Planeten Browsergame

Tim Felix Tanner (1151110)

Bastian Schneider (1151420)

Einleitung

Die Folgende Projektbeschreibung schildert den Aufbau und Ablauf des Projekts im Modul Mobile Applikationen der Fachhochschule Bielefeld, welches wir im laufe des Semesters durchgeführt haben.

Projektbeschreibung

Die Entwicklung eines Spiels, das auf Browserspielen basiert, die früher populär waren. Das Thema des Spiels ist, aufgrund der Tatsache das der Fantasie kaum Grenzen gesetzt sind, Science-Fiction. Das Spiel findet in einem fiktivem Universum statt und der Spieler übernimmt die Rolle des Herrschers eines Planeten.

Projektziel

Die entwickelte Applikation soll anschaulich und intuitiv sein. Es wird bewusst auf Animationen und ähnliches verzichtet, um den Stil und Charme der vorher genannten Browserspiele beizubehalten. Die Benutzeroberfläche soll simpel gehalten werden und dennoch alle nötigen Funktionen klar erkennbar darstellen um eine einfache Bedienung zu gewährleisten. Das Design soll einheitlich sein um der Applikation ein natürliches Aussehen zu verleihen.

Das Spielkonzept soll einfach gehalten werden und trotzdem dafür sorgen dem Spieler Spaß und die Lust zum weiterzuspielen bringen. Berechnungen und ähnliches sollen daher auch im Hintergrund geschehen um den Fluss des Spiels so wenig wie möglich zu unterbrechen.

Projektbegründung

Aufgrund der Tatsache, das kaum Browserspiele, so wie sie sind, als Mobile Applikationen veröffentlicht wurden haben sich wir dazu entschieden dieses Projekt zu bearbeiten.

Projektschnittstellen

Die Applikation, die vom Benutzer bedient wird, ist nur ein Teil eines Systems. Die Eingaben des Benutzers werden an einen Server gesendet und dort verarbeitet. Der Server wurde mithilfe von Node.js programmiert. Des weiteren gibt es eine MySQL Datenbank zur Speicherung von verschiedensten Werten. Es wurde außerdem MySQL Workbench zur Überwachung und zum einpflegen der Daten verwendet. Die Kommunikation zwischen Server und Datenbank ist auch mit Node.js geregelt und im Server implementiert. Genauere Informationen befinden sich im Kapitel Implementierung. Der Benutzer interagiert also nur mit dem Client, während alles andere an anderer Stelle passiert. Mehr dazu im Kapitel, in dem die Architektur beschrieben wird.

Projektabgrenzung

Da die Arbeitszeit am Projekt beschränkt ist, soll die Applikation mit wenigen Spielern im Sinn implementiert werden.

Stand der Technik

Anforderungsdokumentation

Übersicht

Must have:

- [x] Basen können ausgebaut werden (Minen, Truppen)
- [x] Angriffe auf Spieler
- [x] Ressourcenabbau
- [x] Bilder

- [x] Server-Client Architektur
- [x] Datenbank
- [x] Mindestens drei verschiedene Einheiten mit mindestens zwei verschiedenen Werten (Angriff, Leben)
- [x] Funktionierendes Kampfsystem
- [x] Erstellung eines Kontos und Anmeldung

Should have:

- [ ] Nicht-Spieler-Charaktere
- [x] Forschung
- [x] Verbesserungen (Truppen/Gebäude)
- [x] Verteidigungsanlagen
- [x] Ansprechenderes Design
- [x] Einfaches Kampfsystem

Could have:

- [ ] zweidimensionale Karte
- [ ] neue Server (Verschiedene Spielmodi)
- [ ] Sound bzw. Musik
- [x] Mehr Einheitenauswahl
- [x] Komplexes, ausbalanciertes Kampfsystem

Won't have:

- [ ] Animationen
- [ ] Werbung
- [ ] Bildschirmdrehung

Verlauf

Zu Beginn des Projekts waren wir sehr motiviert und haben geglaubt, das wir auch die could have's teilweise noch erreichen könnten. Im Laufe der Arbeit am Projekt hat sich herausgestellt das diese Einschätzung sehr optimistisch war und wir haben uns darauf geeinigt uns mit den should have's zufrieden zu geben. Am Ende ist es dann doch noch dazu gekommen das wir unsere erste Zielsetzung erreicht haben.

Erste Konzepte

Landscape Layout

Bild 1: Mockup der Startseite

Bild 2: Mockup der Übersichtsseite

Bild 3: Mockup der Planetenseite

Bild 4: Mockup der Gebäudeseite

Portrait Layout

Bild 5: Mockup der Übersichtsseite

Bild 6: Mockup des Dropdown Menüs

Bild 7: Mockup der Ressourcenseite

Bild 8: Mockup der Detailansicht

Bild 9: Mockup der Gebäudeseite

Bild 10: Mockup der Forschungsseite

Finales Konzept

Hauptsächlich wurden in den tatsächlichen Entwurf der Benutzeroberfläche Ideen aus den Mockups im Portrait Layout übernommen, da sich diese Darstellungsweise besser für Auflistungen eignet als das Landscape Layout.

Der Startbildschirm wie er in *Bild 1* zu sehen ist wurde verworfen, da es innerhalb der Applikation kein Einstellungsmenü gibt. Damit verliert der Startbildschirm seine Funktion. Nun ist der Startbildschirm eine Übersichtsseite die der in *Bild 5* ähnlich sieht.

Auch das Dropdown Menü (*Bild 6*) wurde fast genau so übernommen. Die Planetenleiste am rechten Rand der Bilder mit Portrait Layout ist ganz weggefallen, da jeder Spieler nur einen Planet hat und nicht mehrere wie zu Anfang festgelegt. Die Seiten zur Anzeige der Gebäude (*Bild 9*) und Forschung (*Bild 10*) sind fast eins zu eins übernommen worden. Der einzige unterschied hier ist, das sie jetzt die gesamte Breite einnehmen.

Die Wahl der Farbe

Architekturbeschreibung

Aufbau

Wir haben nach dem Entwicklungsmodell Modell View Controller gearbeitet, um die Logik des Programms von der Benutzeroberfläche zu trennen. Wobei wir für die Datenspeicherung also der View eine Datenbank verwenden. Der Controller wurde durch einen Server umgesetzt und die View durch den Client. Der Client ist nur dazu da die Daten anzufordern und sie anzuzeigen, die Logik ist im Controller also bei uns im Server realisiert.Nachfolgend werden diese Komponenten beschrieben und unsere Entscheidungen erklärt.

Client

Da wir eine mobile Applikation schreiben wollten haben wir uns für die Entwicklungsumgebung Android Studio entschieden, um den Client darzustellen. Android Studio hat eine umfangreiche Auswahl von Bibliotheken und Möglichkeiten zur Gestaltung der Benutzer-Oberfläche. Der Client ist dazu gedacht die Daten darzustellen und sendet Http-Requests an den Server um Ressourcen anzufordern.

Server

Der Server verwaltet die Daten aus der Datenbank und beantwortet die Anfragen vom Client. Er berechnet die Kämpfe und die manipuliert die Daten nach Aktion des Spielers. Er hat die Aufgabe die gesamte Spiellogik zu halten und ist deshalb auch das komplexeste der drei Teilprogramme. Wir haben uns hier für die Umsetzung in Node.js entschieden. Als Programmiersprache wird Javascript verwendet. So fiel es uns besonders leicht mit JSON-Objekten umzugehen. Javascript

unterstützt Methoden um aus den Datenbank Queries erhaltene Datensets, direkt in JSON umzuwandeln und diese an die Http-Responses anzuhängen.

Datenbank

Wir haben uns für die Nutzung einer Datenbank entschieden, um die Daten von Spielern zu speichern und um den Spielfortschritt festzuhalten. Als Datenbankmanagementsystem haben wir uns für MySQL entschieden. Der Grund dafür ist die einfache Handhabung und die nativen SQL-Queries. Wir haben im Umgang mit diesem DBMS schon positive Erfahrungen gemacht und konnten unser Vorwissen nutzen um relativ schnell eine funktionsfähige Datenbank aufzubauen. Mit der MySql Workbench konnten wir ein Entity-Relationship-Modell erstellen. Aus diesem Modell wurde dann ein Schema erstellt das auf einem Datenbank-Server läuft.

Implementierung

Projektplanung

Zeitplanung

Es war ein Zeitraum von 13 Wochen vorgegeben. Wir haben uns für das Spiralmodell entschieden. Die Begründung kann im Abschnitt Entwicklungsprozess eingesehen werden. Eine grobe Zeiteinteilung haben wir wie folgt vorgenommen. Die Angaben sind in Prozent weil wir in einem iterativen Modell immer wieder zwischen den Phasen wechseln.

Projektphasen	geplante Zeit in %
Planung	10
Entwurf	20
Implementierung	30
Zusammenführung	5
Tests	30
Dokumentation	5

Terminplanung

Nachfolgend ist der Meilensteinplan. Es wurden alle Meilensteine eingehalten.

Nummer	Bezeichnung	Soll Termin	Ist Termin
1	Projektstart	14.10.2019	14.10.2019
2	Projektplanung	21.10.2019	21.10.2019
3	Server Client Kommunikation hergestellt	28.10.2019	28.10.2019
4	Oberfläche entworfen	11.11.2019	11.11.2019
5	Datenbank eingebunden	25.11.2019	25.11.2019
6	erste Funktionalität benutzbar	17.12.2019	17.12.2019

Nummer	Bezeichnung	Soll Termin	Ist Termin
7	Projekt funktionsfähig	06.01.2019	17.12.2019
8	Projektübergabe	13.01.2020	13.01.2020

Personalplanung

An dem Projekt haben zwei Entwickler gearbeitet. Die wichtigsten Entscheidungen wurden zusammengetroffen. Die Arbeiten wurden aufgeteilt und es standen für Nachfragen oder Hilfestellungen zwei Dozenten zur Verfügung.

Sachmittelplanung

Sämtliche benutzte Software ist Open-Source-Software, also kostenfrei verfügbar. Da es kein gewerbliches Projekt ist und auch nie als eines geplant war, wurden keine Investitionen vorgenommen. Von Mockplus wurde eine einwöchige Testversion genutzt. Diese Programme wurden benutzt:

- MySQL Workbench
- Node.js
- Android Studio
- Mockplus
- git

Es wurden zwei Computer für die Programmierung genutzt und ein Laptop für die Präsentation.

Entwicklungsprozess

Wir haben uns für das Spiralmodell als Entwicklungsmodell geeinigt, weil das Projekt zu komplex ist, um es in einem Zug zu implementieren. Es muss immer wieder getestet werden bevor neue Funktionalitäten implementiert werden können. Ansonsten hätten lange Fehlersuchen den Entwicklungsprozess zu stark verlangsamt. Nachdem das Grundgerüst des Programms stand, wurden im gesamten Entwicklungsprozess, iterativ, neue Features implementiert.damit getestet werden konnte ob sie mit dem schon vorhanden Code synergieren.

Projektdurchführung

Nachdem die vorbereitenden Mapnahmen abgeschlossen waren, kommen wir zur Projektdurchführung. In diesem Kapitel beschreiben wir den Ablauf des Projekts, gehen auf einige Arbeitsschritte und die Qualitätssicherung ein.

Entscheidungsfindung

Als erstes galt es ein geeignetes Datenbank Management System zu evaluieren. Die am weitesten verbreiteten Datenbank Management Systeme sind Oracle, MySQL und Microsoft SQL Server, für andere sinkt mit der Popularität auch deren Unterstützung durch Dokumentation. Wir beschränken die Entscheidungsfindung auf diese drei, weil wir auch schon Vorerfahrung haben und so ein schnelles Anlaufen des Projektes gewährleisten können. Da Microsoft SQL Server kostenpflichtig ist und uns keine Ressourcen zu Verfügung stehen, blieben noch Oracle und MySQL übrig. Durch die umfangreiche Unterstützung von MySQL in verschiedenen Sprachen und auf diversen Plattformen haben wir uns letztendlich dafür entschieden.

Für die Auswahl des Servers haben wir uns für eine lokale Variante entschieden, weil uns die Ressourcen fehlen einen Server anzuschaffen. Da wir unsere privaten Computer dafür nutzen, wollten wir bei der Gefahr eine Sicherheitslücke zu erzeugen keinen Zugriff auf unsere privaten Daten ermöglichen. Deshalb haben wir den Server selber in Node.js erzeugt und das Verhalten implementiert.

Beschreibende Arbeitsschritte

Im folgenden beschreiben wir einige elementare Arbeitsschritte, die notwendig waren um diese Projekt durchzuführen.

Aufsetzen der Datenbank

Um die Datenbank aufzusetzen, wurde ein Entity-Relationship-Model erstellt. Da es Probleme bei der Struktur der Datenbank gab, wurde die Hilfe der Dozenten in Anspruch genommen. Die Daten wurden getrennt aufgeteilt eine Benutzer bezogene Gruppe und in Daten die ausschließlich für das Spiel gedacht sind und nicht in irgendeiner Art vom Client verändert werden können. Anschließend wird das Entity-Relationship-Model in ein SQL-Script ungewandelt, damit es in die Datenbank eingespielt werden kann. Dort erzeugt es ein Schema, in dem die Daten eingesehen und verändert werden können.

Implementierung des Servers

Der Server wurde in seiner ersten Version in Java geschrieben, um die Kommunikation zu testen. Durch Recherchen sind wir dann auf eine bessere Methode gestoßen und haben den Server in Node.js implementiert. Da in Javascript Funktionen asynchron ausgeführt werden, haben wir mit Promise-Objekten gearbeitet, die das Programm zwingen auf den Rückgabewert einer Funktion zu warten. Das war unabdingbar, um die Ergebnisse der SQL-Queries an die Http-Responses anzuhängen. Ohne dieses Verfahren wurden die Http-Responses ohne Inhalt im body zurück an den Client gesendet.

Umsetzung des Clients

Zunächst haben wir uns auf die Gestaltung des Oberfläche konzentriert. Da die Oberfläche größtenteils unabhängig von dem Backend entwickelt werden kann, bietet es sich an damit anzufangen. In der ersten Anläufen wurden verschiedene Kommunikationsarten und Bibliotheken getestet. Die Entscheidung fiel dann auf HttpURLConnection, durch die leichte Erlernbarkeit und durch die ausführliche Dokumentation.

Qualitätssicherung

Kontinuierliche Tests

Während der Entwicklungsphase wurde die Software in Intervallen getestet. Es wurden Tests nach jedem neuen Feature gemacht und die Datenbank wurde auf Anomalien überprüft.

Spielkonzepte

Jeder Spieler kann auf seinem Planeten drei verschiedene Ressourcen abbauen, die für verschiedenste Gebäude und Verbesserungen benötigt werden. Der Abbau findet passiv statt. Das heißt, das auch wenn man nicht aktiv im Spiel - also "Offline" - ist werden Ressourcen erzeugt. Diese Ressourcen sind Baumaterialien, Computerchips und Treibstoff. Sie können verwendet werden, um Gebäude zu errichten, Raumschiffe und Verteidigungsanlagen zu bauen und Forschung zu betreiben. Des weiteren können Gebäude verbessert werden, was verschiedene Boni mit sich bringt. Zum einen gibt es Gebäude, die die Produktion bestimmter

Ressourcen verbessern, zum anderen gibt es Gebäude die die maximale Kapazität der Ressourcen erhöht. Des weiteren gibt es Gebäude die die Produktivität verschiedener Aspekte, wie zum Beispiel die Geschwindigkeit der Forschung oder das Bauen von Raumschiffen verbessert.

Die meisten Gebäude können dazu noch aufgewertet werden, was dazu führt das die Ressourcenkosten mit dem jeweiligen Bonus steigen. Das dient dem Zweck dem Spieler einen Sinn für das Erlangen von Ressourcen zu geben. Durch die Verbesserung von Verteidigungsanlagen kann der Spieler seine Chancen vergrößern Angriffe von anderen Spielern abzuwehren.

Angriffe sind ein weiterer Weg um Ressourcen zu erhalten. Der Spieler kann seine Raumschiffe zu einem anderen Planeten schicken und versuchen einem anderen Spieler Ressourcen zu stehlen. An dieser Stelle kommen die Verteidigungsanlagen, die man auf einem Planeten errichten und verbessern kann nun ins Spiel. Durch diese erhöht man seine Chance den Kampf zu gewinnen, da diese zusätzlich zu den eigenen Raumschiffen am Kampf teilnehmen.

Raumschiffe haben vier verschiedene Werte. Trefferpunkte, Schilde, Angriff und Feuerrate. Anhand dieser Werte wird ein Kampf simuliert. Die Verschiedenen Schiffe haben Stärken und Schwächen wie zum Beispiel eine besonders hoher Schaden aber dafür eine geringere Feuerrate oder ähnliches.

Tests und Usability

Usability

Benutzeroberfläche

Die Benutzeroberfläche wurde so entworfen, dass sich alle Funktionen entweder als Einträge im immer sichtbaren Dropdown Menü wiederfinden oder als Buttons an dem von der ausgeführten Aktion betroffenem Element sind. Des weiteren wurde für alle Seiten mit ähnlichem, Listenartigen Aufbau das gleiche Muster verwendet. Einzelne, zusammengehörende Elemente sind von einem Rahmen umgeben um sie voneinander abzugrenzen. Das Dropdown Menü beinhaltet fast ausschließlich Punkte zum Wechseln der Seite. Das sind Aktionen, die man nicht permanent benötigt und daher werden sie durch die Platzierung im Dropdown Menü versteckt. Dadurch kann Platz für die wichtigeren Aktionen geschaffen werden, die somit mehr ins Auge fallen.

Zusammenfassung

Soll-/Ist-Vergleich

Bei einer rückblickenden Betrachtung des Projektes, kann festgehalten werden, dass fast alle zuvor festgelegten Anforderungen erfüllt wurden. Außerdem wurden Aspekte erfüllt, die über die Planung hinaus gehen.

Lessons learned

Im Laufe des Projekts haben wir wertvolle Erfahrungen bezüglich der Entwicklung von Applikationen sammeln können. Des weiteren haben wir festgestellt das die Entwicklung eines Spieles eine größere Herausforderung ist, als wir Anfangs dachten. Außerdem haben wir viel neues über die Client Server Architektur, explizit die Kommunikation über HTTP sowie die Nützlichkeit von Node.js lernen können. Node.js regelt einen Großteil des handshakes von HTTP, sodass man sich nur noch um die Verarbeitung der Daten kümmern muss. Auch im Umgang mit einer Datenbank ist es von großem Nutzen, da man Queries als Strings sehr einfach ausführen kann und man ein Objekt zurück bekommt, dass man direkt versenden kann. Auch haben wir von Neuem gemerkt das das Dateiformat JSON die Kommunikation und Datenverarbeitung durch die Objektifizierung deutlich erleichtert.