Meilenstein 3: «Schiffe versenken»

Autorin: Jetsün Künsel Emchi

Repository: https://github.com/Kemiem/Schiffe-versenken-ms3/tree/main/schiffe-versenken-main

Datum: 19.11.2025

1. Ziel des Meilensteins

Ziel dieses Meilensteins war es eine vollständig spielbare Version von «Schiffe versenken» zu implementieren.

Im Mittelpunkt bei der Umsetzung standen folgende Punkte:

- die Verwaltung des Spielstatus auf dem Server
- die Umsetzung der korrekten Spiellogik
    - Treffer
    - Verfehlung
    - Sieg
- die Möglichkeit von alternierenden Spielzügen
- Echtzeitkommunikation
- Erstellung einer spielbaren Benutzeroberfläche
- Funktionierender Lobby-Chat

2. Architekturübersicht

2.1. Server

Der Server basiert auf *Node.js* und *Socket.IO* und übernimmt die Benutzerverwaltung inklusive Logins, Erstellen und Verwalten von Spielen und die Trefferlogik. Zusätzlich synchronisiert er den Spielzustand für beide Clients, stellt den Lobby-Chat bereit und kümmert sich um Fehler- und Disconnect-Handling.

2.2. Client

Der Client, umgesetzt mit *React* und *TypeScript*, stellt die Login-Oberfläche bereit. Er zeigt eigene und gegnerische Spielfelder an und visualisiert Treffer und Fehlschüsse. Zudem informiert er über den aktuellen Zug und präsentiert die Gewinner: in. Zudem bietet er eine Chat-Oberfläche. Die Darstellung leitet er aus dem vom Server übermittelten «game.state» ab.

3. Verwaltung des Spielstatus

Der Server speichert den Spielstatus im Objekt «currentGame».

Das Objekt enthält:

- Spieler:in-IDs
- 8x8 Boards pro Spieler:in
- Phasen (“waiting”, “playing” und “finished”)
- aktuelle Spieler:in am Zug
- Gewinner:in
- Lister aller Schüsse

4. Ablauf einer Partie

Name	Ablauf
Login	- Client sendet « client.login.senden » - Server validiert den Namen - Server sendet « server.login.ok»
Spielstart	- Nach Login von zwei Personen, startet das Spiel - Beide erhalten ihr Board - «Playing» Phase - Start erster Zug - «game.state» wird gesendet

Schuss	- Client sender «game.shoot {x, y}
	- Server prüft ob:
	  ○ ein Spiel vorhanden ist
	  ○ die richtige Phase stattfindet
	  ○ die Spieler:in zum Spiel gehört
	  ○ die Spieler:in am Zug ist
	  ○ die Koordinaten gültig sind
	  ○ das Feld unbelegt ist
	- der Server setzt Treffer/Miss, wechselt den Zug und sendet den neuen «game.state»

Sieg	- wenn alle Schiffszellen der Gegner:in getroffen wurden wird:
	  ○ Phase auf «finished» gesetzt
	  ○ Gewinner:in wird gesetzt

	○ «game.state» wird gesendet
Disconnect	- Wenn eine Spieler:in das Spiel verlässt: ○ gewinnt die andere Spieler:in das Spiel ○ wird der finale «game.state» gesendet

5. GUI 2. Entwurf

Die Benutzeroberfläche enthält:

- eigenes Feld mit Schiffen und gegnerischen Treffern
- Gegnerfeld mit eigenen Schüssen
- Anzeige, wer am Zug ist
- Anzeige der Gewinner:in
- Lobby-Chat
- Responsives Layout

6. Änderungen seit «Meilenstein 2»

6.1. Serverseitig

- komplette Spiellogik
- Boards + Demo-Flotte
- Trefferprüfung und Spielzugwechsel
- Sieglogik
- Fehlerbehandlung
- zentrales Event «game.state»
- Disconnect-Behandlung

6.2. Clientseitig

- Rendering der Spielfelder
- Visualisierung von «Hit/Miss»
- Gewinneranzeige
- Finalisierung chat

6.3. Dokumentation

- Überarbeitung Kommunikationsprotokoll
- Erstellung «Meilenstein-3» Dokument