FireForceDefense
Technical Report

Cameron Barbee, Tim Hoffmann, Christian Piffel, Tobias Schotter,
Sebastian Schuscha, Philipp Stangl, Thomas Stangl

I. Einführung und Ziele

Im vorliegenden Projekt soll das Tower-Defense-Spiel „FireForceDefense“ erweitert werden. Ziel ist es, das Spiel um eine einfache Benutzerverwaltung, d.h. Registrierung und Anmeldung, zu ergänzen. Außerdem soll eine Rangliste geschaffen werden, wofür eine Spielstand-Speicherung notwendig ist.

II. Bausteinsicht

Diese Sicht zeigt die statische Zerlegung des Systems in Bausteine sowie deren Beziehungen.

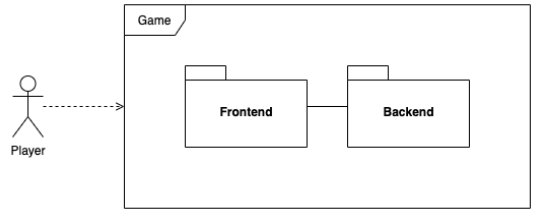

Fig. 1. Kontextabgrenzung

A. Gesamtsystem

Die Anwendung basiert auf einer Client-Server-Architektur (1). Frontend und Backend kommunizieren über eine RESTful-API.

B. Frontend

Dieser Abschnitt beschreibt die client-seitige Frontend-Architektur. Das Frontend wird unter Zuhilfenahme des Frameworks Vue.js realisiert.

Es ist selbst in mehrere Unter-Bausteine zerlegt:

1) Model: Dieser Baustein enthält die Logik für den Problembereich im Frontend. Die eigentlichen Anzeigekomponenten besitzen Referenzen auf die Instanzen der Klassen des Model-Bereichs, sodass Eingaben an das Model weitergegeben und dort verarbeitet werden können. Schließlich besitzt das Model Schnittstellen, mittels denen Informationen über den aktuellen Zustand abgefragt werden können, um basierend darauf die Anzeige anzupassen.

Im Model ist beispielsweise die Spiellogik und die Schnittstelle zum Speichern und Abrufen von Spielständen verortet.

2) Components: In diesem Modul sind die Vue-Komponenten gesammelt, mit denen die eigentliche Anzeige im Webbrowser realisiert wird. Die Komponenten sind dabei hierarchisch geordnet, so besteht ein Level beispielsweise aus einer Sidebar und der LevelMap, die LevelMap wiederum enthält einzelne Zellen und so weiter.

Die Komponenten behandeln alle Eingaben und senden bei Bedarf entsprechende Nachrichten an das Model.

3) SCSS: Hier werden zentral genutzte Stile im SCSS-Format abgelegt.

4) Lang: Dieser Baustein ist für die Internationalisierung, genauer gesagt die Übersetzung, zuständig. Aktuell ist lediglich eine deutsche Sprachvariante hinterlegt.

5) Levels: Dieses Modul enthält die Level-Definitionen. Eine Level-Definition beschreibt den initialen Aufbau des Spielfelds und die auftretenden Effekte. Jedes Level muss anhand der jeweiligen Definition im LevelManager des Model-Bereichs registriert werden.

6) Cells, Contents und Effects: In diesen drei Bausteinen sind die verschiedenen, konkreten Zell-, Inhalts- bzw. Effekt-Typen samt ihren jeweiligen Eigenschaften hinterlegt.

C. Backend

Dieser Abschnitt beschreibt die server-seitige Backend-Architektur.

1) Datenbank: Die Speicherung der Daten erfolgt im dokumentenorientierten NoSQL-Datenbankmanagementsystem *MongoDB*. Zusätzlich wird die Bibliothek *mongoose* für das Object Data Modeling (ODM) verwendet.

2) Laufzeitumgebung: JavaScript-basierte Plattform Node.js mit dem serverseitigen Webframework ExpressJS.

III. Verteilungssicht

Das Spiel wird in einem Docker Container bei dem Cloud-Anbieter *Amazon Web Services* bereitgestellt.

Dieser Abschnitt beschreibt:

- die Verteilung des Gesamtsystems,
- wichtige Begründungen für diese Verteilungsstruktur,
- Qualitäts- und/oder Leistungsmerkmale dieser Infrastruktur,
- Zuordnung von Softwareartefakten zu Bestandteilen der Infrastruktur

IV. Querschnittliche Konzepte

Dieser Abschnitt beschreibt übergreifende, prinzipielle Regelungen und Lösungsansätze, die an mehreren Stellen relevant sind.

A. <Konzept 1>

<*Erklärung*>

B. <Konzept 2>

<*Erklärung*>

References

[1] Schmidt, B. (2020). Richtig zitieren: Eine Anleitung für Studierende (2. Aufl.). Springer.

[2] Erichsen, C. (2020, 17. Juli). Inklusion im Internet: So werden Social-Media-Inhalte barrierefrei. `t3n.https://t3n.de/magazin/inklusion-im-internet-so-werden-249553/`