Handout – Dinder

Tobias Raible, Lukas Schulz, Jannis Fuchs, Jonathan Schäfer, Manuel Franz

---

Wir haben uns für die Entwicklung einer Webanwendung entschieden, deren Hauptziel es ist, die Vermittlung verschiedener Dienstleistungen zu erleichtern. Unter dem Namen Dinder präsentieren wir unser Projekt, das als eine Art "Tinder für Dienstleistungen" konzipiert ist.

Wir wollen mit Dinder eine neue Möglichkeit bieten, die Art und Weise zu revolutionieren, wie Menschen Dienstleistungen finden und anbieten.

Inhalt

Aufgabenverteilung

- Jonathan Schäfer hat sich die beiden Semester mit der Backend Implementierung befasst, sowie gegen Ende des vierten Semesters mit SonarCloud, CI/CD und der Testorganisation.
- Tobias Raible hat sich die Zeit über ebenfalls mit der Implementierung des Backends beschäftigt.
- Jannis Fuchs hat sich hauptsächlich die beiden Semester mit dem Frontend beschäftigt.
- Manuel Franz hat sich hauptsächlich um den Datenbankentwurf im ersten Semester gekümmert. Zusätzlich hat er sich auch um das Projektmanagement (Youtrack, Scrum), sowie im zweiten Semester auch um Teile des Frontends befasst, da Lukas nicht mehr Teil dieses Bereichs war.
- Lukas Schulz war nur im dritten Semester Teil dieses Projekts, da er danach das Studium verlassen hat. In den drei Monaten hat er sich hauptsächlich mit der Bearbeitung des Frontends beschäftigt.

Zeitmanagement

Das ganze Projekt haben wir in insgesamt neun Sprints mit je zwei Wochen aufgeteilt. Die Arbeitsstunden pro Person sind in der folgenden Tabelle dargestellt (Stand 10.06.2024; 19 Uhr):

	Gesamt	**Drittes Semester**	**Viertes Semester**
Lukas Schulz	18	18	-
Jonathan Schäfer	43,75	18.3	25,45
Tobias Raible	46.75	16.5	30.25
Jannis Fuchs	52.75	20	32.75
Manuel Franz	57.5	22.3	35.2
Gesamtzeit	218.75	95.1	123.65

Tabelle 1: Zeitüberblick

In der Abbildung 1 (Zeitmanagement) sind die summierten Zeiten der Teilnehmer über die Sprints verteilt dargestellt. Zur besseren Orientierung sind auch die Durchschnittswerte der gesamten Sprints angegeben (blaue Linie).

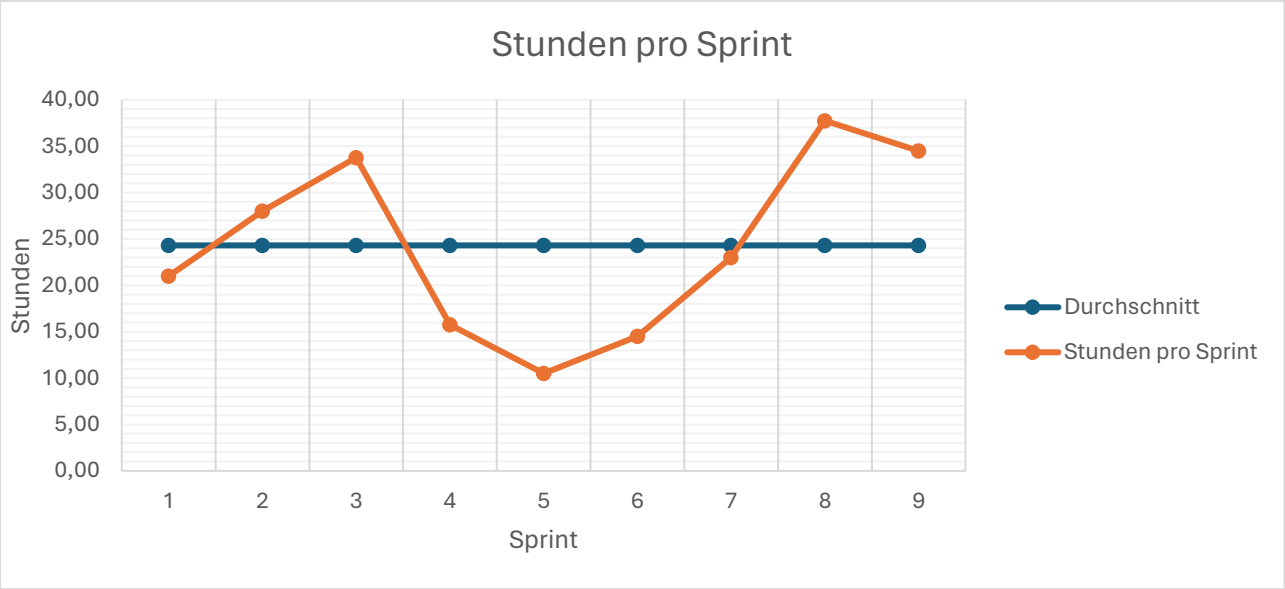


Abbildung 1: Zeitmanagement - Sprints

In der folgenden Grafik sind die Arbeitsstunden pro Phase dargestellt. Hierbei wurde die gesamte Projektzeit in vier Teilabschnitte untergliedert: Planung (Sprint 1 & 2), Erste Implementierung und Planungsabschluss (Sprint 3 & 4); Fortgeschrittene Implementierung und Neuordnung des Teams (Sprint 5 - 7), sowie die Test- und Projektfinalisierungsphase (Sprint 8 & 9).

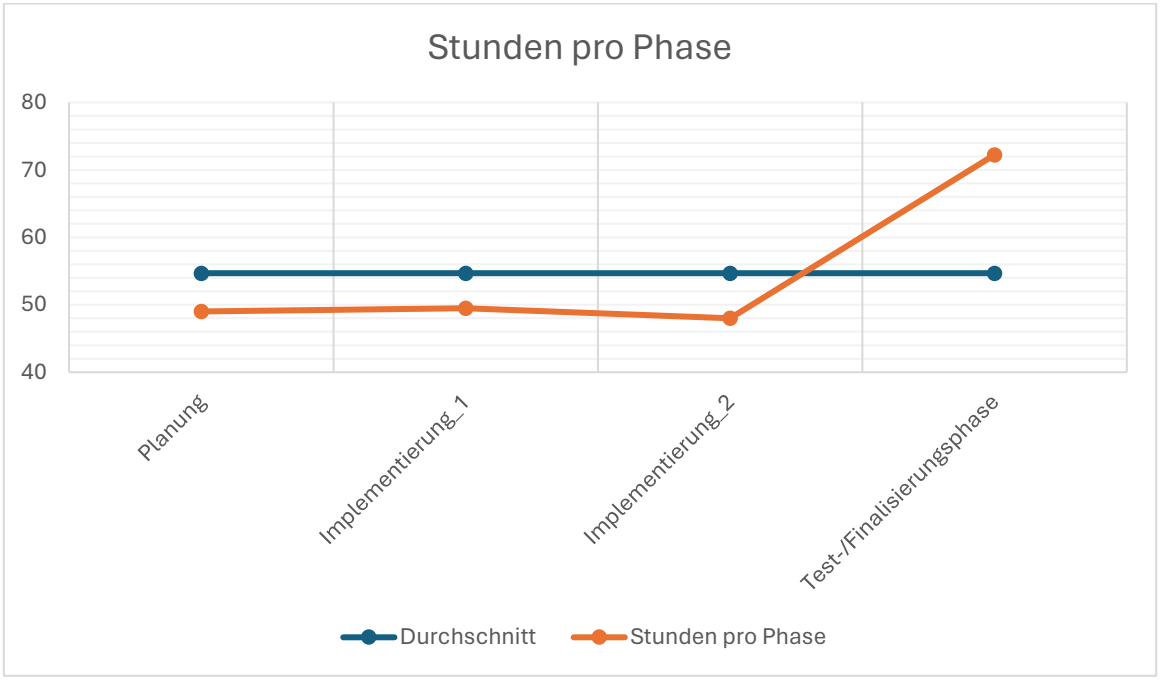


Abbildung 2: Zeitmanagement - Phasen

Demo

Da unsere Anwendung die Nutzung von verschiedenen Accounts erfordert, gibt es eine Anmelde- und Registrierungsfunktion.

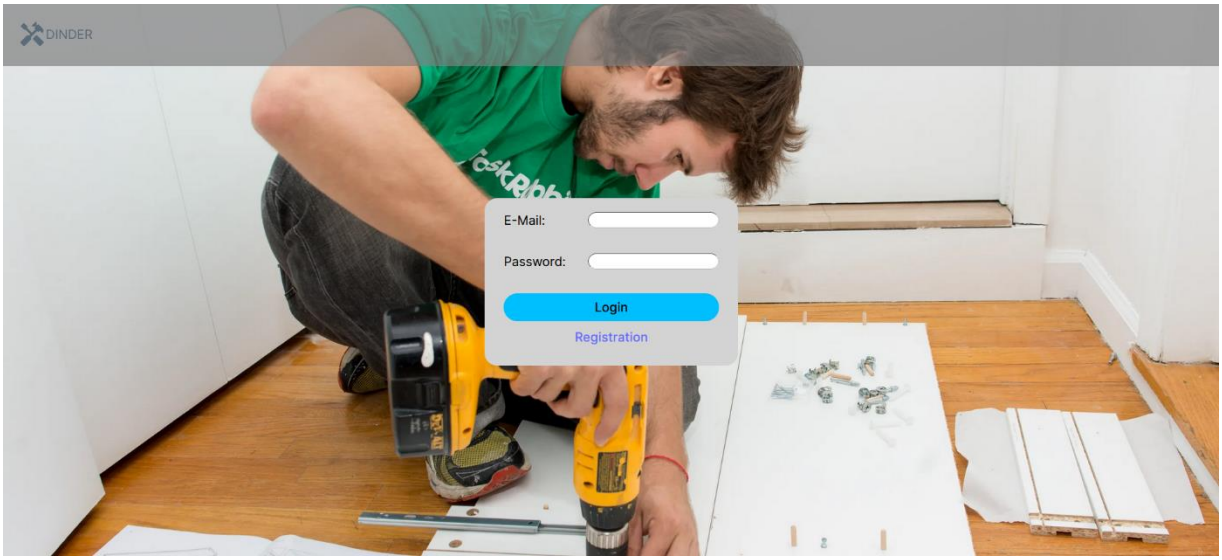


Abbildung 3: Demo - Login

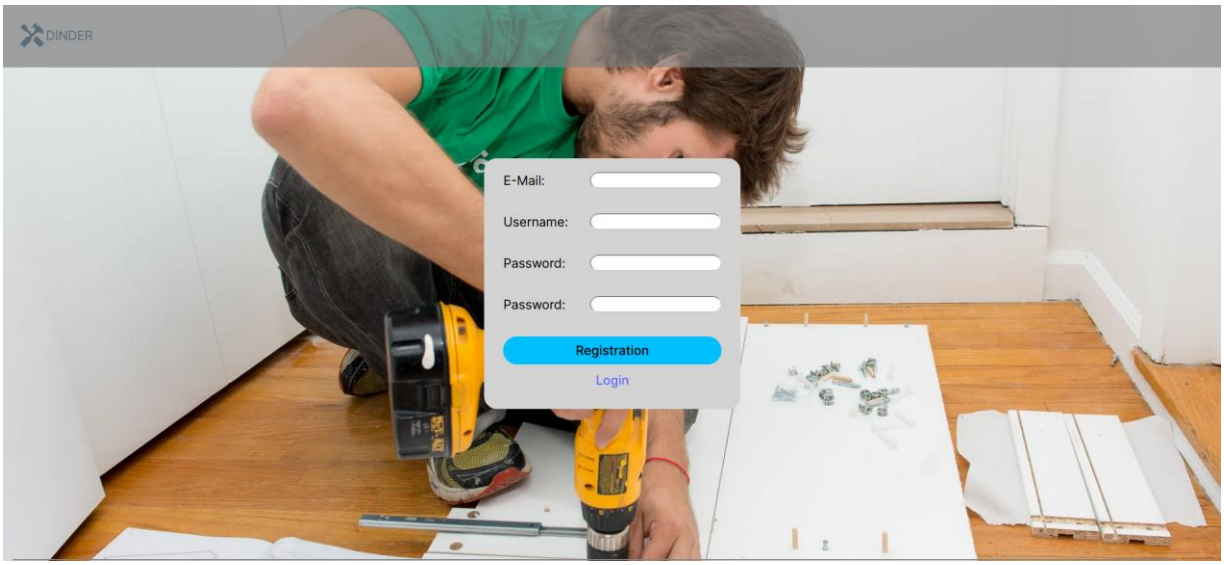


Abbildung 4: Demo - Registration

Anschließend gelangt man zu einer Swipe-Page, welche alle verfügbaren Anzeigen darstellt. Hierbei kann der Benutzer die einzelnen Anzeigen akzeptieren oder ablehnen.

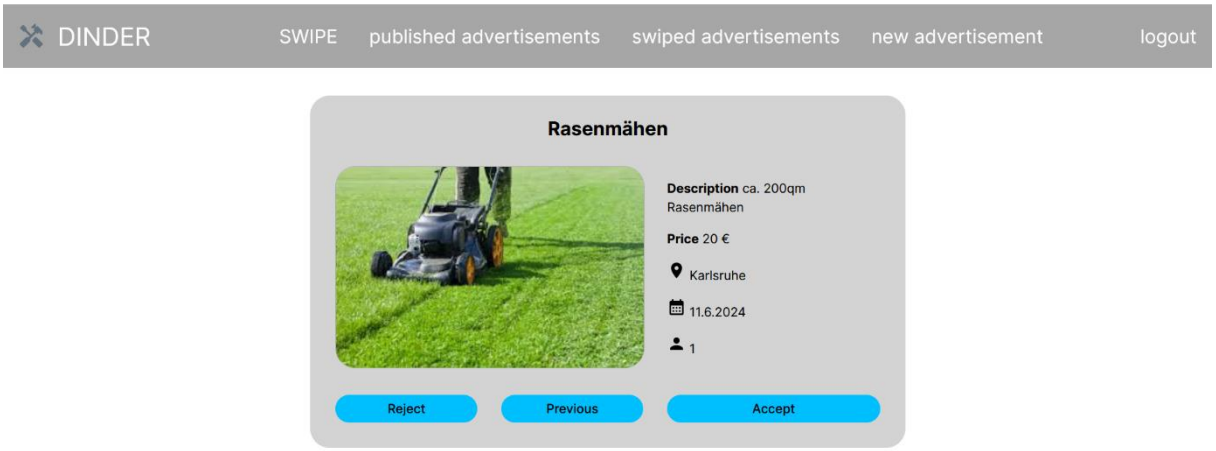


Abbildung 5: Demo - Swipe

Zur Anzeigen-Verwaltung gibt es eine Übersichtsseite, welche die Anzeigen darstellt, welche man auf der Hauptseite akzeptiert hat. Hierbei wird der aktuelle Status der Anzeige visualisiert (Pending, Denied, Accepted), sowie gibt es die Möglichkeit den „Interesse“-Zustand zurückzunehmen.

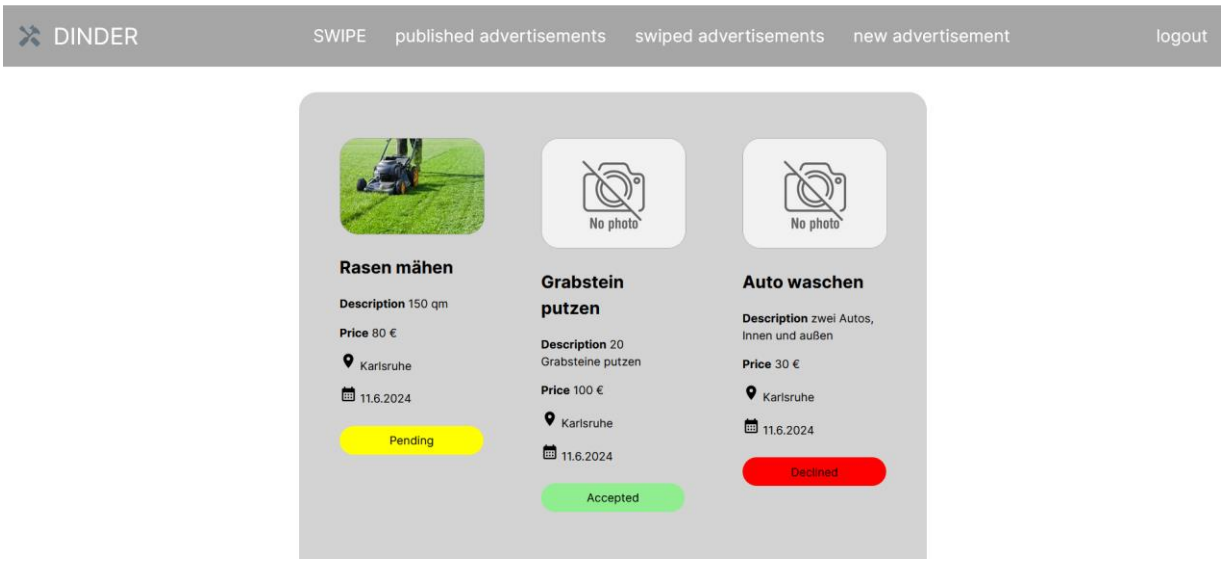


Abbildung 6: Demo – Published advertisements

Im Fenster „new advertisement“ gibt es die Möglichkeit neue Anzeigen zu erstellen. Die Eingabefelder können mit den gewünschten Angaben befüllt werden und es gibt die Option ein Bild zu der Anzeige hochzuladen.

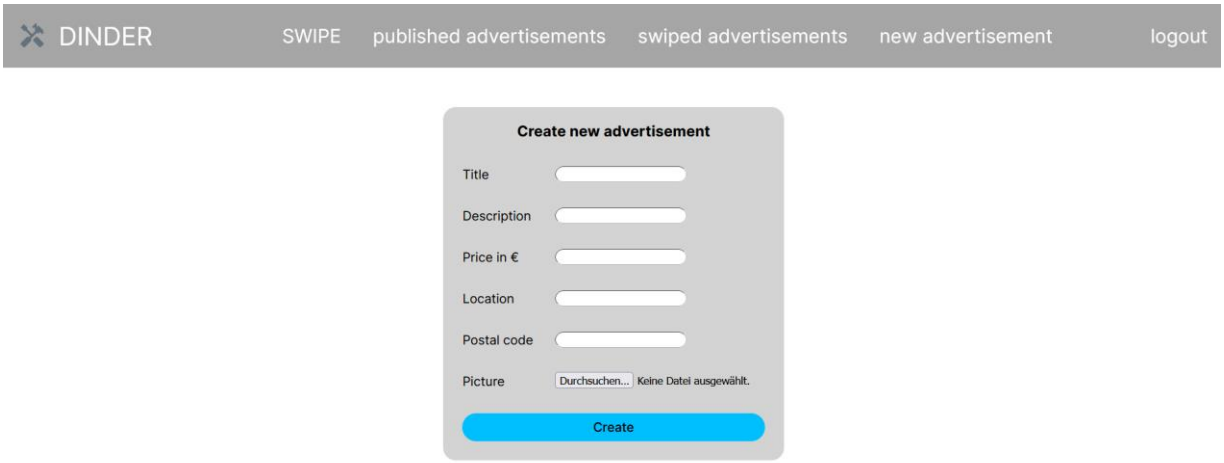


Abbildung 7: Demo – Create advertisement

Um die eigens erstellten Anzeigen zu verwalten stellt die Seite „published advertisements“ diese in einer Übersicht dar, wobei hier zu jeder Anzeige die Interessenten aufgelistet werden können. In diesem Kontext können diese dann akzeptiert oder abgelehnt werden.

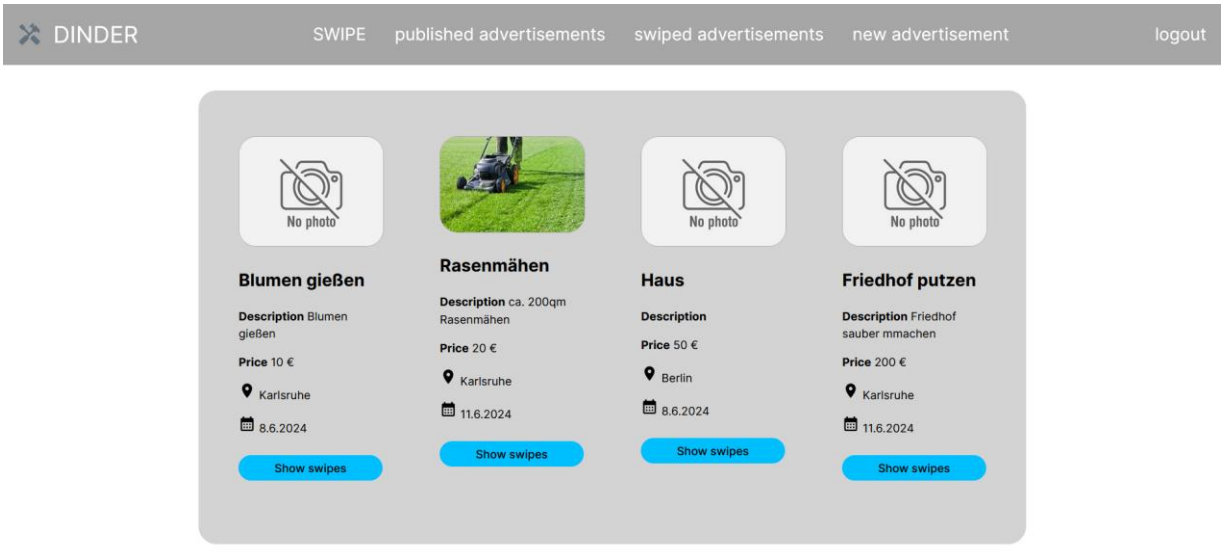


Abbildung 8: Published advertisements

Die anfangs definierte Chat-Funktion wurde in dieser Demo noch nicht umgesetzt, da wir uns nach der Gruppenreduzierung im vierten Semester auf die Hauptfunktionen des Programms fokussiert haben.

Tech-Stack

Frontend

- React
- NextJS
- TypeScript
- Redux
- Styled-components

Backend

- Spring Boot
- Redoc mit Spring Redoc
- JSON Web Tokens
- Hibernate
- Gradle

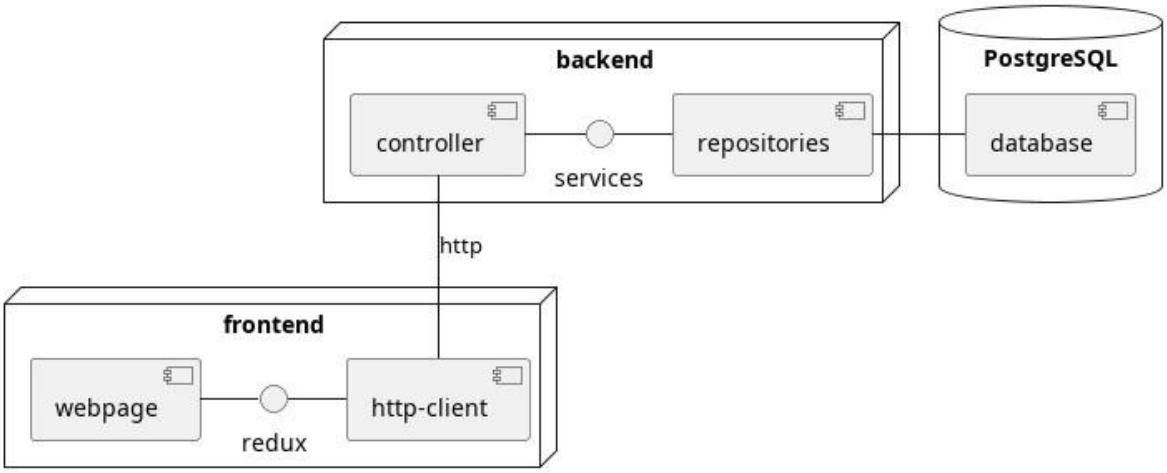
Datenbank

- PostgreSQL
- Supabase
- Flyway

Testing/Codequalität

- CI/CD-Pipeline
- GitHub Actions
- SonarCloud
- Biome Linter
- Biome Formatter
- REST-assured
- Mockito

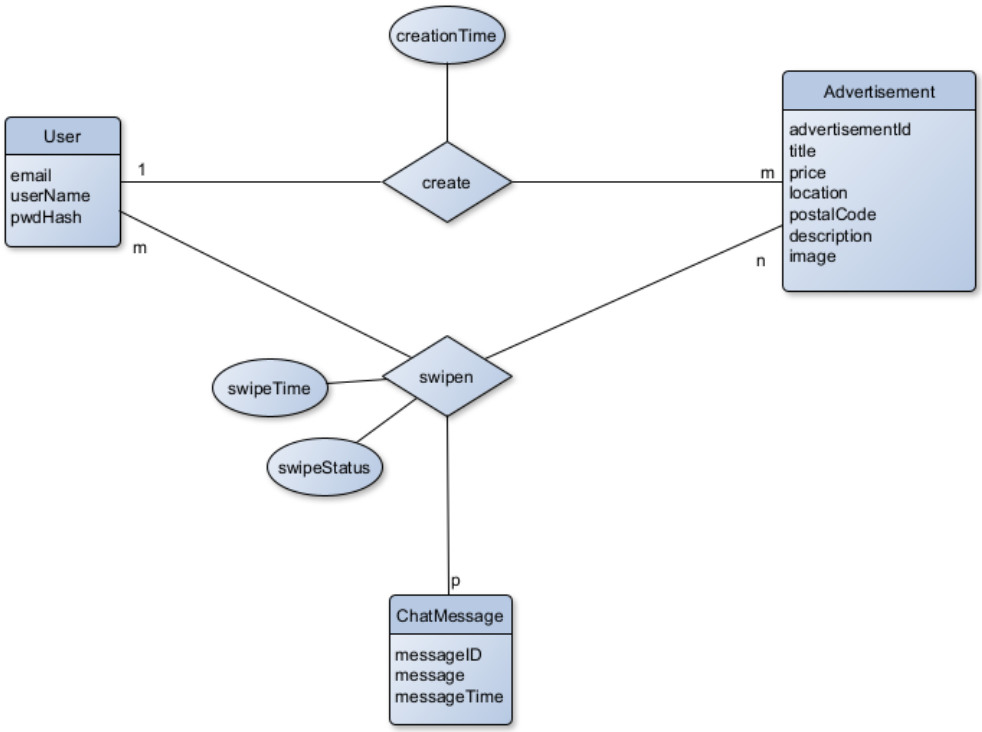
Architekturstile/-entscheidungen



Generell haben wir das MVC-Pattern benutzt.

Dabei stellt die Webpage im Frontend die View dar. Der Controller im Backend entspricht dem Controller im MVC-Pattern. Die Entity-Klassen im Backend, sowie die Repository-Klassen repräsentieren das Model im MVC-Pattern.

Datenbankdesign



Beim Datenbankdesgin haben wir den Benutzer (User), sowie die Anzeige (Advertisement) als eine Relation definiert. Hierbei kann ein Nutzer sowohl beliebig viele Anzeigen erstellen als auch für beliebig viele Anzeigen Interesse anmelden (swipen). Die dritte Relation Chat ist im Datenbankdesign vorhanden und sollte den Funktionsumfang der Anwendung ausweiten. Aus zeitlichen Gründen ist die Umsetzung nur auf der Datenbankebene erfüllt und ist anderweitig noch nicht in die Demonstration eingebaut.

Softwarequalität

CI/CD

Um eine erhöhte Codequalität zu gewährleisten, wurden zwei Pipelines erstellt, welche Biome ausführen. Biome stellt hierbei ein Tool zur statischen Codeanalyse dar. Einerseits werden in einer Pipeline Fehler und Verbesserungen des Codes als Bericht hinterlegt. Die zweite Pipeline formatiert den Frontend Code automatisch bei einem Push. Dadurch wird zwar durch das Tool aktiv in den Code eingegriffen, aber durch die geringe Größe des Projekts und gegenseitige Absprachen sind wir damit gut zurechtgekommen.

Eine weitere Pipeline bildet das Projekt und führt SonarCloud aus. Dadurch werden die Tests im Backend durchlaufen und die Ergebnisse in GitHub Actions, sowie in SonarCloud angezeigt. Nach erfolgreichen Testdurchläufen wird zusätzlich ein Artefakt gebildet, welches in der Pipeline hinterlegt wird. Durch die direkte Benachrichtigung des Entwicklers fallen auch fehlgeschlagene Tests sofort auf und können behoben werden.

Hier können sich die verschiedenen Pipelines angeschaut werden.

Tests

Zum Testen des Backends verwenden wir API-Tests. Diese überprüfen die Korrektheit der Endpoints bei richtigen und falschen Eingaben. Hierfür wird REST-assured und Mockito verwendet.

Das Frontend wird durch einen in der Pipeline ausgeführten Biome Linter überprüft, der eine statische Codeanalyse durchführt. Zusätzlich gibt es einen Biome Formatter in einer weiteren Pipeline der automatisiert den gepushten Quellcode des Frontends formatiert.

Metriken

Durch die Nutzung von SonarCloud haben wir uns für die folgenden drei Metriken im Backend entschieden, welche automatisch berechnet werden:

- Reliability
- Security
- Maintainability

Die Metriken werden auf der Skala von A (am besten) bis F (am schlechtesten) bewertet. Unser Code erfüllt die von gestellten Anforderungen (in jeder Kategorie A), was unter SonarCloud dokumentiert ist.

Somit stellen die Metriken sicher das unser Code so wartbar, sicher und zuverlässig wie möglichst ist. Dennoch bietet dies keine Garantie dafür.

Für das Frontend wurden aufgrund der schweren Testbarkeit keine dedizierten Tests geschrieben. Um dennoch die Qualität des Frontends zu bewerten, wird der Linter verwendet. Hierbei ist das Ziel keine Linter-Fehler zu haben.