IMPFTERMINREGISTRIERUNGSANWENDUNG

Hossein Akbari Informatik B.Sc.
Fachbereich II – Management und Informationssysteme
https://informatik.hs-bremerhaven.de/docker-swe3-2024-14-java/

PROJEKTÜBERBLICK & MOTIVATION

Die Impfterminregistrierungsanwendung wurde im Rahmen des Moduls Software Engineering III an der Hochschule Bremerhaven entwickelt, um die IT-Infrastruktur in Zeiten einer Pandemie zukunftssicher zu gestalten. Ziel des Projekts ist es, eine skalierbare, robuste und datensparsame Webanwendung zu realisieren, über die sich Bürger einer Großstadt - exemparisch Nemerb - schnell und unkompliziert für Impftermine registrieren und buchen können.

Angesichts der Erfahrungen während der Corona-Pandemit, bei denen plötzlich hohe Anmeldezahlen die Systeme überlastet haben, stand die Herausforderung im Vordergrund, ein System zu entwerfen, das auch unter extremen Lastbedingungen stabil bleibt. Besonders wichtig war es, den gesamten Prozess - von der Registrierung über die Terminbuchung bis hin zur Verwaltung der Termine - so zu gestalten, dass er auch in Zeiten großer Nachfrage zuverlässig funktioniert und eine schnelle Reaktionszeit bietet.

Das System zeichnet sich durch eine klare und intuitive Benutzeroberfläche aus, die es sowohl Einzelpersonen als auch Gruppen ermöglicht, Termine zu buchen - sei es für sich selbst oder für Familienmitglieder und Freunde. Neben der Benutzerfreundlichkeit legt das Projekt großen Wert auf den Schutz persönlicher Daten, sodass die Privatsphäre der Bürger stets gewahrt bleibt. Darüber hinaus zeigt die Lösung, dass es möglich ist, ein stabiles und anpassungsfähiges System zu entwickeln, das in Krisenzeiten den hohen Anforderungen einer Großstadt gerecht wird und zugleich einfach zu bedienen ist.

Diese Anwendung zeigt, dass ein gutes Design und eine sorgfältige Planung helfen, die Herausforderungen einer modernen, stark genutzten Webanwendung zu meistern - so ist man im Ernstfall schnell und effizient bereit.

SYSTEMARCHITEKTUR

Die Impfterminregistrierungsanwendung ist als zweiteiliges System konzipiert, bestehend aus einem komponentenbasierten Frontend und einer mehrschichtigen Backend-Struktur. Im Frontend wird eine Single Page Application (SPA) eingesetzt, bei der die Hauptdateien (main.js, fetch.js, bookingPage.js und adminPage.js klar voneinander abgegrenzt sind. So steuert main.js die allgemeine Navigation und Initialisierung, bookingPage.js verwaltet die Buchungslogik, während adminPage.js spezielle Verwaltungsfunktionen für Administratoren bereitstellt. Die Kommunikation mit dem Server erfolgt unter anderem dabei über fetch.js, das alle HTTP-Anfragen seines des Benutzers bündelt und die JSON-Antworten an die jeweiligen Module weiterleitet.

Im Backend setzen wir auf eine mehrschichtige Architektur: Servlets (z.B. LoginServlet, RegisterServlet, SlotsServlet) bilden die API-Schicht und nehmen HTTP-Anfragen entgegen. Diese leiten sie an die Service-Schicht (etwa AppointmentService, UserService) weiter, wo die eigentliche Geschäftslogik ausgeführt wird - inklusive Transaktionsverwaltung, Datenvalidierung und Sicherheitsaspekten. Utility-Klassen (z.B. DataBaseUtil, AppointmentUtil) und Data Transfer Objects (DTOs) unterstützen die Services bei Datenbankzugriffen sowie der strukturierten Datenübertragung. ZUletzt speichert eine relationale MariaDB-Datenbank alle relevanten Informationen (Benutzerdaten, Termine, Impfstoffbestände).

Durch diese modulare und klar strukturierte Architektur wird gewährelistet, dass die Impfterminregistrierungsanwendung nicht nur den aktuellen Anforderungen an Skalierbarkeit, Wartbarkeit und Sicherheit gerecht wird, sondern auch flexibel genug ist, um zukünftigen Herausforderungen standzuhalten. Die Trennung im Frontend und Backend, unterstützt durch spezialisierte Komponenten und Schichten, ermöglicht eine effiziente Entwicklung und einfache Erweiterbarkeit. Damit bildet dieses System eine gute Grundlage, um auch in Kristenzeiten einen reibungslosen Ablauf zu garantieren und eine zuverlässige Servicebereitstellung für alle Nutzer sicherzustellen.

ARCHITEKTURÜBERSICHT

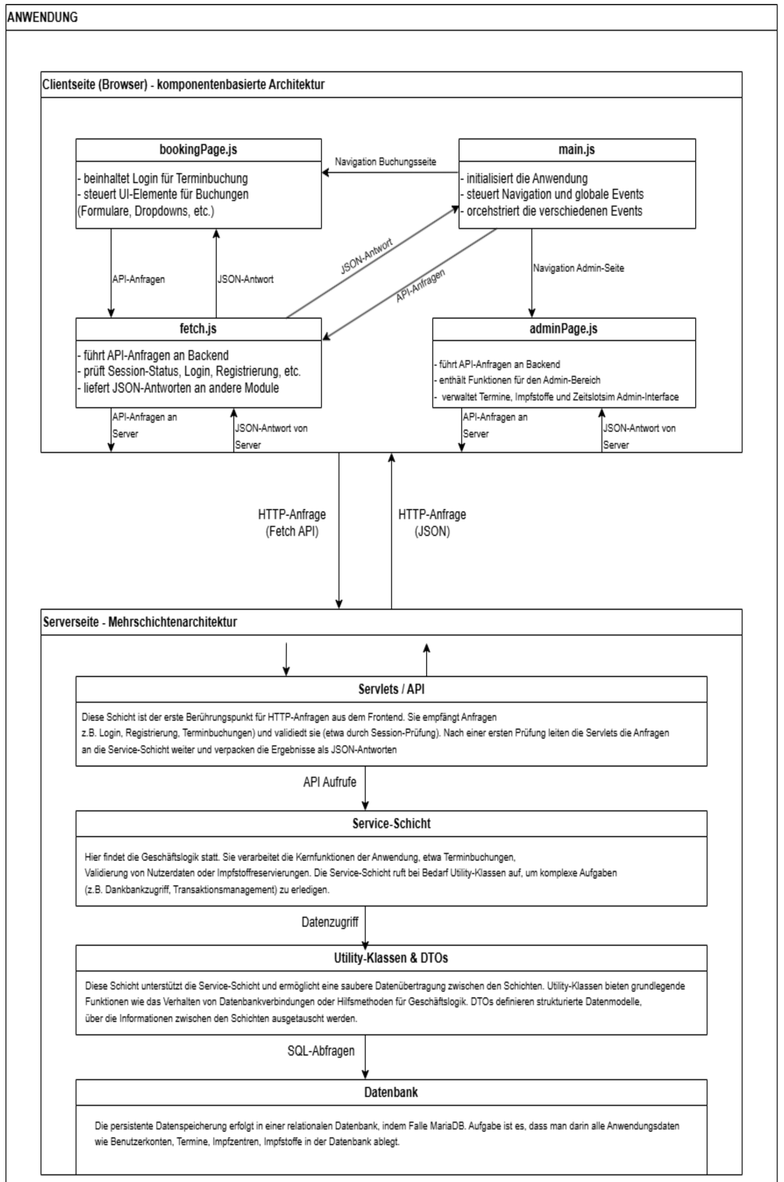

Abbildung 1. Frontend-Backend-Architektur[1]

Performance und Skalierung

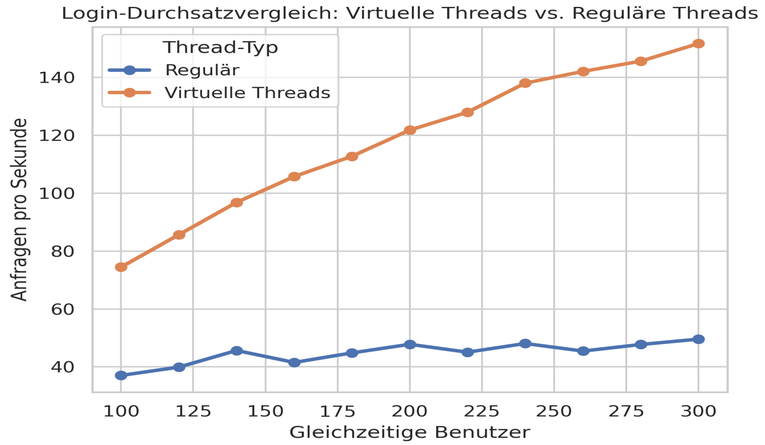

Abbildung 2. Performancevergleich für Login: Virtuelle Threads vs. klassische Threads

Für die Performance- und Skalierungstestes der Impfterminregistrierungsanwendung wurde gezielt der Login-Prozess untersucht. Dabei wurden realistische Bedingungen simuliert, indem zunächst eine Datenbanklatenz von etwa 400-500 ms eingestellt worden ist, um die tatsächlichen Herausforderungen in einer Krisensituation abzubilden. Ziel war es, den Unterschied zwischen einer herkömmlichen Implementierung, die reguläre Threads verwendet, und einer modernen Implementierung mit virtuellen Threads zu evaluieren. Hierzu wurde schrittweise die Anzahl der gleichzeitig aktiven Benutzer (100, 200 und 300 Benutzer) erhöht und der Durchsatz in Afragen pro Sekunde bemessen. Die Ergebnisse zeigen, dass die moderne Variante mit virtuellen Threads deutlich besser skaliert: Bereits bei 100 Benutzern liegt der Durchsatz nahezu doppelt so hoch wie bei der herkömmlichen Methode, und dieser Unterschied wird mit steigender Benutzeranzahl noch ausgeprägter. Die folgende Tabelle fasst die wesentlichen Ergebnisse zusammen:

Benutzeranzahl	Traditionelle Impl.	Moderne Impl.	Steigerung
100	37,05	74,46	2,01x
200	47,80	121,80	2,55x
300	49,59	151,67	3,06x

Diese Daten deuten darauf hin, dass die moderne Implementierung nicht nur den Durchsatz signifikant erhöht, sondern auch in Bezug auf Skalierbarkeit und Ressourcennutzung überlegen ist. Besonders unter hoher Last werden mehr Anfragen parallel verarbeitet, was zu einer deutlich schnelleren Reaktionszeit führt. Diese Vorteile sind in Szenarien mit hoher gleichzeitiger Nutzung, wie sie bei Impfterminregistrierungen in Krisenzeiten auftreten, von entscheidender Bedeutung. Zusammengefasst belegen die Ergebnisse, dass der Einsatz von virtuellen Threads die Leistungsfähigkeit der Anwendung erheblich steigert und eine robuste, skalierbare Infrastruktur ermöglicht - ein wesentlicher Erfolgsfaktor, um in kritischen Situationen eine reibungslose Servicebereitstellung sicherzustellen.

LITERATURVERZEICHNIS

[1] Philip Ackermann. Webentwicklung - Das Handbuch für Fullstack-Entwickler. Rheinwerk Computing, Bonn, 2021. ISBN 978-3-836-26884-4.