Nautical Nonsense: Because Sometimes You Just Need to Sink Something

Jakob Götz
j.goetz@oth-aw.de

Uwe Kölbel
u.koelbel@oth-aw.de

Maximilian Schlosser
m.schlosser@oth-aw.de

Oliver Schmidts
o.schmidts@oth-aw.de

Jan Schuster
j.schuster@oth-aw.de

Philipp Seufert
p.seufert@oth-aw.de

Fabian Wagner
f.wagner@oth-aw.de

***Zusammenfassung*—Dieser Technical Report beschreibt die Architektur von dem Cloud Native Browser-Game *Nautical Nonsense*.**

I. Einleitung

Schiffeversenken ist seit je her ein beliebtes Spiel, egal, ob für jung oder alt. Hauptsächlich wird dieses Spiel jedoch als Brettspiel in Präsenz gespielt. Als 2019 die Corona-Pandemie startete, mussten zwangsweise die Kontakte reduziert werden. Dies ruf eine massive Veränderung in der Lebensweise der Menschen hervor. Besonders im Bereich der Unterhaltung entwickelten sich in kurzer Zeit viele Lösungen, um Aktivitäten, welche zuvor in Präsenz waren, in den Online-Raum zu migrieren. Inspiriert von dieser Entwicklung entstand die Idee, ein Cloud-Nativ Schiffeversenken zu erstellen.
In den weiteren Abschnitten dieses Dokuments wird auf die technischen Details des Vorhabens eingegangen. Abschnitt II enthält dabei die Vorgehensweise, welche sicherstellt, dass die Projektziele erreicht werden. Es folgt Abschnitt III, welches einen Überblick über die einzelnen Bausteine gibt und Abschnitt IV, worin die Entwicklungswerkzeuge erläutert werden. Abschließend wird in Abschnitt V auf das Fazit und den Ausblick eingegangen.

II. Vorgehensweise

III. Bausteinsicht

A. Datenbank

B. Backend

C. Frontend

1) Kommunikation mit Backend: Beim Aufrufen der Webseite sendet der Client einen HTTP-GET-Request an „localhost:8000“ und erhält ein JSON-Objekt mit der Client-ID. Diese ID wird der globalen Variable *sharedData.clientID* zugewiesen.

Die Websocket-Verbindungs-URL wird durch das Hinzufügen der Client-ID zu *sharedData.websocket_url* erstellt.

Wenn der Client in der Start-Szene „Spiel gegen Random“ auswählt, erfolgt ein HTTP-POST-Request an „localhost:8000/play“ mit Informationen wie der Client-ID, dem ausgewählten Modus und dem Spielernamen. Die empfangene Game-ID wird in *sharedData.game_id* gespeichert.

Die Websocket-Verbindung wird erst nach erfolgreicher Ausführung dieses Befehls initialisiert, bevor zur Szene „Waiting1“ gewechselt wird. Dieser Websocket existiert ebenfalls als globale Variable namens *sharedData.socket*.

In allen Szenen ist die Methode *onmessage* implementiert, die es ermöglicht, über Websockets eingehende Nachrichten vom Backend zu empfangen. Diese Methode ist Bestandteil des JavaScript-Objekts *WebSocket*.

Wenn sich ein zweiter Client auf der Webseite verbindet und die Option „Spiele gegen Random“ auswählt, sendet das Backend über die Websocket-Verbindung ein „ready“-Flag an beide Clients. In Folge dessen wird die Variable *sharedData.ready* auf den Wert „true“ gesetzt und es erfolgt ein nahtloser Wechsel zur Szene „Shipplacement“.

D. Infrastruktur

IV. Entwicklungswerkzeuge

A. Backend

B. Frontend

V. Fazit und Ausblick