.2 MySQL-Datenbank

Dieses Unterkapitel behandelt den Modellentwurf sowie die konkrete Umsetzung dieses Modells auf eine MySQL-Datenbank.

6.2.2.1 Modellentwurf

Um eine funktionierende Datenbank betreiben können, wird ein gut durchdachtes Datenmodell benötigt-dieses wurde mithilfe des Modellierungswerkzeuges *MySQL Workbench* auf dem eigenen Entwicklungsrechner erstellt.

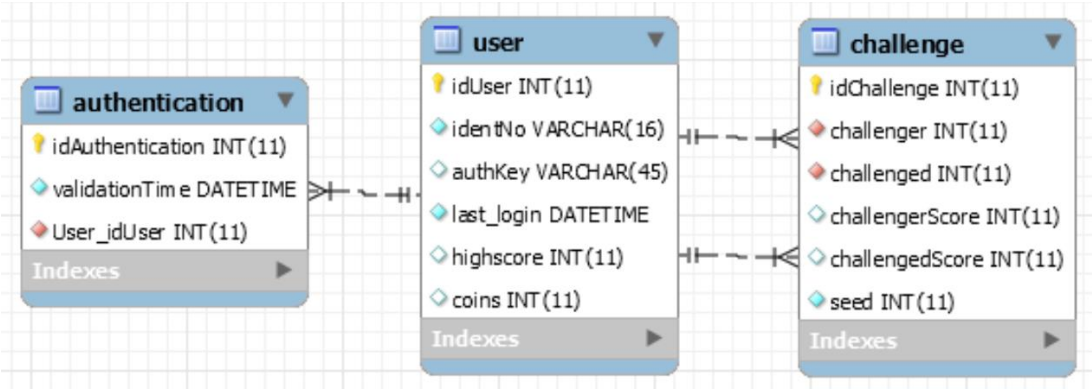


Abbildung 63: Datenmodell

Das verwendete Datenmodell besteht grundsätzlich aus drei Tabellen (*user, challenge, authentication*).

Wie bereits der Name der Tabellen verrät, werden in den Tabelle

- ***user*** die Benutzerinformationen
  - *idUser (der Primary Key der Tabelle user)*
  - *identNo* (Facebook ID des Spielers)
  - *authKey* (Leerfeld)
  - *last_login* (der Zeitstempel, an dem sich der User das letzte Mal angemeldet hat)
  - *highscore* (die höchste Punkteanzahl, die der Spieler erreicht hat)
  - *coins* (Die Münzanzahl, die der Spieler besitzt)

- ***challenge*** die Challenges der Spieler untereinander
  - *idChallenge* (der Primärschlüssel der Tabelle *challenge*)
  - *challenger* (die ID des Herausforderers; Fremdschlüssel der Tabelle *user*)
  - *challenged* (die ID des Herausgeforderten; Fremdschlüssel der Tabelle *user*)
  - *challengerScore* (die erreichte Punkteanzahl des Herausforderers)
  - *challengedScore* (die erreichte Punkteanzahl des Herausgeforderten)
  - *seed* (Zufallszahl, mit der der Abstand zwischen den Hindernissen berechnet wird)

- *authentication* die Authentication Keys (Tabelle wird nicht verwendet)
    - *idAuthentication* (der Primärschlüssel der Tabelle *authentication*)
    - *validationTime* (der Zeitstempel, an dem der Authentication-Key vergeben wurde)
    - *User_idUser* (die ID des Users, dem der Authentication-Key zugewiesen wurde; Fremdschlüssel der Tabelle *user*)

gespeichert.

Damit die Datenbank bereits Testdaten bei der Portierung enthält, wurden Dummy-Daten (Benutzer, Challenges) eingefügt.

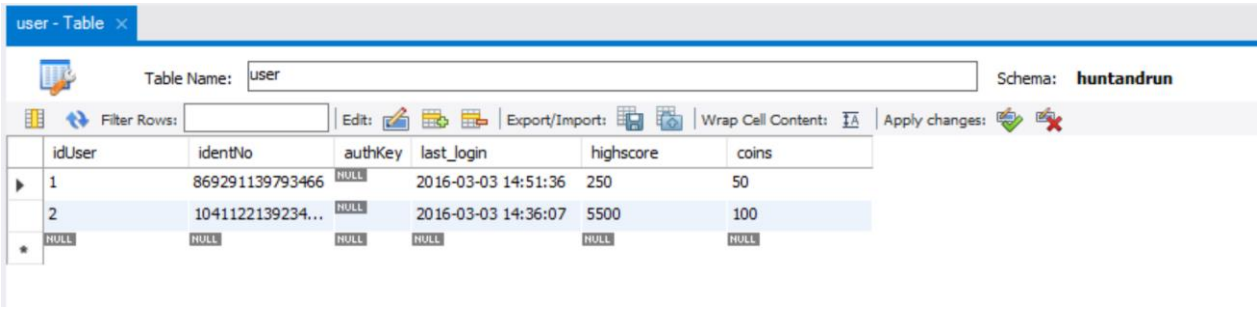


Abbildung 64: Beispielhafte Dummy-Daten der Tabelle user

Nach der erfolgreichen Modellierung wurde ein Datenexport veranlasst, der die entworfenen Tabellen sowie deren Dummy-Daten zu SQL-Befehlen umgewandelt und gespeichert hat.

6.2.2.2 Datenbankkonfiguration und Portierung

Nachdem der Datenbankentwurf abgeschlossen wurde, ist die Datenbank auf dem entfernten Schulserver konfiguriert worden.
Bei der Installation wurde zusätzlich darauf geachtet, dass sich der MySQL-Dienst auch nach einem eventuellen Neustart des Servers automatisch startet, um einen dauerhaften Ausfall der Datenbank zu verhindern.
Als Benutzer wurde der Account *root* eingerichtet, der die DBA-Rechte besitzt. Auch wurde darauf geachtet, dass sich dieser Benutzer nur von der Adresse *localhost* anmelden kann, um eine Übernahme der Datenbank von einem entfernten Computer zu verhindern. Somit muss man, um Änderungen an der Datenbank zu unternehmen, sich den physikalischen Zugang zum Computer verschaffen, oder per Remote-Desktop-Verbindung am Server arbeiten.

Nach dem erfolgreichen Start des MySQL-Dienstes wurde wiederum mit der *MySQL Workbench* das entworfene Datenmodell eingespielt. Dabei kam die erstellte Datei mit den exportierten SQL-Befehlen zum Einsatz.

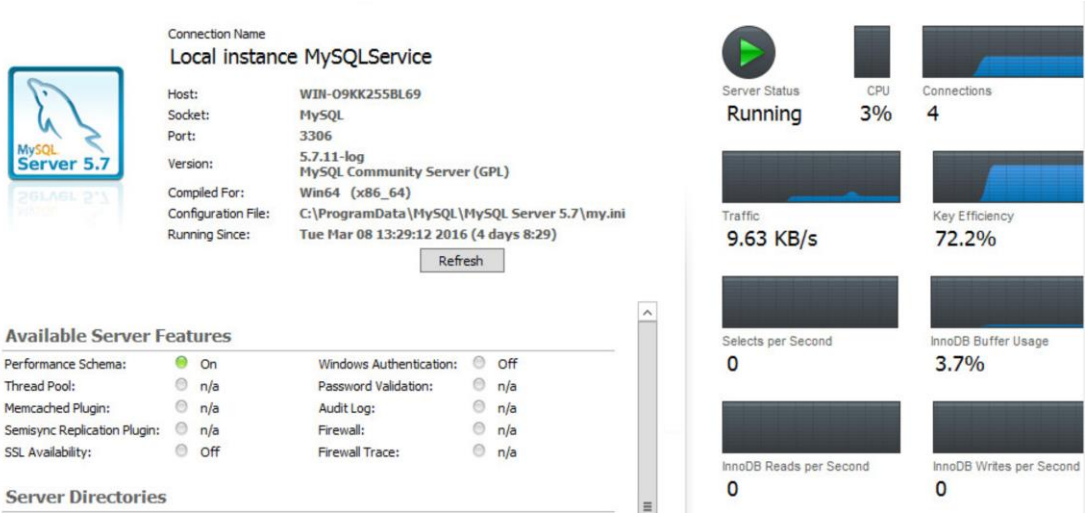


Abbildung 65: Laufende DB-Instanz

Man kann sich auch in der MySQL Workbench den aktuellen Serverstatus ansehen, sowie die Serverauslastung und die Connections, die zu diesem Zeitpunkt mit der Datenbank offen sind.

7 Probleme und Zukunftsaussichten

Dieses Kapitel handelt von unseren Problemen bei der Ideenfindung für unsere Diplomarbeit und vom Entwurf und der Umsetzung unserer endgültigen Spielidee.

7.1 Probleme bei Ideenfindung

Die Probleme, die bei der Ideenfindung aufgetreten sind, werden in diesem Kapitel näher erläutert.

7.1.1 NFC

Im Anfangsstadium der Kooperation mit der Johannes Kepler Universität Linz war die Idee unserer Diplomarbeit an die NFC-Technologie angelehnt. Dabei sollten wir eine App entwickeln, die die letzten fünf Transaktion einer Bankomatkarte ausliest und diese am Server speichert.

Gleichzeitig sollte diesen Transaktionen auch Geschäften bzw. den sogenannten Terminals zugeordnet werden.

Genau daran scheiterte unsere Idee: der Standard ISO 14443, der die Richtlinien zum Auslesen der EMV (Europay International, MasterCard und VISA) Karten festlegt, deklariert keine Schnittstelle zum Auslesen der ID des Terminals, auf diesem die Transaktion stattgefunden hat. Somit war es nicht möglich, die Transaktion einem Geschäft zuzuordnen.

7.1.2 Spielidee finden

Wir haben bereits relativ früh nach dem Abbruch unseres NFC-Projekts gewusst, dass wir ein Spiel entwickeln möchten. Jedoch ist es gar nicht so einfach eine Spielidee zu finden, die es so in der Form nicht schon gibt. Einige Zeit floss also schon vor dem Start der Diplomarbeit in die Recherche, ob unsere Spielideen schon in einem der App-Stores vorhanden sind.

Obwohl es schon viele „Jump and Run“ in den Stores gibt, gibt es unsere App in dieser Form noch nicht und daher haben wir uns beschlossen dieses Spiel umzusetzen.

7.2 Probleme bei Entwurf und Implementierung

Im folgenden Abschnitt werden einige unserer Probleme während des Entwurfes und der Implementierung erläutert.

7.2.1 Konkretisierung des Spielablaufes

Da das Spiel auf der einen Seite einfach zu spielen sein soll, aber wiederum auch mit der Zeit schwieriger werden soll, musste der Spielablauf dementsprechend entwickelt werden.

Durch die Hindernisse, die in gewissen Abständen auf dem Bildschirm erscheinen, erhöht sich der Schwierigkeitsgrad für den Spieler. Gleichzeitig wird auch periodisch die Geschwindigkeit, mit der sich der Spieler fortbewegt, erhöht. Dies führt dazu, dass der Spieler immer schneller auf Hindernisse reagieren muss, um kein Leben zu verlieren.

7.2.2 Controllerstruktur

Da die anfangs programmierten Menü-sowie Spielcontroller sehr komplex und verschachtelt waren, waren die Aufrufe dieser sehr unübersichtlich. Beispielhaft wurde im Controller das Menü aufgebaut, gleichzeitig wurde aber auch der Facebook-Login aufgerufen.
Um eine klare Struktur zu schaffen, wurde für jeden Bereich (Login-Controller, Hauptmenü-Controller…) ein möglichst abgekapselter Controller geschrieben, der die Aufrufe vereinfacht und auch voneinander trennt.
Der Vorteil darin liegt auch, dass die Controller, die in einer Scene nicht mehr gebraucht werden, nun nicht mehr im Hauptspeicher zwischengespeichert werden. Somit kann der Speicherplatz freigegeben werden, dies führt wiederum auch zur besseren Hauptspeichereffizienz.

7.2.3 Charakter-Animation auf verschiedenen Geräten

Um die Animationen des Charakters möglichst realistisch aussehen zu lassen haben wir die Füße des Charakters entsprechend in Bewegung gesetzt und den Oberkörper durch leichte Bewegung auf der y-Achse bzw. durch eine seitliche Rotation versucht an die Laufbewegung anzupassen.
Beim Testen der Animation auf Android-Geräten wurde der Charakter doch dann plötzlich einfach ausgeblendet. Auf dem Unity-Editor allerdings lief die Animation einwandfrei ab. Es war nicht klar, wodurch diese Probleme verursacht werden. Nach langer Recherche hat sich dann herausgestellt, dass eine Rotation des Game-Objekts leider gar nicht möglich ist auf einigen Geräten. Die Animation musste darauf dahingehend angepasst werden, so dass die Rotation des Oberkörpers nicht mehr verwendet wird. Das hat zur Folge, dass die Animation nicht so natürlich aussieht wie geplant.
Das zeigt, dass der Vorteil von Cross-Platform Entwicklungsumgebungen, nur einmal entwickeln zu müssen, oft aufgehoben wird durch die Probleme, die bei der Ausführung auf den verschiedensten Geräten entstehen.

7.2.4 Git-Problem mit Unity Assets

Um das Problem der Datensynchronisation im Projektverlauf zu beheben, benutzten wir ein kostenloses GitHub-Repository, auf dem wir die Änderungen des Programmcodes speicherten.
Unity speichert allerdings die teilweise automatisch erstellten Dateien nicht im Textformat ab, sondern im Binärformat-dieses Format wird nicht, wie beim Textformat, automatisch zusammengefügt, sondern bei jeder Änderung überschrieben. Dies hat dazu geführt, dass die Änderungen teilweise vom Partner überschrieben worden sind, obwohl man dies nicht wollte.
Zur Lösung dieses Problems musste man Unity so einstellen, dass die automatisch generierten Metadaten sowie Assets im Textformat abgespeichert werden.

7.2.5 Grafiken

Da unsere Kenntnisse im Zeichnen von Comic-Zeichnungen beschränkt sind, mussten wir uns bei der Suche von Grafiken auf rechtlich freie Materialien aus dem Internet beschränken. Diese haben wir dann mithilfe des Bildbearbeitungsprogrammes Photoshop an unsere Bedürfnisse angepasst.
Die Suche war sehr zeitaufwändig. Vor allem weil wir nur kostenlose Materialien gesucht haben, welche sehr rar sind im Internet.

7.2.6 Facebook SDK für Unity

Als großes Hindernis hat sich das zuvor so vielversprechend aussehende Facebook SDK für Unity herausgestellt. Denn es funktioniert nicht auf jeder von Unity bereitgestellten Plattform einwandfrei. Außerdem ändern sich die Versionen schnell. Hört sich erstmal gut an, da man glaubt, man hat immer ein neues, sehr gut funktionierendes, SDK. Doch es treten leider bei der Aktualisierung auf eine neue Version sehr viele Fehler auf und das nimmt einiges an Zeit in Anspruch.

7.3 Zukunftsaussichten

Um den im Rahmen dieser Diplomarbeit entwickelten Prototypen zu verbessern haben wir uns bereits Gedanken über mögliche Erweiterungen gemacht. Im folgenden Abschnitt werden unsere Gedanken zu den Erweiterungsmöglichkeiten erläutert.

7.3.1 Mehrere Levels

Ein sehr wichtiger Punkt, um das Produkt zu verbessern ist die Erweiterung um Levels. Derzeit gibt es nur eine Umgebung in der das Spiel gespielt werden kann. Um aber dem Nutzer mehr Möglichkeiten zu bieten, so dass dieser die Motivation am Spiel schwerer verliert.
Mögliche Umgebungen für neue Levels könnten sein: Wüste, Dschungel, Berglandschaft, Eislandschaft und viele andere. Bei den unterschiedlichen Levels wären dann auch andere Gegner und Hindernisse, jeweils angepasst an die Umgebung, möglich.
Um diese Levels freizuschalten muss der Benutzer dann eine gewisse Summe an Coins bezahlen, die er sich zuvor erspielt hat. Das motiviert auch wieder, das Spiel zu spielen.

7.3.2 Verschiedene Charaktere

Um die Bindung des Benutzers zu seinem Spielecharakter zu erweitern werden verschiedene Spielecharaktere angeboten. Eventuell können diese Spielecharaktere auch mit einem Editor bearbeitet werden und individuell angepasst werden. Um die Anpassung vorzunehmen muss der Benutzer aber zuerst durch das wiederholte Spielen die Gegenstände am Ende von Levels finden und so freischalten.

7.3.3 Power-Ups

Power-Ups, also Gegenstände, die der Benutzer einsetzen kann, um zum Beispiel die Lebenspunkte im derzeitigen Lauf zu erhöhen oder eine temporäre Unverwundbarkeit zu erreichen, können eine gewissen Eintönigkeit des Spiels überspielen und für mehr Spielspaß sorgen.

7.3.4 Ingame Store

Dies ist vor allem ein wichtiger Punkt für die Finanzierung des Spiels. Viele Spiele heutzutage basieren auf den InApp-Käufen. In unserem Ingame-Store können wir Levels, Charaktere, Kleidung für Charaktere und Power-Ups anbieten. Um das „Pay to Win“-Prinzip, welches Spielen oft angeprangert wird, wenn der Erfolg in diesen Spielen zu stark vom Kauf von Gegenständen aus Ingame-Stores abhängt, zu vermeiden können diese entweder mittels erspielten Coins oder mittels Echtgeld gekauft werden. So hat jeder Benutzer die Möglichkeit das vollständige Spiel auch kostenlos zu spielen.

7.3.5 Musik und Geräusche

Ein wohl auch sehr wichtiger Erfolgsfaktor eines Spiels sind Musik und Geräusche. Viele Spiele haben aufgrund dieser einen hohen Erkennungswert und fördern so die Verbreitung. Außerdem wird der Spielspaß durch die Wiedergabe von Musik und Geräuschen erweitert.

7.3.6 Anbindung an mehrere soziale Plattformen

Derzeit ist nur eine Facebook-Anbindung vorhanden, welches im Grunde schon den größten Teil an potentiellen Nutzern abdeckt. Jedoch gibt es auch viele Nutzer die Facebook generell meiden, oder Nutzer, die lieber nicht ihren Facebook-Account zur Anmeldung für Spiele heranziehen, sondern ihren Google-Account. Um es also möglichst allen Benutzern recht zu machen ist es gut, die verschiedensten Anmeldeoptionen zu bieten. Von Facebook bis Google+ bzw. Google Play oder auch eine einfache Anmeldung mit der E-Mail-Adresse.

7.3.7 Release in Stores

Das Spiel wird nach Implementierung der zuvor genannten Features in den Google-Play Store veröffentlicht und danach eventuell in den Windows-Store.

8 Anhang und Ergänzungen

8.1 Aufgabenteilung

Simon Angerbauer	Roman Socovka
Kurzfassung/Abstract	Kurzfassung/Abstract
1.1 Diplomanden	1.1 Diplomanden
1.2 Auftraggeber	1.3.1 Schule
1.3.3 Ansprechpartner des Auftraggebers	1.3.2 Betreuungslehrer
1.5 Zuständigkeiten	1.4 Projektorganigramm
2.2.1 Ziel für den Auftraggeber	2.1 Thema und Aufgabenstellung
2.3 Ausgangssituation	2.2 Zielsetzung
3.1.3 Produktfunktionen mit GUI-Abbildung	3.1.1 Grenzkriterien
3.3 Projektstrukturplan	3.1.2 Produkteinsatz
4.1 Game Cycle	3.2 Projektablaufplan
4.2 Designentwurf	3.4 Ressourcenplan
5.1.1 Unity	3.5 Aufwandsschätzung
5.1.4 Facebook Graph API	4.2 Designentwurf
6.1.1 Anwendung	5.1.2 WildFly Application Server
6.1.2 Facebook-Anbindung	5.1.3 MySQL
7.1 Probleme bei Ideenfindung	5.2 Schnittstellen
7.2 Probleme bei Entwurf und Implementierung	6.1.1.4 Parallaxing
7.3 Zukunftsaussichten	6.2 Back-End
	7.1 Probleme bei Ideenfindung
	7.2 Probleme bei Entwurf und Implementierung

Tabelle 12: Aufgabenteilung

8.2 Verweise

ECLIPSE.ORG. *Eclipse- Bezeichnung*. [online]. [Accessed 2016 Jan 5]. Available from World Wide Web: <https://www.eclipse.org/downloads/>
ECLIPSE.ORG. *Eclipse- Logo*. [online]. [Accessed 5 Jan 2016]. Available from World Wide Web: <https://eclipse.org/eclipse.org-common/themes/solstice/public/images/logo/eclipse-800x188.png>
FACEBOOK. *Facebook Graph API*. [online]. [Accessed 7 Mar 2016]. Available from World Wide Web: <https://developers.facebook.com/docs/graph-api/using-graph-api>
FACEBOOK. *Facebook Logo*. [online]. [Accessed 15 Jan 2016]. Available from World Wide Web: <https://upload.wikimedia.org/wikipedia/commons/thumb/7/7c/Facebook_New_Logo_%282015%29.svg/399px-Facebook_New_Logo_%282015%29.svg.png>
HTL PERG. *HTL Perg Logo*. [online]. [Accessed 28 Dec 2015]. Available from World Wide Web: <http://www.htl-perg.ac.at/templates/htl-perg/images/logo.gif>
IDC. *Smartphone OS Market Share Statistic*. [online]. [Accessed 30 Dec 2015]. Available from World Wide Web: <http://www.idc.com/prodserv/smartphone-os-market-share.jsp>
JKU LINZ. [online]. [Accessed 19 Feb 2016]. Available from World Wide Web: <http://www.jku.at/fs_resources/img/jku_logo_de.svg>
MET.GUC.EDU.EG. [online]. [Accessed 15 Mar 2016]. Available from World Wide Web: <http://met.guc.edu.eg/OnlineTutorials/static/article_media/jsp%20-%20servlets/LoginExample%20[4].jpg>
MICHAELSTULTS.COM. *MySQL Workbench- Logo*. [online]. [Accessed 6 Jan 2016]. Available from World Wide Web: <http://www.michaelstults.com/wp-content/uploads/2014/10/MySQLWorkbench.png>
OBERMÜLLER, Rupert. [online]. [Accessed 30 Dec 2015]. Available from World Wide Web: <http://www.htl-perg.ac.at/organisation/lehrer>
ONS.GOV.UK. *Internetbenutzungsstatistik*. [online]. [Accessed 30 Dec 2015]. Available from World Wide Web: <http://www.ons.gov.uk/ons/rel/rdit2/internet-access---households-and-individuals/2012-part-2/stb-ia-2012part2.html>
OTREVA.COM. *MySQL- Logo*. [online]. [Accessed 6 Jan 2016]. Available from World Wide Web: <https://s3.amazonaws.com/otreva-cdn-3/wp-content/uploads/2013/12/mysql-logo.jpg>
S.RICHMOND. *Unity Source Control Settings*. [online]. [Accessed 24 Mar 2016]. Available from World Wide Web: <http://stackoverflow.com/questions/18225126/how-to-use-git-for-unity-source-control>
SMARTINSIGHTS.COM. [online]. [Accessed 2015 Dec 30]. Available from World Wide Web: <http://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics/>
STEINBAUER, Matthias. *Matthias Steinbauer*. [online]. [Accessed 28 Dec 2015]. Available from World Wide Web: <https://x1.xingassets.com/image/5_e_f_eb4f8e40d_9928458_1/matthias-steinbauer-foto.1024x1024.jpg>
TUTORIALSPOINT.COM. [online]. [Accessed 18 Jan 2016]. Available from World Wide Web: <http://www.tutorialspoint.com/jpa/jpa_architecture.htm>
UNITY. *Unity Logo*. [online]. [Accessed 7 Jan 2016]. Available from World Wide Web: <http://forum.unity3d.com/attachments/logo-titled-png.16698/>

UNITY. *Unity Ziel Plattformen*. [online]. [Accessed 7 Jan 2016]. Available from World Wide Web: <https://unity3d.com/unity/multiplatform>
WIKIPEDIA. *Eclipse- Informationen*. [online]. [Accessed 05 Jan 2016]. Available from World Wide Web: <https://de.wikipedia.org/wiki/Eclipse_(IDE)>
WIKIPEDIA. *Facebook-Informationen*. [online]. [Accessed 15 Jan 2016]. Available from World Wide Web: <https://de.wikipedia.org/wiki/Facebook>
WIKIPEDIA. *MySQL- Informationen*. [online]. [Accessed 6 Jan 2016]. Available from World Wide Web: <https://de.wikipedia.org/wiki/MySQL>
WIKIPEDIA. *MySQL Workbench- Informationen*. [online]. [Accessed 6 Jan 2016]. Available from World Wide Web: <https://de.wikipedia.org/wiki/MySQL_Workbench>
WIKIPEDIA. *Unity Informationen*. [online]. [Accessed 7 Jan 2016]. Available from World Wide Web: <https://de.wikipedia.org/wiki/Unity_%28Spiel-Engine%29>
WIKIPEDIA. *WildFly- Informationen*. [online]. [Accessed 2016 Jan 5]. Available from World Wide Web: <https://de.wikipedia.org/wiki/WildFly>
WIKIPEDIA.ORG. *Bewegungsparallaxe*. [online]. [Accessed 20 Mar 2016]. Available from World Wide Web: <https://de.wikipedia.org/wiki/Bewegungsparallaxe>
WILDFLY. *WildFly- Logo*. [online]. [Accessed 5 Jan 2016]. Available from World Wide Web: <http://wildfly.org/>

8.3 Abbildungsverzeichnis

8.4 Tabellenverzeichnis

8.5 Abkürzungsverzeichnis

A

API
Programmierschnittstelle 2, 19, 38, 42, 45, 46, 53, 60, 61, 80

D

DAO
Data Access Object 67
DBA
Datenbankadministrator 73

E

EPL
Eclipse Public License 39
EULA
Endbenutzer-Lizenzvertrag 34

G

GPL
GNU General Public License 40
GUI

I

J

L

N

R

S