Hochschule der Medien

Nobelstraße 10

70569 Stuttgart

Wintersemester 2022/2023

Dokumentation
OpenBarber

Betreuer: Prof. Walter Kriha

Gruppe: Vu, Son Hai (sv048)

Redinger, David (dr080)

Seitz, Dominik (ds189)

Flocken, Tom (tf054)

Vidovic, Kristian (kv021)

Abgabeort und –datum: Stuttgart, 28.02.2023

Inhaltsverzeichnis

1 Einleitung

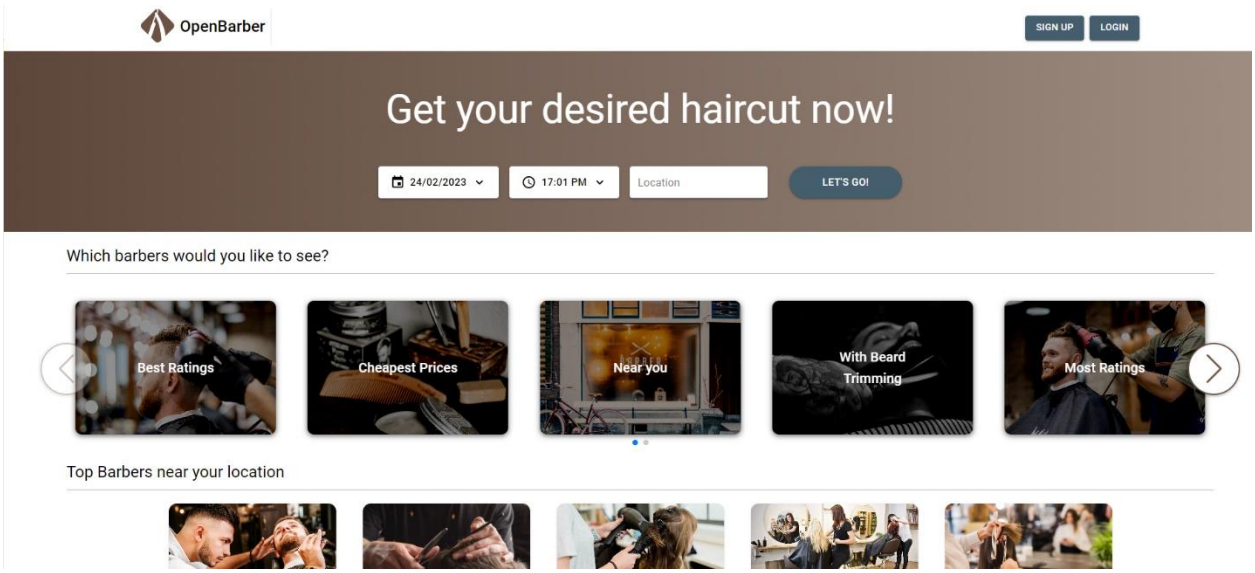
Bei unserem Projekt „OpenBarber“ handelt es sich um eine webbasierte Anwendung die einerseits für die Verwaltung von Reservierungen für den Friseurbereich fungiert aber auch andererseits um eine Möglichkeit für Endkunden das Angebot an registrierten Friseuren zu durchsuchen und einen Termin zu buchen.

Ziel und Zweck des Projekts ist die Vertiefung der Kenntnisse im Bereich der Softwareentwicklung. Dabei stellt das Projekt „ObenBarber“ nur ein Fallbeispiel dar, da wir darauf achten wollen wie man diese Idee unabhängig von einem Thema, wie der Friseurbereich, als Template Plattform hätte aufbauen kann.

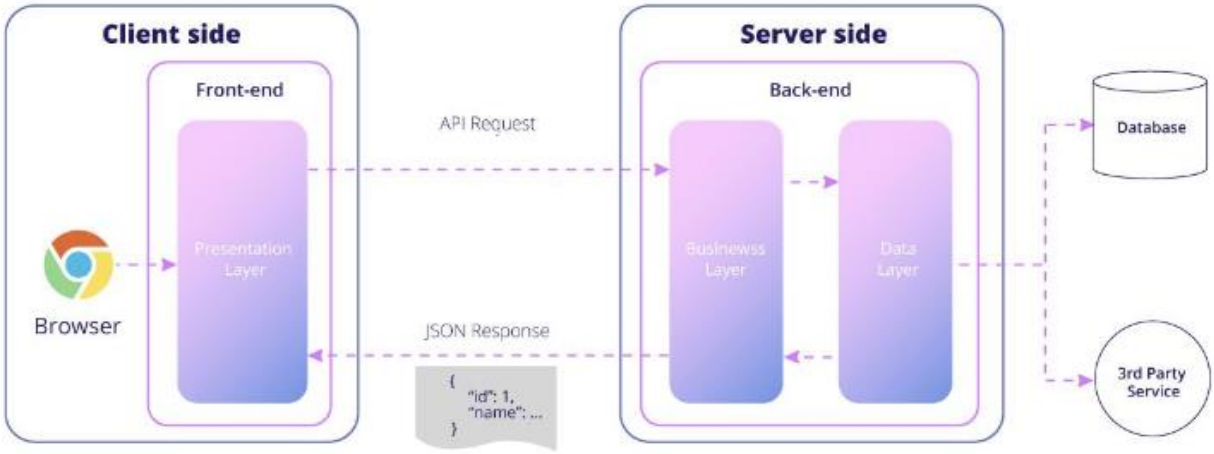
Der Zeitrahmen beträgt ca. vier Monate. Ein Prototyp der Anwendung soll dann am 26.01.2023 an der MediaNight präsentiert werden. Dazu gehört ein Pitch am MI-Präsentationstag sowie der Auf- und Abbau des Standes am Tag der MediaNight selbst.

Unsere Gruppe besteht aus vier Bachelor MI / MM Studenten in verschiedenen Fachsemestern. Dazu reiht sich ein CSM Masterstudent als Scrum Master und Projekt Manager im Rahmen seines Moduls mit ein.

OpenBarber ist ein Reservierungsmanagementsystem speziell für Friseure.
Endkunden können nach Friseuren in bestimmten Städten suchen und bei Bedarf einen Termin reservieren. Die Friseure werden über Reservierungen informiert.
Zudem können Friseure kontaktiert, bewertet und kommentiert werden.



2 Architektur & Funktion



Wir haben uns für eine Single Page Application Architektur entschieden. Diese ermöglicht es uns eine interaktive Web App zu implementieren die via API mit dem Backend kommunizieren kann. Diese Architektur zeichnet sich unteranderem durch eine gute Skalierbarkeit aus. Wenn zukünftig eine mobile App benötigt wird, sind keine zusätzlichen Anstrengungen für die API-Entwicklung erforderlich – die mobile App könnte dieselbe API wie die Webanwendung verwenden. Da es sich um eine eher kostspielige Art von Webarchitektur handelt eignet sich SPA gut zur Erstellung einer responsive Benutzeroberfläche für B2C-Benutzer.

Unsere Business-Logik wird zentral im Backend ausgeführt und ist so von der Client Side unabhängig und nicht manipulierbar. Backend, Frontend und Datenbank lassen sich dank Container-Technologie auch auf CI/CD Pipelines deployen.

Wir haben uns im Rahmen dieses Projektes für eine relationale Datenbank entschieden. Da sich das Thema, Friseure & Reservierungen, hervorragend mit Relationen bzw. Entities mappen lässt.

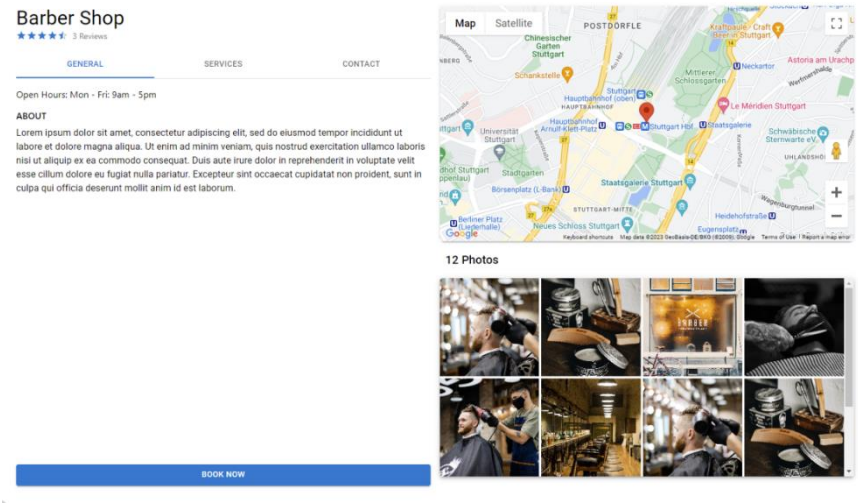
3rd Party Services nutzen wir unter anderem, um Mails zu senden & responsive HTML-Code zu erzeugen. Dabei haben wir einen besonderen Fokus daraufgelegt, dass die Services sich für Templates eignen.

3 Eingesetzte Technologien

Die Technologien, die wir im Projekt angewendet haben, bestehen aus React.js für das Frontend, Spring im Backend, sowie eine PostgreSQL Datenbank.

3.1 React.js

Das Frontend Framework React liefert uns die Option Component Libraries zu nutzen. „Material UI“ ist dabei unsere Wahl, da sehr viele Endnutzer mit dem Material UI Design vertraut sind. Außerdem nutzen wir auch die Google Maps API um Shops zu lokalisieren und auf einer Karte darstellen zu können. Die Projektstruktur im Frontend teilt sich in Assets, Components, Pages, CSS, Layout und Mocks auf.



3.2 Spring

Spring Boot bringt von sich aus viele Packages mit die wir im Backend brauchen. Rest Controller, Mail Sender Service, JWT Authentication und Datenbank Anbindung sind daher verpflichtend zu implementieren in unserem Projekt. Die Projektstruktur im Backend teilt sich in Services, Controller, Repository, (Mail-)Templates, Models und Configs auf.

Um unseren Anspruch an eine Template Engine für E-Mails zu gewährleisten, haben wir uns für MJML entschieden. Diese HTML-erzeugende Template Engine hilft uns dabei responsive HTML zu generieren und an den entscheidenden Stellen Placeholder zu setzen. Zum Beispiel: Ein Verifikation Code via Mail. MJML bietet den Vorteil sich

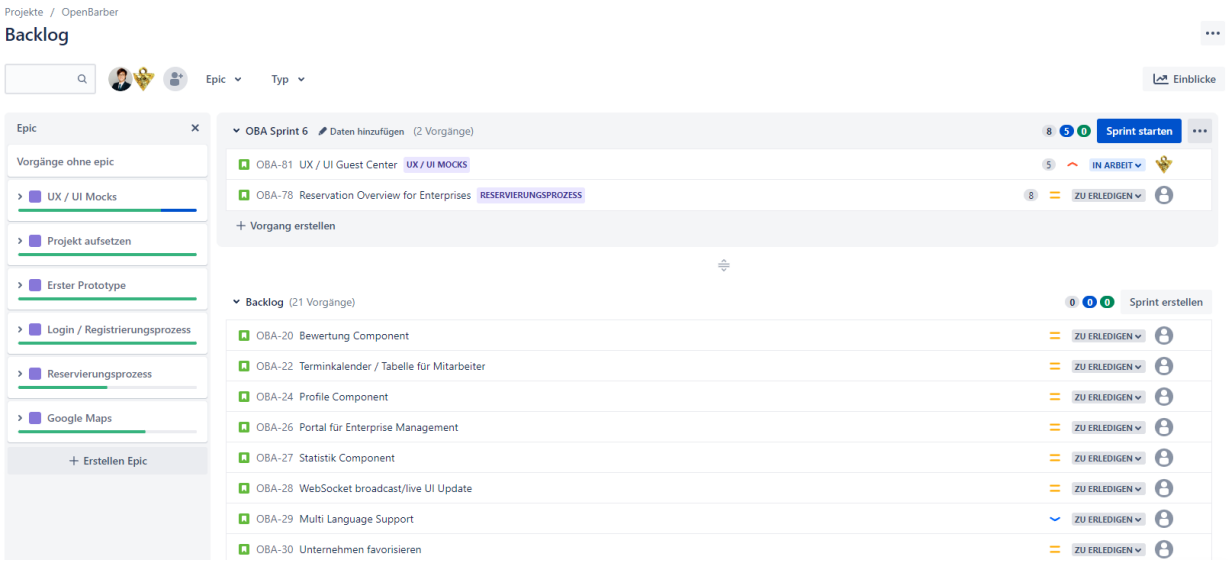
nicht mehr um das Aussehen der Mail in allen E-Mail-Clients spezifisch kümmern zu müssen. Da jeder E-Mail-Client Anbieter (zum Beispiel: Gmail, Outlook, …) die einkommende HTML-Datei selbst individuell rendert, ist es eine große Herausforderung eine Mail auf verschiedenen Endgeräten und E-Mail-Clients einheitlich anzeigen zu lassen.

3.3 PostgreSQL

Eine relationale Datenbank hilft uns dabei strukturiert und persistent unsere Daten zu speichern. Wir haben früh gemerkt, dass unser Projekt Thema nicht simple als Datenbank-Model widergespiegelt werden kann da sehr viele Relationen entstehen können. PostgreSQL ist eine Open Source Software und besonders gut erweiterbar in ihrer Funktionalität.

4 Arbeitsmethodik

Wir haben nach Scrum gearbeitet. Das bedeutet, dass wir wöchentliche Meetings durchgeführt haben, um uns auszutauschen und den aktuellen Stand festzuhalten. Zur Dokumentation haben wir das Projektmanagement-Tool JIRA herangezogen. Zunächst haben wir uns zusammengesetzt und die Anforderungen gesammelt, was das Projekt beinhalten soll. Diese haben wir gegliedert in Must-have, Should-have und Nice-to-have. Diese Anforderungen haben wir dann in unser Backlog auf JIRA hinzugefügt. Unseren Backlog haben wir dann nach und nach während des Projekts erweitert, was der agilen Softwaremethodik entspricht. So konnten wir die Aufgaben gut einteilen und jeder wusste, was seine Aufgabe war. Die ganzen Anforderungen haben wir wiederum in kleineren Tasks untergliedert. Das bedeutet, dass man in der Regel eine größere Anforderung bzw. ein größeres Feature zu implementieren hat, sogenannte User Storys, die dann aus mehreren kleineren Tasks bestehen. Wir haben alle Elemente von Scrum implementiert, d.h. Sprint-Reviews, Sprint-Planning und Sprint-Retrospektiven durchgeführt. Die User Storys wurden ebenfalls gemeinsam anhand eines Planning Pokers geschätzt.



5 Fazit

Unserer Einschätzung nach könnte man diese Projektidee mit etwas mehr Aufwand in eine Template Reservierungs- und Dienstleistungsmanagement Plattform umwandeln. Da der Content dieser Plattform nicht von uns als Anbieter generiert wird, sondern die Dienstleister selbst für ihr Angebot verantwortlich sind, kann man sich gut vom eigentlichen Plattform-Thema lösen. Während der Implementierung haben wir auch darauf geachtet Technologien, Libraries und Tools zu nutzen die das Erweitern unseres Projektes auf eine universelle Template-Plattform gut ermöglichen. Registrierung, Login, Reservierungen, Filtern von Dienstleistungen und Managementtätigkeiten sind alles Prozesse die nicht nur auf ein Thema passen.

Durch das Projekt wurden Kenntnisse in der Softwareentwicklung weiter vertieft. Wir konnten unser erworbenes Wissen hier praktisch umsetzen. Dazu gehören Planung und Konzeption, sowie ein vernünftiges Zeit- und Ressourcenmanagement. Wir haben versucht, so gut es geht, das Projekt unter realen Bedingungen durchzuführen, wie es in der Praxis üblich ist. So haben wir viel Wert auf sauberen Code, Projektmanagement und Design Patterns gelegt. Auch haben wir Pair Programming betrieben, was sich ebenfalls als sehr effizient bewiesen hat.

Es klappte nicht alles sofort auf Anhieb und vieles musste selbständig recherchiert werden, dafür ist jedoch die Lernkurve sehr steil und man hat durch das Projekt viel Neues gelernt.
Zwischendurch gab es immer wieder Schwierigkeiten und Blocker, die den Workflow behinderten. Beispielsweise die Implementierung des eigentlichen Reservierungsmanagementsystem war schwieriger als zuerst angenommen.
Das Arbeiten im Team erfordert eine gute Koordination. So hatten wir hin und wieder Merge-Konflikte gehabt und andere Meinungsverschiedenheiten, die aber zum Glück immer gelöst werden konnten. Zudem hängen einige Features von anderen Features ab, die wiederum von anderen Teammitgliedern bearbeitet werden. Aus diesem Grund musste man auf das Teammitglied warten bzw. man schaut sich gemeinsam den Code an und hilft sich gegenseitig.

Das Projekt hatte fast alle Bereiche der Webentwicklung abgedeckt (Fullstack). Dazu gehören die Implementierungen im Backend und im Frontend, sowie die dazugehörigen E2E-Tests und Unit-Tests und natürlich auch das Verfassen der Dokumentation.
Von der Entwicklung bis zum Deployment war alles dabei.
Das Projekt hat uns gezeigt, dass wir in der Lage waren, innerhalb einer gegebenen Zeit, eine fertige Anwendung zu entwickeln.

Zusammenfassend kann man sagen, dass so ein Software-Projekt eine sehr große und wertvolle Erfahrung war. Vor allem aber hat es auch sehr viel Spaß gemacht und man ist am Ende doch sehr stolz diese Anwendung auf der MediaNight präsentieren zu dürfen.
Vielleicht bietet sich noch die Gelegenheit an, unsere Anwendung in Zukunft zu vermarkten. Wir sehen da auf jeden Fall Potential.