PowPow

Vorwort

Der Ablauf dieser Beschreibung haltet sich an die Ordnerstruktur des Projekts. Dabei wird vom Haupt-Order aus hierarchisch tiefer in das Projekt eingedrungen während jede Ordner-Ebene technisch beschrieben wird.

Ebene (Hauptordner) /:

Im Hauptorder liegen die Files, die global für das ganze Projekt relevant sind.

Bei diesem Projekt wurde “npm” als Dependency-Manager verwendet (Frontend und Backend). Im weiteren Verlauf dieser Dokumentation wird nur auf die relevanten Module eingegangen.

Um den Arbeitsablauf effizienter zu gestalten, wurde “Grunt” als Task-Runner hergenommen um Aufgaben wie das kompilieren von “scss”-Files zu übernehmen.

Im “config” Ordner befinden sich umgebungsabhängige Konfigurationsdateien, die als “default.json” und “production.json” benannt sind.

Im “src” Ordner befinden sich 2 Unterordner:

- client
- server

Dadurch konnte der Code sauberer geordnet werden. Der “root”-Pfad des Servers zeigt auf den “client”-Ordner, auf den wir als nächstes eingehen.

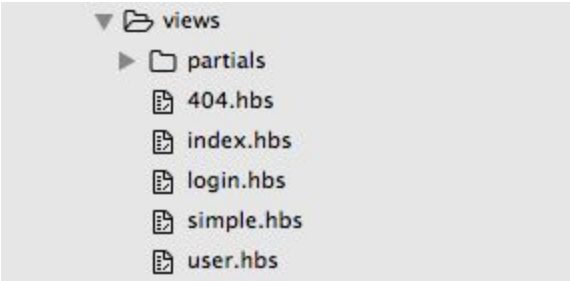
Im “README.md” wird beschrieben mit welchen befehlen man das Projekt startet.

Ebene src/client:

Im “client”-Ordner befinden sich sowohl die Spiel-Assets als auch die restlichen statischen Daten der Website (Templates, SCSS, etc.)

Direkt im Ordner befindet sich das “master.hbs”-File. Die Endung steht für die “Handlebars”-Template-Engine. Das “master”-Template wird vom Server aufgerufen und gerendert. Dabei wird die passende View aus dem “views”-Ordner hineingesetzt.

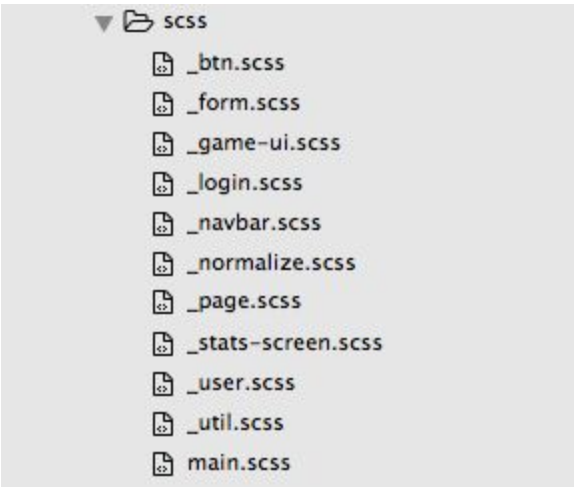
Ebene src/client/views



Im “views”-Odner befinden sich die jeweiligen Templates für die Unterseiten. Diese werden vom Server geladen, gerendert (mit Daten befüllt) und an den Browser ausgeliefert.

Im “partials”-Ordner befinden sich HTML-Schnippsel die in mehreren Templates vorkommen könnten.

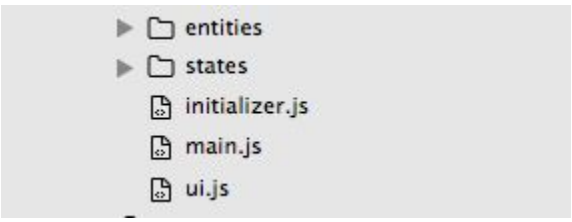
Ebene src/client/scss



Die “scss”-Files sind modular strukturiert und definieren ein Widget auf der Webpage.

Als Reset-CSS wurde “normalize.css” verwendet. Außerdem wird für den Aufbau der Seite das extrahierte Bootstrap-Grid verwendet, das als npm-Modul eingebunden wurde.

Ebene src/client/js



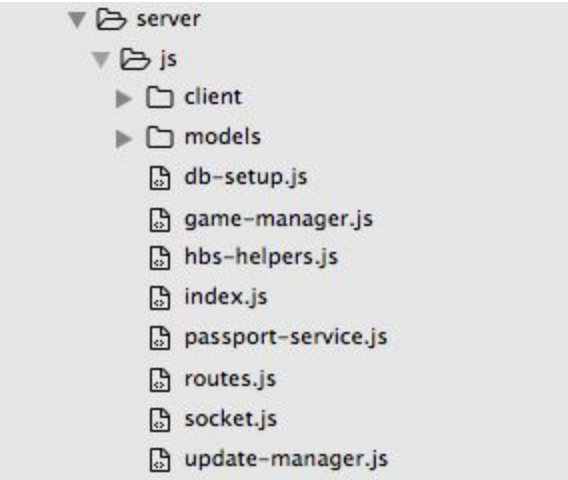
Im “js”Order befindet sich UI-Code für die Website und die Game-Logik.

Das “main.js” ist der Einstiegspunkt für die JavaScript Applikation.
Der Initializer startet das Spiel bzw. wechselt in den “Game-State”. Im Entities-Ordner befinden sich Klassen der Entitäten die im Spiel verwendet werden wie z.B.: Projektile oder der spielbare Charakter.

Die States definiere in welchem Status das Spiel sich gerade befindet. U.A. wäre da ein Preload-State, welches zum Vorladen der Assets dient.

In “ui.js” befindet sich der ganze UI-Code der Website, welcher in modulare Abschnitte geteilt ist.

Ebene src/server/js



Das “index.js”-File bietet den Einstiegspunkt in die Server Applikation. Da drin befinden sich grundsätzliche Setups für das Backend. Als Backend Framework wurde “Express” verwendet.

Im “client”-Ordner befindet sich der WebSocket-relevante Code.

Info: Die Files mit den Endungen “manager” oder “service” sind Singelton-Klassen.

Der “models”-Ordner beinhaltet die Schemas für das verwendet ORM (sequelize).

Alles ausschließlich spiele-relevante wie z.B. Generierung der Items auf der Map (Health, Munition) findet man im “game-manager”.

Zur Vereinfachung des Authentifizierungsprozesses wurde “passport.js” genommen, welches im “passport-service.js” konfiguriert bzw. eingesetzt wird.