

Meilenstein 4: «Schiffe versenken»

Autorin: Jetsün Künsel Emchi

Repository: <https://github.com/Kemiem/schiffe-versenken-ms4.git>

Commit Hash: 383b962113760f792d3498f6070811ef798e447a

Datum: 18.12.2025

1. Ziel und Zweck

Das Ziel von «Meilenstein 4» ist es, ein feature-komplettes und stabiles Multiplayer-Spiel zu schaffen.

Der Server agiert dabei als «Source of Truth», während der Client eine gefilterte Sicht auf den Spielzustand darstellt. Die relevanten Aktionen werden validiert und in Echtzeit synchronisiert.

2. Serverseitige Spiellogik

Der Spielzustand wird serverseitig im Memory verwaltet. Dazu gehören die eingeloggten Benutzer:innen, das aktuell laufende Spiel sowie alle spielrelevanten Informationen wie Boards, Schüsse, Zugreihenfolge und Spielphase. Das Spiel kennt die Phasen «waiting», «playing» und «finished». Sobald zwei Spieler:innen eingeloggt sind und noch kein Spiel läuft, wird automatisch eine neue Partie gestartet. Die Generierung der Spielfelder (Boards) erfolgt ausschliesslich serverseitig, um eine clientseitige Manipulation zu verhindern.

Die Spiellogik wird vollständig auf dem Server ausgeführt. Bei jedem Spielzug wird überprüft, ob ein aktives Spiel existiert, ob sich das Spiel in der korrekten Phase befindet, ob die Spieler:in Teil des Spiels ist und ob sie an der Reihe ist. Zusätzlich werden die Koordinaten der abgegebenen Schüsse auf ihre Gültigkeit geprüft, um mehrfaches Beschiessen desselben Feldes auszuschliessen. Bei ungültigen Aktionen werden Fehlermeldungen inklusive Fehlercodes zurückgegeben. Nach jedem gültigen Zug wird der aktualisierte Spielzustand an alle verbundenen Clients übertragen.

3. GameState (Sichtschutz)

Um eine sichere Übertragung des Spielzustands an die Clients zu gewährleisten, wird der interne GameState serverseitig in eine reduzierte Client-Ansicht umgewandelt. Die jeweilige Spieler:in sieht die eigenen Schiffe inklusive Trefferstatus, die eigenen Schüsse auf das gegnerische Feld sowie die Schüsse der Gegner:in auf das eigene Board. Informationen über die Positionen gegnerischer Schiffe werden bewusst nicht an den Client übermittelt. Metadaten wie aktuelle Spielphase, Zugrecht, Gegner:in (Name und ID) sowie eine allfällige Gewinner:in werden jedoch übertragen. Dadurch wird sichergestellt, dass jeder Client nur die für ihn bestimmten Informationen erhält.

4. Spielende, Neustart und Sonderfälle

Das Spiel endet, sobald alle Schiffe einer Seite getroffen wurden. In diesem Fall wird die Gewinner:in serverseitig gesetzt und der finale Spielzustand an alle Clients gesendet. Nach einer kurzen Pause wird das Spiel automatisch zurückgesetzt, sodass ohne manuelles Eingreifen eine neue Partie gestartet werden kann. Verlässt eine Spieler:in das Spiel während der Spielphase, gewinnt die verbleibende Spieler:in automatisch.

5. Lobby und Echtzeit-Kommunikation

Neben der Spiellogik wurde auch die Lobby-Funktionalität erweitert. Der Login prüft auf leere, zu lange oder bereits vergebene Nicknames. Änderungen in der Benutzerliste werden in Echtzeit an alle Clients übertragen. Zusätzlich steht ein Lobby-Chat zur Verfügung, der es eingeloggten Benutzer:innen erlaubt, in Echtzeit miteinander zu kommunizieren. Auch hier werden einfache Validierungen durchgeführt.

6. Tests und Qualitätssicherung

Die Funktionalität des Systems wurde anhand verschiedener Szenarien manuell getestet. Dazu zählen der reguläre Spielablauf von Spielstart bis Spielende, ungültige Spielzüge wie Schüsse ausserhalb des Spielfelds oder ausserhalb des eigenen Zuges, doppelte Schüsse auf dasselbe Feld sowie Verbindungsabbrüche während eines laufenden Spiels. In allen getesteten Fällen verhielt sich das System stabil und der Spielzustand blieb konsistent.

7. Bekannte Limitationen und Ausblick

Nicht Bestandteil von Meilenstein 4 sind eine persistente Speicherung des Spielzustands sowie ein stabiler Rejoin nach einem Seitenreload, da die Identifikation aktuell über die Socket-ID erfolgt. Ebenso sind umfassende Zod-Schemas für alle Events und Payloads vorbereitet, jedoch noch nicht vollständig integriert. Diese Punkte stellen mögliche Erweiterungen für einen späteren Ausbauschritt dar.

8. Fazit

Nach Abschluss von Meilenstein 4 befindet sich das Projekt in einem spielbaren, stabilen und feature-kompletten Zustand. Die Kernanforderungen eines serverautoritativen Multiplayer-Spiels mit Echtzeit-Kommunikation wurden erfolgreich umgesetzt. Die bestehende Architektur bildet eine solide Grundlage für zukünftige Erweiterungen wie Persistenz, Rejoin-Funktionalität und weitergehende Validierung.