

Fachbereich VI – Informatik und Medien

Studiengang Medieninformatik Bachelor

Konzeption und Umsetzung einer Web- Applikation für Studentenjobvermittlung

Bachelorarbeit

Zur Erlangung des akademischen Grades

- *Bachelor of Science* –

vorgelegt von

Mohamad Alschammari

Matrikelnummer: 838703

am 20. Sep 2022

Betreuerin: Prof. Dr. Siu

Gutachterin: Prof. Dr. Strippgen

Kurzfassung

Es wäre schön, wenn es eine Plattform im Internet für ein Studentenjobportal gibt, damit die Studenten sich schnell und flexible Jobs finden.

Es gibt im Internet zwar viele Jobsportale, aber sie sind nicht spezifisch für Zeitarbeit für Studenten, und es kann Zeitaufwendig bei der Suche und der Findung sein, dadurch, dass sie viele Jobsportale und lang suchen sollen, um passende Angebote für Studenten zu finden.

Deswegen im Rahmen dieser Bachelorarbeit wird von mir eine Applikation mit React.js, Node.js, Express, MySQL für Studentenjobvermittlung entwickelt werden, auf der die Studenten flexible Jobs finden können.

Diese Applikation wird außerdem den Firmen, die nach Studenten für vorläufige Zeit, Anzeigen hinzufügen ermöglichen.

Der Admin wäre der Vermittler zwischen den Studenten und den Unternehmen, indem er die Bewerbungen der Studenten und Anzeigen verwalten würde.

Im Frontend wird React.js umgesetzt werden, um interaktive Benutzeroberflächen zu erstellen, und für das Style Sheets wird SASS verwendet werden.

Im Backend wird Node.js umgesetzt werden, um einen Server, der mit der Datenbank und dem Frontend kommuniziert, zu bauen.

MySQL wird zum Verwalten der Datenbank benutzt werden.

Es werden Prototyps und grafische Benutzeroberfläche für die App mit WireframeSketcher und Adobe XD -Tool erstellt.

Inhaltsverzeichnis

Kapital 1.....	1
Einleitung.....	1
1.1 Motivation	1
1.2 Aufgabenstellung und Zielsetzung.....	2
1.3 Gliederung der Arbeit.....	3
Kapital 2.....	4
Technische Grundlagen	4
2.1 WireframeSketcher.....	4
2.2 Adobe XD	5
2.3 Visual Studio Code	6
2.4 JavaScript	7
2.5 React.JS.....	8
2.6 SASS.....	9
2.7 Node.JS	10
2.8 Express.JS	11
2.9 MySQL.....	12
Kapital 3.....	14
Analyse	14
3.1 Zielgruppe	14
3.2 Use-Case-Diagramm	14
3.3 Zielbestimmung	15
Kapital 4.....	17
Entwurf	17
4.1 Oberflächenentwurf.....	17
4.1.1 Landing-Page.....	17

4.1.2 Registrierung-Page.....	18
4.1.3 Anmeldung-Page.....	20
4.1.4 Home-Page-Admin	21
4.1.5 Alle-Angebote-Page-Admin	22
4.1.6 Jobs-Page-Admin	23
4.1.7 Home-Page-Student.....	24
4.1.8 Offer-Details-Page-Student	25
4.1.9 Meine-Jobs-Page-Student	26
4.1.10 Home-Page-Firma.....	27
4.1.11 Angebot-hinzufügen-Page-Firma.....	28
4.2 ER-Diagramm	29
4.4 Klassendiagramm	30
4.3 Aktivitätsdiagramme	31
4.3.1 Student.....	31
4.3.2 Firma	32
4.3.3 Admin.....	33
Kapital 5.....	34
Implementierung	34
4.1 Frontend	34
4.2 Backend	39
4.3 Frontend-Seiten.....	49
Kapital 6.....	55
Test.....	55
6.1 Anwendungsfälle-Manuell-Testen.....	55
6.1.1 Angebot Hinzufügen.....	55
6.1.2 Bewerben	56
6.1.3 Bewerbung-Bestätigen-Ablehnen.....	57

6.2 Übersicht der erfüllten Kriterien.....	58
Kapital 7.....	61
Ausblick	61
7.1 Zusammenfassung.....	61
7.2 Optimierung.....	62
Abbildungsverzeichnis	65
Tabellenverzeichnis	67

Kapital 1

Einleitung

In diesem Kapital werden Zweck, Aufgabenstellungen, und Struktur der Bachelorarbeit dargelegt.

1.1 Motivation

Heutzutage suchen die meisten Menschen nach Stellen im Internet, und dies ist weltweit erweitert.

Auch die meisten Unternehmen momentan bevorzugen es, Angebote in Jobportale statt der Post zu posten bzw. Bewerbungen zu bekommen.

Bewerbungen ausdrucken und per Post schicken gehört diese Methode schon lange der Vergangenheit an.

Die Mehrheit der in Deutschland studierenden brauchen flexible Jobs neben dem Studium zu finden, die ihrer Zeit im Lauf des Semesters passen.

Es ist den Studenten zeitaufwändig, sich im Internet bei vielen und verschiedenen Jobportale, die eigentlich nicht für Studenten spezialisiert sind, passende Jobs zu finden.

Es wäre eine gute Lösung für dieses Problem, wenn es eine Applikation für Studienjobvermittlung gibt, auf der die Studenten sich um flexible Jobs bewerben bzw. die Firmen Angebote anbieten können.

Diese Applikation soll einen klaren digitalen Buchungsprozess und geringe Verwaltungsaufwand sowohl für die Studenten als auch für die Unternehmen.

Deswegen ist die Idee in dieser Bachelorarbeit entstanden, die Bencom-App aufzubauen.

1.2 Aufgabenstellung und Zielsetzung

Im Rahmen dieser Bachelorarbeit wird die Applikation (Backend – Frontend) entwickelt.

Die Applikation soll einen digitalen Buchungsprozess für die Studenten bzw. die Firmen bieten.

Die Daten sollen sicher in einer sicheren Datenbank abgespeichert werden.

Jeder Benutzer soll laut seiner Rolle den Zugriff auf seine eigenen Daten haben.

Die Firmen sollen Stellen für Studenten mit allen Informationen wie zum Beispiel das Gehalt, die Tätigkeit, und Zeit anbieten können.

Die Studenten sollen in der Lage sein, sich die allen verfügbare Angebote anschauen und bewerben.

Der Admin soll die Fähigkeit haben, die Entscheidung zu treffen, entweder die Bewerbung zu bestätigen oder zu ablehnen.

Die allen Nutzer sollen den Zustand der Bewerbung zu verfolgen können.

Die Firmen sollen nach dem erfolgreichen Einsatz den Zustand der Bewerbung zum "beendet" ändern können, um zu bestätigen, dass der Student zum Job gekommen ist bzw. gearbeitet hat.

Die Applikation soll gut und einfach organisiert und strukturiert werden, damit die Benutzung der App einfach und schnell für die allen Benutzer sein kann.

Die Vorteile dieser Applikation sind:

- Für Studenten: Sie können sich schnell und dynamisch um verschiedene passende Stellungen bei verschiedenen Firmen auf einer Plattform bewerben.
- Für Unternehmen: Sie haben die Möglichkeit, Schnell und dynamisch mehrere vorqualifizierte Studenten in Zeitarbeit zu finden.

1.3 Gliederung der Arbeit

Das erste Kapitel bietet eine Einleitung in das Thema. Die Motivation, Aufgabenstellung und Zielsetzung werden erläutert.

Im zweiten Kapitel geht es um technische Grundlagen, welche für das Nachvollziehen der Arbeit hilfreich sind.

Es wird in Kapitel Drei das Verhalten der Applikation aus Anwendersicht mit Use Case Diagramm dargestellt.

Das Design der Benutzeroberflächen bzw. die Struktur des Systems werden im vierten Kapitel beschreibt, indem der Prototyp angezeigt, und Klassendiagramm erstellt werden.

Im Kapitel Fünf wird das zuvor geplante Entwurf implementiert, und Codeausschnitte des Quellcodes von Frontend bzw. Backend in Abbildungen dargestellt.

Dann wird die Applikation getestet und ausgewertet im sechsten Kapitel.

Zum Schluss im letzten Kapitel würde ich gerne meinen Ausblick auf die Arbeit äußern, und erklären, wie sie auch verbessert werden könnte.

Kapital 2

Technische Grundlagen

Dieses Kapitel erläutert die technischen Grundlagen, welche bei der Implantierung Anwendung finden.

2.1 WireframeSketcher

Es ist ein Tool, das zum Entwurf Prototyp "Wireframes" zur Modellierung von Web oder Mobile-Applikationen verwendet wird. [B1]

Es hilft dabei, die Ideen und wie die Applikation aussehen soll zu visualisieren.

Der Bildschirme, die mit dieser Software erstellen kann, sehen wie Handgezeichnet aus.

Mit diesem Tool können einfach flexible Benutzeroberfläche auf verschiedene Arten angeordnet werden.

In der „Abbildung 1“ wird die allen wichtigen Ansichten gezeigt, die von WireframeSketcher benutzt werden können.

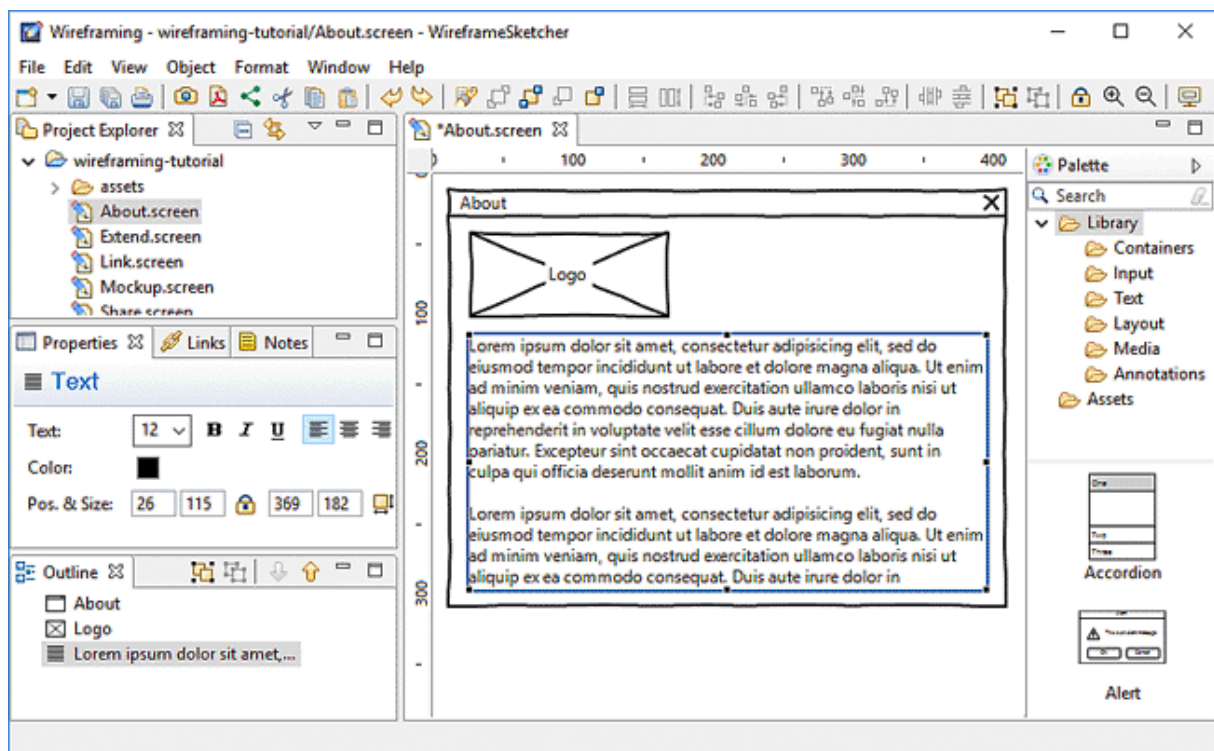


Abbildung 1: Die Ansichten in Wireframesketcher [B1]

2.2 Adobe XD

Es ist eine Software, die zum Entwurf grafischen Benutzeroberflächen für Web oder Mobile-Applikationen verwendet wird. [B2]

Adobe XD wird vor allem im Bereich Interaktionsdesign benutzt, um ein ideales Design für die Mobile und Webapplikationen erstellen zu können, welches als Vorlage während der Entwicklung verwendet werden können.

Dieses Tool bietet auch verschiedene Funktionen, die das Erstellen komplexer Designs leichter machen,

Außerdem bietet Adobe XD eine große Auswahl an integrierten Formen und Symbolen.

Es hat auch ein Animationstool, das es einfach, Prototypen Bewegung hinzuzufügen. Und mit Adobe XD kann man die Funktionalität der interaktiven Prototypen auf realen Geräten testen

In der „Abbildung 2“ sieht man ein Beispiel vom Entwurf einer Mobile-App und wie die Plattform von Adobe XD und ihre Bearbeitungstools aussehen.

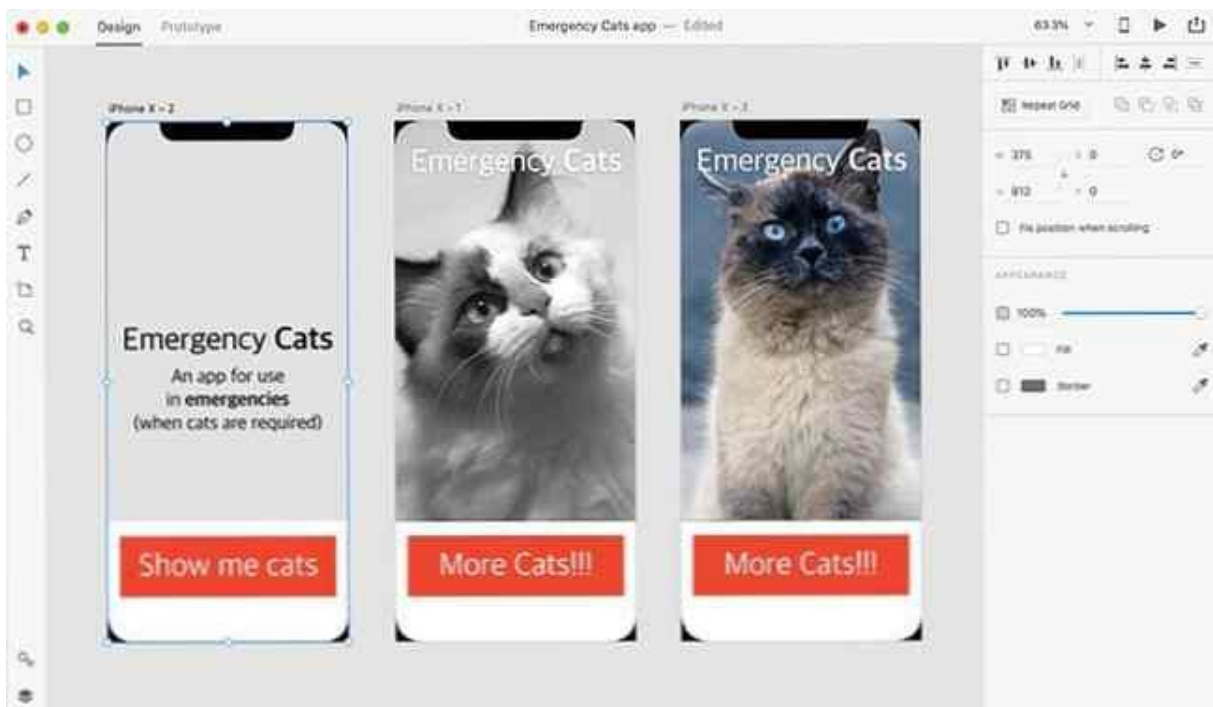


Abbildung 2: Screenshot von der Plattform von Adobe XD [B3]

2.3 Visual Studio Code

Es ist ein Quelltext-Editor, der von Microsoft entwickelt worden ist, und wird zum Code schreiben bei der Appentwicklung benutzt. [B4]

VS-Code unterstützt viele Programmiersprache wie JavaScript, PHP usw.

Visual Studio Code hat auch Themen, bei den der Syntax farblich ist, und Extensionen, damit die Entwickler schneller und einfacher den Code schreiben können.

Mit VS-Code unterstützt das Testen, das Verwalten, Validierung eines Codes bei der Entwicklung.

Außerdem enthält VS-Code einen eigenen Terminal, mit dem man die Befehle eingeben und die Antworten sehen kann.

Es hat sehr guter Debugger für Node.JS Applikationen, und Textsuche-Feature in Dateien und Ordnungen.

Die allen Änderungen in der Datei werden automaisch abgespeichert.

Git ist von VS-Code unterstützt, damit man auch den Quellcode verwalten kann, ohne den Text-Editor zu verlassen.

Es hat auch eine gute Eigenschaft und zwar Auto-Vervollständigung, mit der der Syntax automatisch ergänzt wird, und dies macht es schneller, beim Code schreiben.

Beispiele der Erweiterungen vom VS-Code, die in dieser Bachelorarbeit benutzt wurden und bei der Entwicklung hilfreich sind:

1- Auto Close Tag: diese Erweiterung ist sehr praktisch für HTML-Code, indem der HTML automaisch beim Tippen abgeschlossen wird.

2- Live SASS-Compiler: Mit dieser Erweiterung kann man CSS-Datei automatisch aus einer SASS-Datei kompilieren lassen.

Um VS-Code benutzten zu können, muss es zuerst auf dem Rechner installiert sein, und es ist verfügbar für Betriebssysteme Windows, macOS, und Linux.

2.4 JavaScript

JavaScript ist eine Objekt Orientierte Programmiersprache, die am häufigsten zur Entwicklung dynamischen attraktiven Webseiten verwendet wird. [B5]

JavaScript kann auch sowohl für das Frontend (Benutzeroberfläche) als auch für das Backend (Server) verwendet wird, also die Entwickler brauchen keine andere Programmiersprache für das Backend zu benutzen.

Darüber hinaus kann JavaScript integriert mit HTML und CSS um dynamische Aktionen und Styling auszuführen.

Mit JavaScript können die Entwickler dynamische Elemente und Animationen erstellen.

Einer der Vorteile der JavaScript ist, dass sie keine zusätzliche Belastung auf dem Webserver, da es clientseitig ist.

JavaScript kann einfach implementiert werden, und die Browser-Probleme automatisch beheben.

Um JavaScript auszuführen, braucht man keine besondere erforderliche Einrichtung zu machen.

Man braucht Vorkenntnisse in HTML und CSS zu haben.

JavaScript unterstützt asynchronen Prozess mit Promises- und Async-Funktionen, also in Promises kann man die Anfrage stellen und ihr eine „.then()-Klausel“ am Ende anhängen, die nur nach dem Abschluss des Promises ausgeführt wird. Als Alternativ von Promises kann man async-await-Funktionen verwenden.

Asynchrone Funktionen werden nicht sequentiell, sondern parallel ausgeführt, welches eine positive Auswirkung auf die Verarbeitungszeit und Reaktionsfähigkeiten der Seite hat.

Was als Nachteil von JavaScript betrachtet wird, ist, dass die JavaScript für den Arbeitsspeicherplatz des ausführenden Rechners zusätzliche Belastung verursacht.

Auch ist der JavaScript-Code immer vom Client einsehbar.

Es wird aus Sicherheitsgründen empfohlen, dass die Nutzer JavaScript zu deaktivieren, um sich gegen die Angriffe der Hackern zu schützen.

Es gibt für JavaScript mehrere Frontend-Frameworks wie zum Beispiel React.JS, Angular.JS, Vue.JS usw.

Für das Backend gibt's auch mehrere Frameworks wie zum Beispiel Node.JS, Meteor.JS und Express.JS

2.5 React.JS

React.JS ist ein JavaScript Bibliothek und wurde von Facebook entwickelt. [B6]

Es wird für Frontend-Entwicklung verwendet, um dynamische wiederverwendbaren Benutzeroberflächen- Komponenten zu erstellen.

Um React.JS verwenden zu können, braucht der Entwickler solide Kenntnisse in HTML5, CSS, und JavaScript haben.

React.JS kann auch auf dem Server mit Node rendern und für Mobile-Applikation mit React Native verwendet werden.

React.JS ist einfach zu lernen und zu verstehen, und sie hat keine Probleme mit Suchmaschinenoptimierung.

Außerdem können die Entwickler mit React.JS die Komponente mehrmals verwendet, ohne denselben Code wieder oder mehrmals zu schreiben.

(Siehe die Abbildung 3)

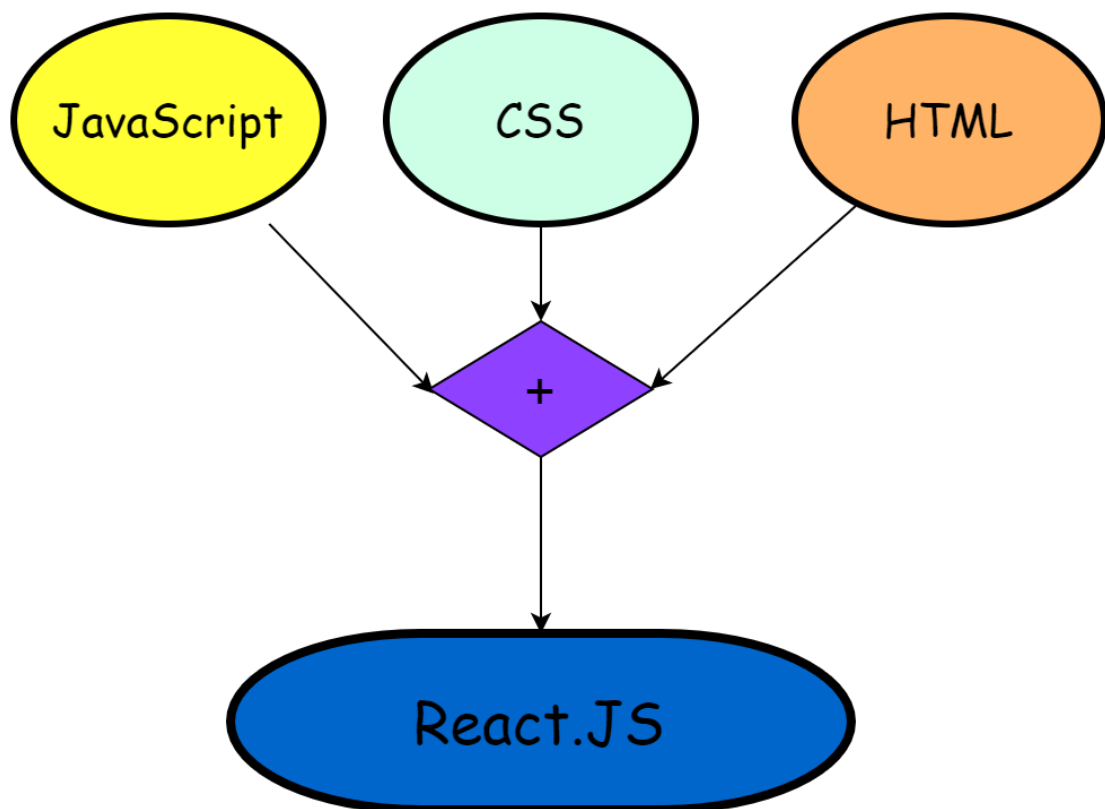


Abbildung 3: Wie die Komponente in React.JS gebaut wird.

2.6 SASS

SASS (Syntactically Awesome Style Sheets) ist eine Erweiterung von CSS und zum Styling im Frontend verwendet. [B7]

Es lässt die Entwickler Variablen, Funktionen, verschachtelte Regeln, Mixins, Inline-Importe und mehr verwenden, und dies hilft den Entwicklern, Styling schneller und dynamisch zu erstellen.

Man kann mit SASS das Projekteinfach organisieren und sinnvolle abschnitten unterteilen.

Da das Stylesheet kombiniert wird, wird die Anzahl von HTTP-Anfragen reduziert. (Siehe die Abbildung 4)

Im Vergleich zu CSS, kann man each, while oder for-Schleifen bzw. if-else-Bedingungen, welche es bei der Entwicklung einfacher machen.

Außerdem ist CSS Beim DRY (Do not repeat your Self) nicht gut, aber SASS bietet dagegen die Möglichkeit, Mixin, und Variablen zu verwenden.

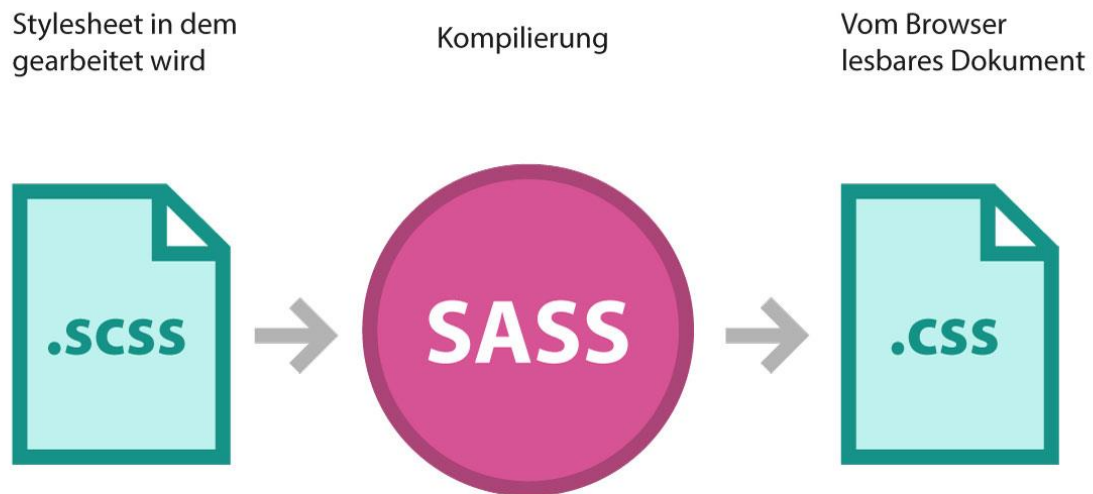


Abbildung 4: Wie SASS funktioniert. [B8]

2.7 Node.JS

Node.js ist eine serverseitige, eventbasierte, basierte auf der V8-Engine von Google Chrome Plattform, die für Netzwerkanwendungen konzipiert worden ist. [B9]

Es wurde von Ryan Dahl in 2009 entwickelt, und seitdem es weltweit verbreitet ist.

Node.JS ist schnell und wird zum schicken und empfangen die Daten von Client und der Datenbank eingesetzt.

Node.JS ist eine ideale Lösung für die Echtzeit-Web-Applikationen, die enthält zum Beispiel Chat Anwendung oder Spielen.

ein Single Thread Applikation.

Node.JS sieht die allen Anfragen als Event und trägt sie in Event Queue, und hat das Prinzip „First In First Out“, als die Anfrage zuerst ankommt, wird zuerst behandelt.

Das Event Loop ist eine endlose Schleife, die die Anfragen ins „Worker Thread“ bringen, und mittlerweile wird eine Callback-Funktion für jede Anfrage erstellt.

Wenn eine Behandlung einer Anfrage beendet ist, wird eine Callback-Funktion angerufen.

Multi Thread ist von Worker Threads Programm, das mit C++ Programmiersprache geschrieben wird, unterstützt, und das heißt, dass die Anfragen auf verschiedene Threads behandelt werden. (Siehe die Abbildung 5)

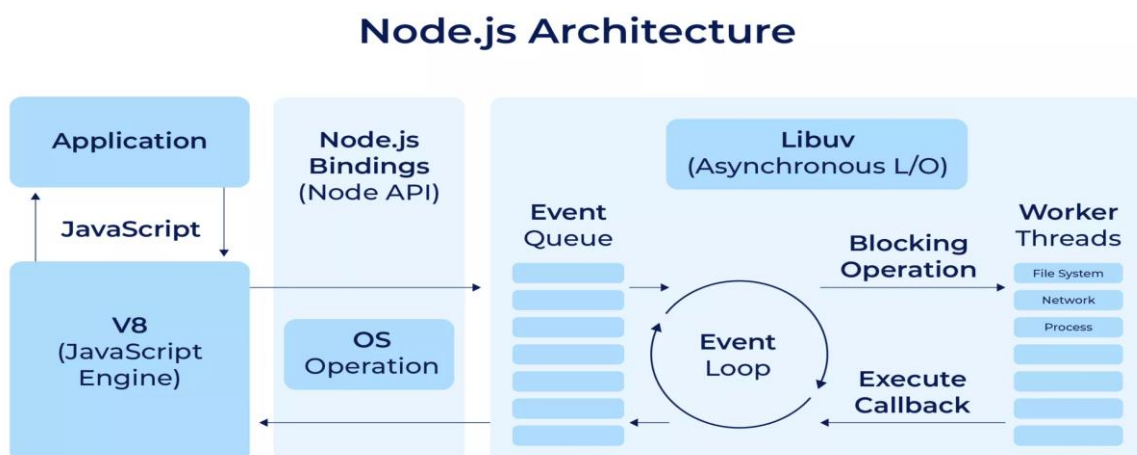


Abbildung 5: Die Architektur von Node.JS [B10]

2.8 Express.JS

Express.JS ist ein einfache und flexible serviceseitiges Node.JS-Framework von Web-Applikationen. [B11]

Die Entwickler können mit Express.JS Middleware einstellen, um HTTP-Anfragen wie POST, DELETE, UPDATE, GET zu erstellen.

Bei Express.JS wird die Routing-Tabelle für verschiedene HTTP-Anfragen-Funktionen einsetzen definiert.

Die Entwickler sind in der Lage mit Express.JS, die Anfragen mit Parameter zu übergeben.

Mit Express.JS wird die Entwicklung und Leistungsfähigen API schnell und einfach sein, um dynamische Webseite zu entwickeln.

Der Server empfängt die HTTP-Anfragen bekommen bzw. verarbeitet und dann schickt der Server die Antwort nach der Bearbeitung an den Client zurück, um die Informationen auf der Website angezeigt zu werden.

Express.JS kann mit Node.JS viele gleichzeitige Funktionen unterstützen.

Einer die Vorteile von Express.JS ist, dass sie auch die Caching-Funktionen unterstützt, damit der Code nicht immer wieder neue ausgeführt werden müsste, und dies macht die Ladung der Seiten schneller. (Siehe die Abbildung 6)

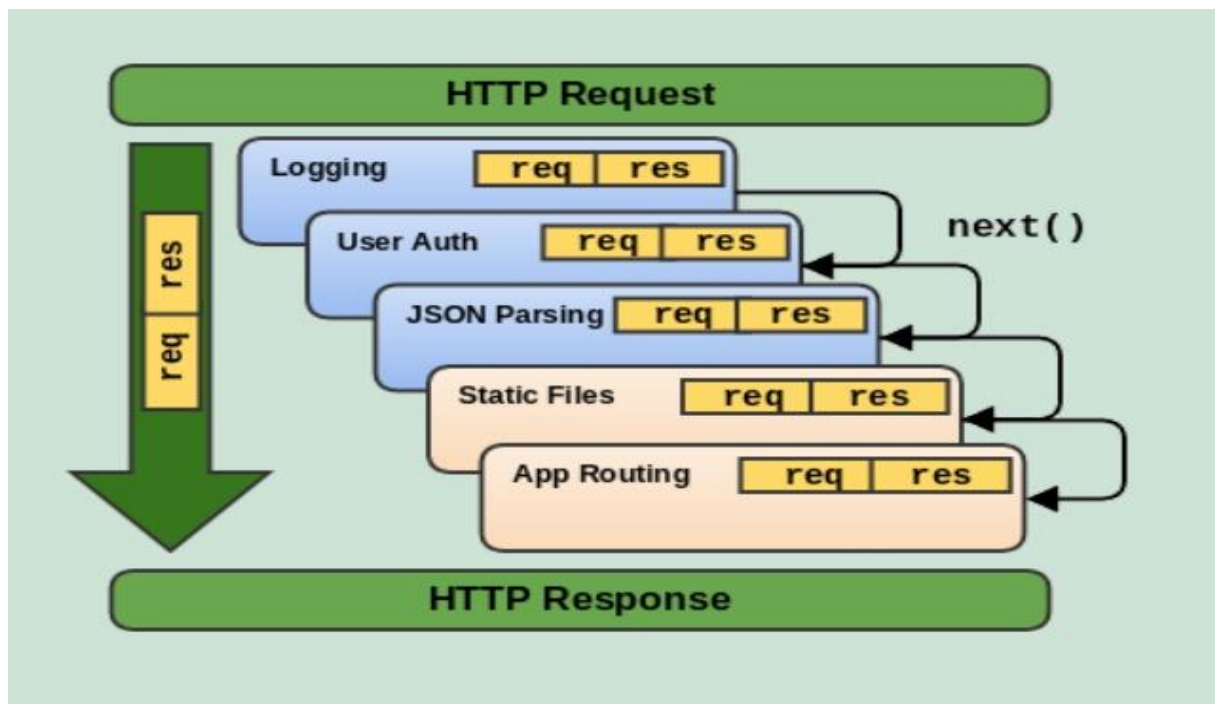


Abbildung 6: Middleware-Aufrufen in Express.JS [B12]

2.9 MySQL

Bei MySQL handelt sich um ein relationales Datenbanksystem, das von Unternehmen MySQL AB in 1994 erstellt bzw. von Oracle unterstützt ist.

Oracle ist ein US-amerikanischer Soft- und Hardwarehersteller mit Hauptsitz in Austin, Texas. Das Unternehmen ist spezialisiert auf die Entwicklung und Vermarktung von Computer-Hardware und -Software für Unternehmenskunden – insbesondere des Datenbanksystems Oracle Database [B13]

Mit MySQL wird eine Datenbank erstellt, auf der die Daten abgespeichert bzw. verarbeitet werden.

MySQL-Server ist in der Lage mehrere Anfragen gleichzeitig von Backend Servers empfangen, die MySQL Statement enthalten und von Client Anfragen abhängig sind, um die bereits abgespeicherten Daten abzurufen oder zu bearbeiten bzw. neu Daten in der Datenbank zu erstellen.

Man kann MySQL auf mehreren Plattformen einsetzen, und mit ihm Daten schnell und flexibel speichern und verwalten.

Die Daten werden in der Datenbank in zweidimensionalen Tabellen mit mehreren Reihen und Spalten bzw. mit verschiedenen Datentypen wie zum Beispiel: Datumformate, Numerische Daten, Text usw.

Jeder Datensatz kann maximal nur einen Primärschlüssel, dessen Wert nur einmal in jeder Tabelle existieren darf, und er besteht aus minimal einem Attribut.

Der Fremdschlüssel verweist auf den Primärschlüssel in einer anderen Tabelle. (Siehe die Abbildung 7)

Einer der Nachteile von MySQL ist es, dass sie für Großen Daten nicht entsprechend und empfohlen ist.

Außerdem hat sie Probleme mit der Stabilität und verursacht Beschädigung in bestimmten Anwendungsfällen.

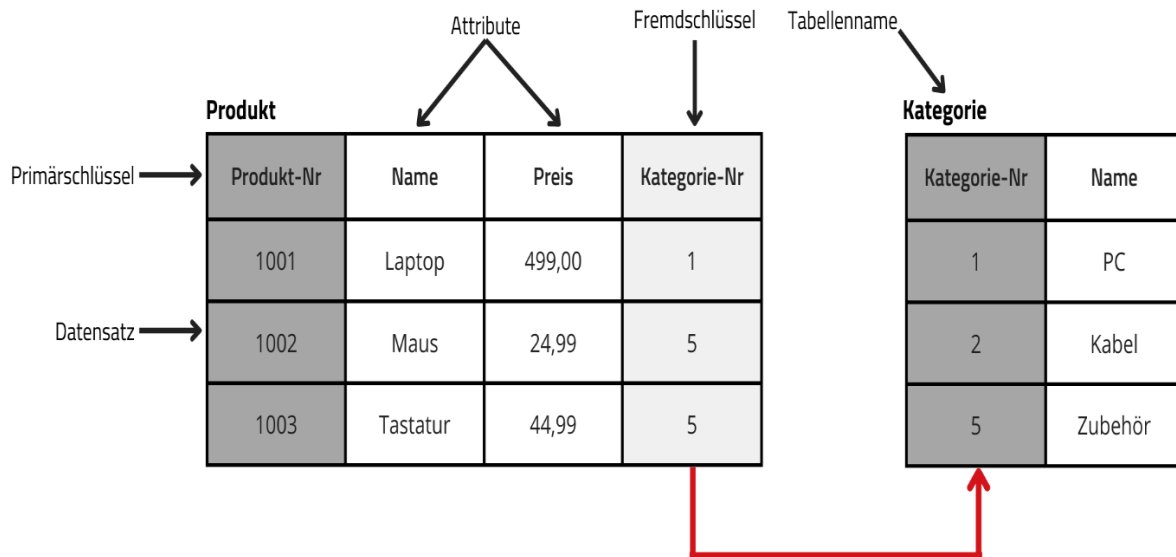


Abbildung 7: Ein Beispiel von relationaler Beziehung. [B14]

Kapital 3

Analyse

Laut der Zielsetzung werden in diesem folgenden Kapitel die Anforderungen an das zu entwickelnde Applikation festgelegt.

3.1 Zielgruppe

Die Anwendung kann von Studenten, die nach flexible Jobs suchen, eingesetzt werden. Weiterhin können auch die Unternehmen, die nach vorqualifiziertem studentischem Personal in Zeitarbeit suchen verwenden.

3.2 Use-Case-Diagramm

Gemäß den definierten Kriterien der Anforderungsdefinition (siehe Kapitel 3.1) ergeben sich die Anwendungsfälle, welche im folgenden Anwendungsfall-Diagramm dargestellt sind. (Siehe die Abbildung 8)

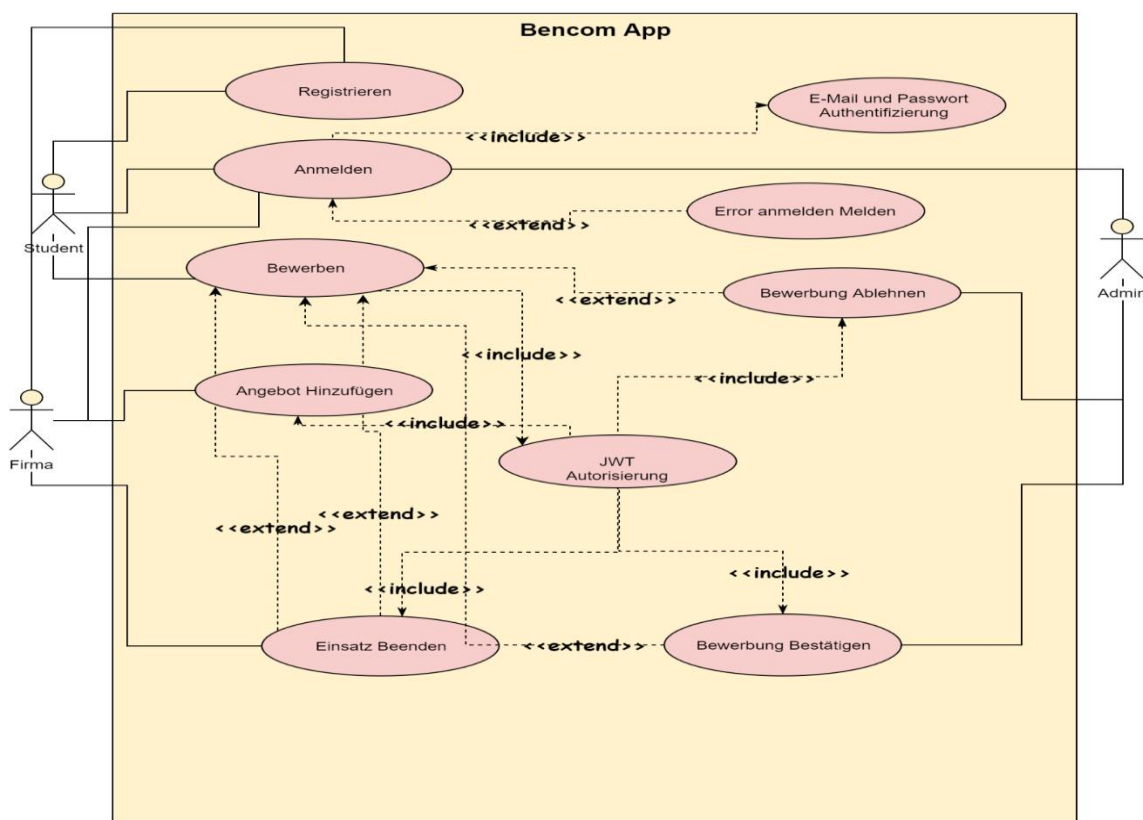


Abbildung 8: Use Case Diagramm von "Bencom" Applikation

3.3 Zielbestimmung

Die Applikation soll in der Lage sein, einen Buchungsprozess für die Benutzer (Studenten und Unternehmen) zu bieten. Zu diesem Zweck wird in der vorliegenden Abschlussarbeit ein Prototyp entwickelt, der die geforderten Kriterien erfüllt.

Die Zielbestimmung bezüglich der geplanten Applikationsfunktionen stellt sich wie folgt dar:

Muss-Kriterien:

1. Die Applikation soll die Möglichkeit den Nutzern bieten, sich zu registrieren.
2. Bei der Registrierung wird die Daten vom Client mit Axios ⁽¹⁾ (HTTP-POST-Request, der die Eingegabene Daten trägt) an den Server geschickt.
3. Der Server bei der Registrierung empfängt die Anfrage vom Client bzw. wird die Daten an die Datenbank senden, und das eingegebene Passwort wird auch verschlüsselt im Backend, bevor es an die Datenbank geschickt werden.
4. Es soll drei Rollen der Benutzer für die Applikation geben, und zwar "Admin, Student, und Firma".
5. Bei der Anmeldung der Benutzer nach der Validierung die Eingegabene Daten werden geprüft, ob die eingegebene E-Mail und Passwort vorhanden und richtig sind.
6. Bei der Anmeldung wird eine HTTP-GET-Anfrage vom Client an den Backend-Server gesendet, und dann die Antwort wird an den Client zurückgesendet.
7. Wenn die Daten richtig sind, wird der Benutzer in der Lage sein, die Applikation zu benutzen.
8. Nach der erfolgreichen Anmeldung wird der Server ein Key mit JWT ⁽²⁾ zur Autorisierung für den Benutzer erstellt.
9. Die Daten der Benutzer: Name, ID, Rolle, und das JWT-Key im lokalen Speicher abgespeichert, um die im Frontend bzw. bei den Funktionen zu verwenden.

⁽¹⁾ Axios ist ein *Promise-basierter* HTTP-Client für Node.JS und den Browser. [C1]

⁽²⁾ JWT ist eine Abkürzung von: JSON Web Token.

10. Der Client als Firma soll die Möglichkeit haben, Angeboten zu posten.
11. Die hinzugefügten von den Unternehmen Angebote sollen im Frontend angezeigt werden.
12. Der Client als Student soll sich um die Angebote bewerben können.
13. Nach der Bewerbung bekommt der Admin die Bewerbungen, und soll der Admin die Möglichkeit haben, zu bestätigen bzw. zu ablehnen.
14. Nach der Bestätigung wird der Status der Bewerbung zum "Anstehend" in der Datenbank geändert.
15. Der Zustand „beworben, anstehend, fertig“ der Bewerbung wird im Frontend gezeigt.
16. Der Client als Firma soll in der Lage sein, den Status der Bewerbung nach einem erfolgreichen Einsatz zum „Fertig“ ändern.
17. Bei jeder Funktion soll das Key des Nutzers geprüft, um den Zugang zu den Privilegien zu haben.
18. Die Nutzer sollen auch die Möglichkeit haben, auszuloggen.
19. Es wird Routing für die Seiten der Applikation verwendet, also das Zuordnen einer URL in der Anwendung zu einer bestimmten Funktion.
20. Beim Ausloggen sollen die Daten, die Im lokalen Speicher des Browsers sind, gelöscht werden.

Soll-Kriterien:

1. Die Firma soll in der Lage sein, die Angebote zu löschen.

Kapital 4

Entwurf

In diesem Kapitel wird der Entwurf der Applikation vom Frontend bzw. Backend dargestellt.

4.1 Oberflächenentwurf

In der Entwicklung dient der Applikation-Entwurf als technischer Plan für eine zu entwickelnde Applikation, um die Anforderungsanalyse technisch umzusetzen. Der Applikation-Entwurf kann maßgeblich dabei helfen, die Entwicklung zu rationalisieren.

Die Benutzeroberfläche einer Anwendung oder App ist das Gesicht des digitalen Produktes. Es wird ein Prototyp für das Frontend mit WireframeSketcher-App erstellt.

4.1.1 Landing-Page

Es wird ein Prototyp für das Landing-Page, welches den Benutzern Informationen über die App zeigt bzw. sowohl zur Registrierung als Student oder Firma als auch zur Anmeldung weiterleitet.

Bei allen Pages wird auch ein Header erstellt, der das Logo der Applikation und Kontakt Auswahl hat.

In der Fußzeile werden die Linke bzw. Sozialmedia Icons bezeichnet, die der Benutzer braucht, um weitere Informationen zu haben. (Siehe Abbildung 9).

Zwei Boxen werden erstellt, die Informationen und Vorteile für Studenten und Unternehmen haben, und was der Benutzer von der Benutzung der Applikation profitieren können, je nach seinen Bedürfnissen.

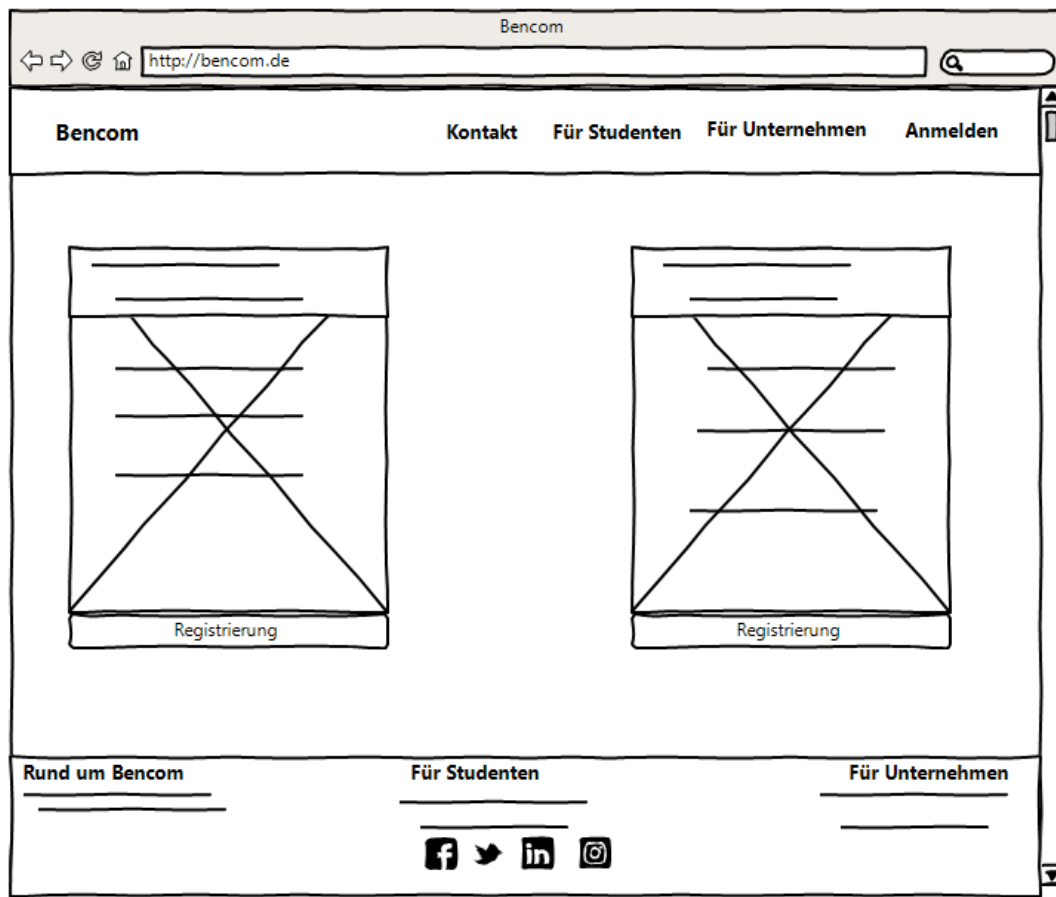


Abbildung 9: Landing-Page Prototyp

4.1.2 Registrierung-Page

Für die Registrierung werden zwei Seiten erstellt, und zwar eine für die Studenten und die andere für die Unternehmen, da jede von denen verschiedene passende Informationen eingeben soll.

Genauso wie das Landing-Page werden die Seiten ein Header und Fußzeile haben.

Hier wird nur ein Prototyp in der „Abbildung 10“ für die Registrierung der Unternehmen gezeigt.


Die Studenten sollen im Gegenteil zu den Firmen die Ausweisnummer, den Name der Hochschule, wo sie studierenden sind, und das Geburtsdatum eingeben.

Die Beiden Nutzer sollen die E-Mail und das Passwort eingeben, um nach der erfolgreichen Registrierung mit denen anmelden zu können.

Bencom

[http://bencom.de](#)

[Bencom](#)
[Kontakt](#)
[Für Studenten](#)
[Für Unternehmen](#)
[Anmelden](#)



Registren als Unternehmen

Name der Firma

Herkunft

Grundungsdatum

Reg.Nr

Adresse

Straße

Hausnummer

Postleitzahl

Stadt

Tel.Nummer

Fax.Nummer

Email

Zusätzlich

Vertreten durch:

Vorname

Nachname

Email

Tel.Nummer

Passwort

Passwort wiederholen

Registrieren

☐ Ich stimme zu

Rund um Bencom

Für Studenten

Für Unternehmen










Abbildung 10: Registrierung-Firma-Page Prototyp

4.1.3 Anmeldung-Page

Um die Applikation benutzen zu können, soll die Benutzer sich anmelden.

Die allen Benutzer so wie der Admin können sich durch dieselbe Seite anmelden.

Sie brauchen nur die E-Mail und das Passwort, mit denen sich bereits registriert haben. (Siehe die Abbildung 11)

The image shows a hand-drawn prototype of a web browser window for the login page of 'Bencom'. The browser's address bar displays 'http://bencom.de'. The page has a header with the 'Bencom' logo and navigation links: 'Kontakt', 'Für Studenten', 'Für Unternehmen', and 'Anmelden'. The main content area is a large rectangle containing a square icon with an 'X' inside, representing a missing image. Below this icon are three input fields: 'Email', 'Passwort', and a button labeled 'Anmelden'. A blue link 'Dein Passwort vergessen' is positioned below the password field. The footer contains three sections: 'Rund um Bencom' with two horizontal lines, 'Für Studenten' with one horizontal line, and 'Für Unternehmen' with one horizontal line. At the bottom center are four social media icons: Facebook, Twitter, LinkedIn, and Instagram.

Abbildung 11: Anmeldung-Page-Prototyp

4.1.4 Home-Page-Admin

Jeder Benutzer hat seine eigene Homepage, wohin er sofort nach der erfolgreichen Anmeldung weitergeleitet wird.

In der Homepage vom Admin wird eine Box gezeigt, die die Bewerbungen der Studenten mit ihren Informationen enthält.

Die angezeigte Box hat auch zwei Buttons, ein für Bestätigung und das andere für die Ablehnung der Anfrage.

Wenn der Benutzer angemeldet ist, zeigt der Header den Namen des Benutzers und anderen Linken gemäß der Rolle, ob die Rolle Student, Firma, oder Admin ist.

(Siehe die Abbildung 12)

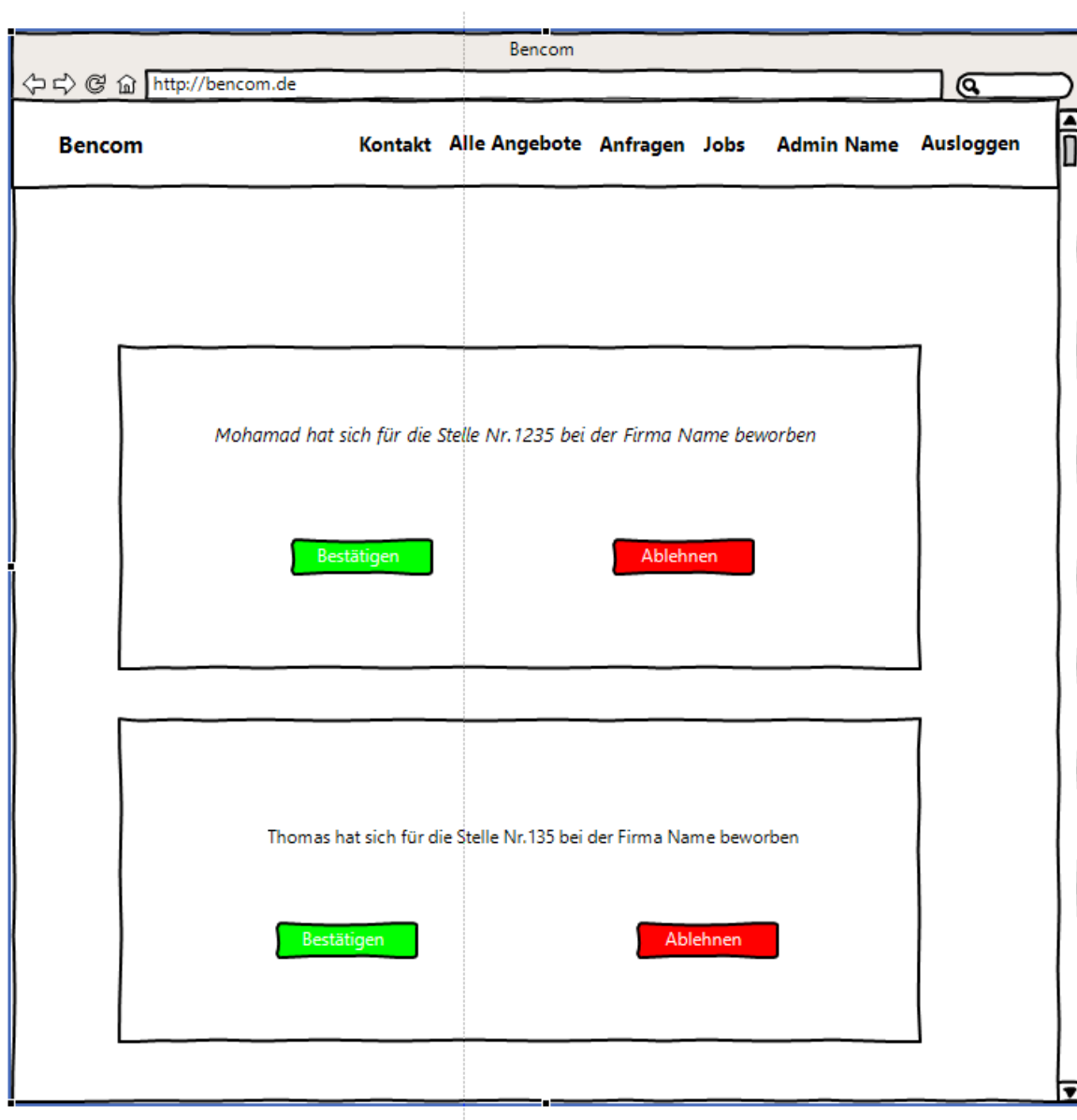


Abbildung 12: Home-Page-Admin-Prototyp des Benutzers

4.1.5 Alle-Angebote-Page-Admin

In dieser Seite kann der Admin die Allen Angebote sehen, die von den Firmen hinzugefügt wurden.

Jedes Angebot hat die Informationen über den Title der Stelle, Zeit, Gehalt, und die Adresse. (Siehe Abbildung 13)

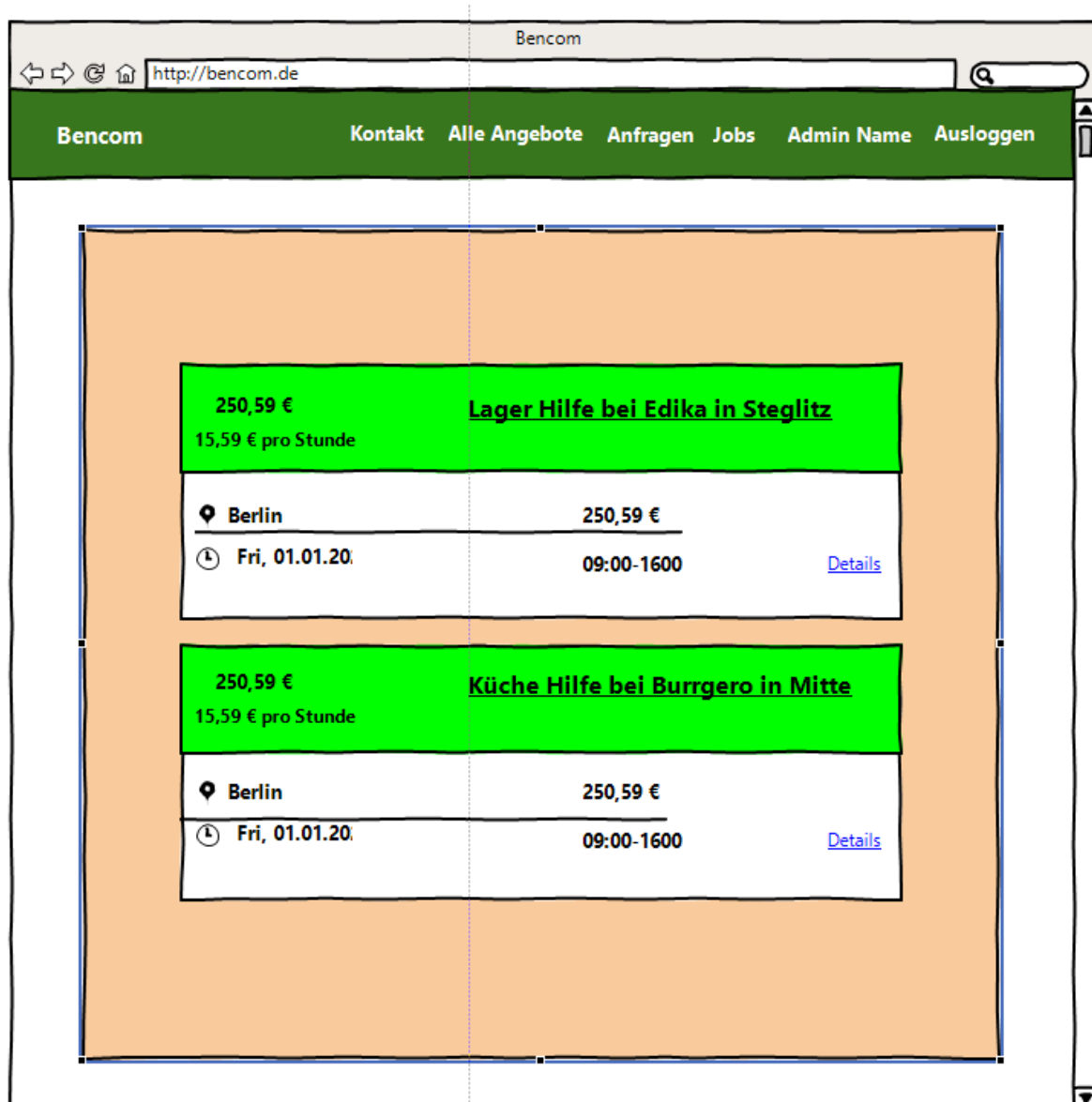


Abbildung 13: Alle Angebote-Page-Admin Prototyp

4.1.6 Jobs-Page-Admin

Auf dieser Seite soll der Admin in der Lage sein, den Zustand der allen Bewerbungen zu verfolgen.

Es gibt drei Zustände für die Bewerbungen, und zwar beworben, anstehend, und fertig.

Wenn der Student sich bewirbt, wird die Bewerbung in den „Beworben“ Zustand landen.

Danach wenn die Bewerbung bestätigt würde, würde die Bewerbung in den Anstehend Zustand landen würde.

Zum Schluss, wenn der Arbeitgeber nach dem erfolgreichen Einsatz kann den Zustand der Bewerbung zum Fertig ändern, damit der Bewerbung in die „fertig“ Spalte landet. (Siehe die Abbildung 14)

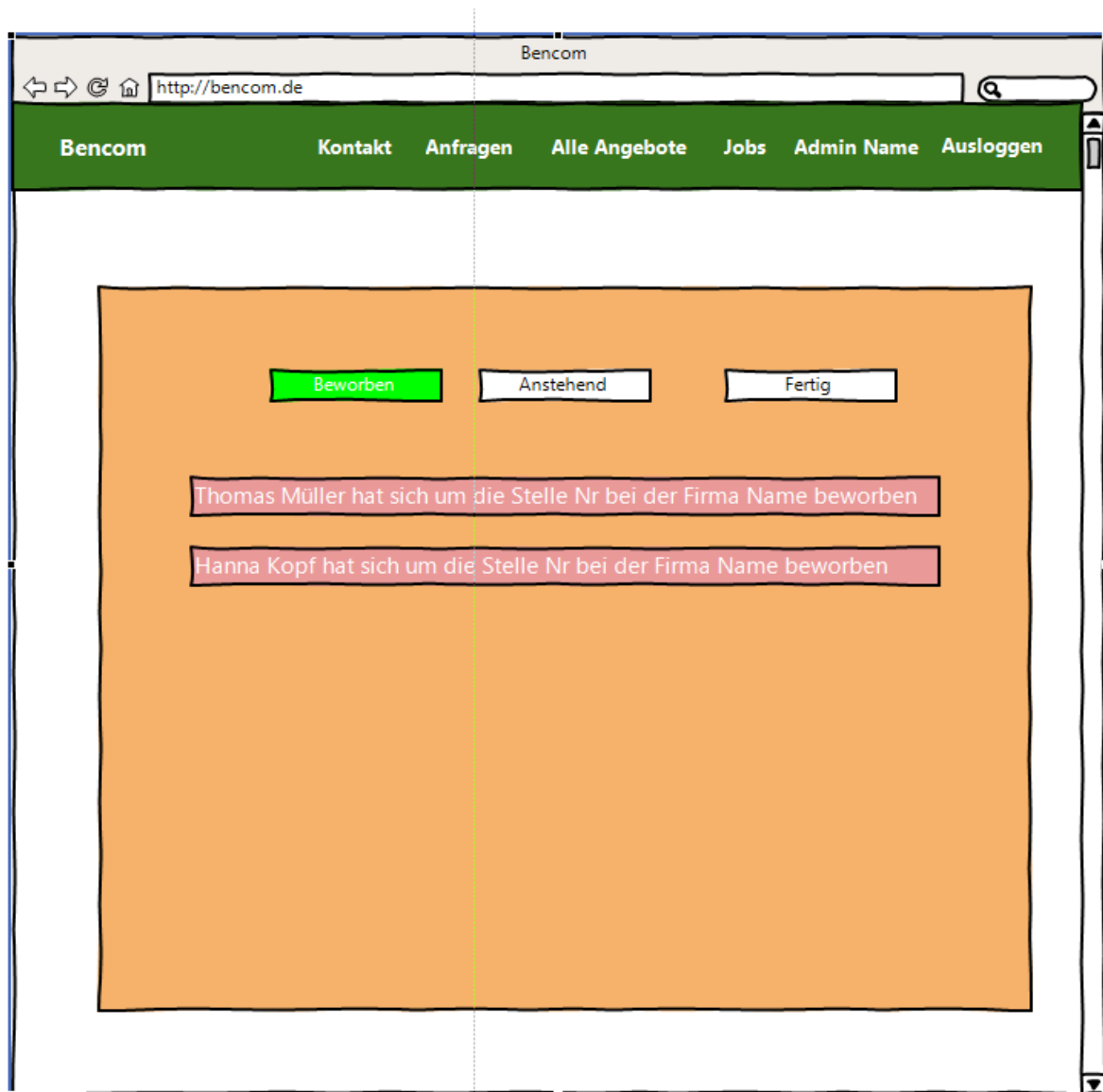


Abbildung 14: Jobs-Page-Admin Prototype

4.1.7 Home-Page-Student

In der Homepage für den Student wird die allen verfügbaren Angeboten als Komponenten angezeigt.

Jede Komponente hat allgemeine Informationen von der Stelle und einen Link, der „Details“ heißt und den Benutzer zu einer Seite, in der es die ganzen Informationen des Angebots bzw. Bewerbung Button gibt, weiterleitet. (Siehe Die Abbildung 15).

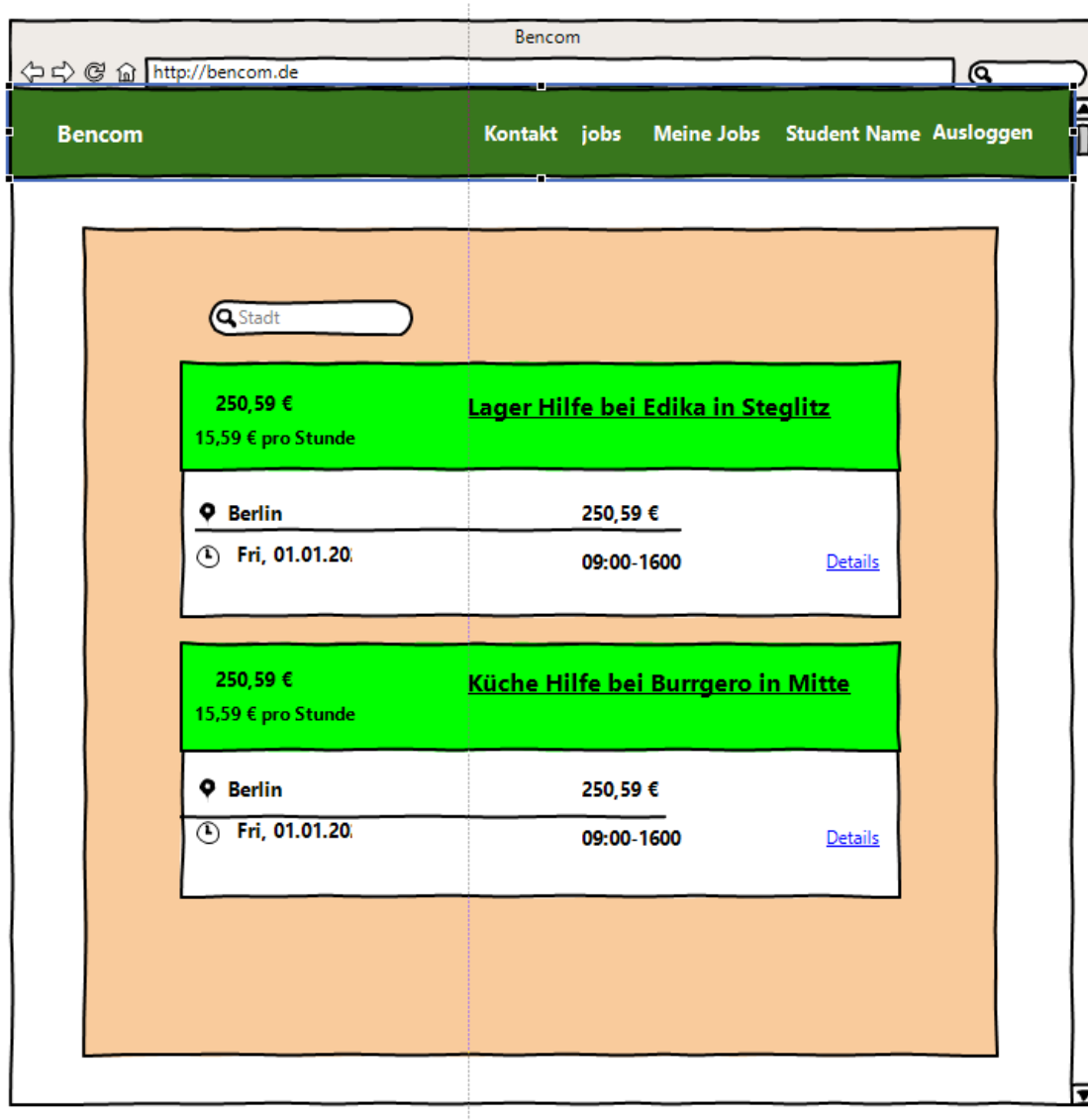


Abbildung 15: Home-Page-Student Prototyp

4.1.8 Offer-Details-Page-Student

Wenn der Benutzer, auf der Benutzer auf den Link „Details“ klickt, wird er zur Offer-Details-Seite weitergeleitet, wo die ganzen Informationen sich befinden, die er braucht zu wissen, wie zum Beispiel die Beschreibung, Anforderungen und Hinweise, und die Adresse der Stelle.

Es soll auch ein Bewerbung -Button geben, damit der Student sich bewerben kann, wenn er das Interesse dran hat. (Sieh die Abbildung 16)

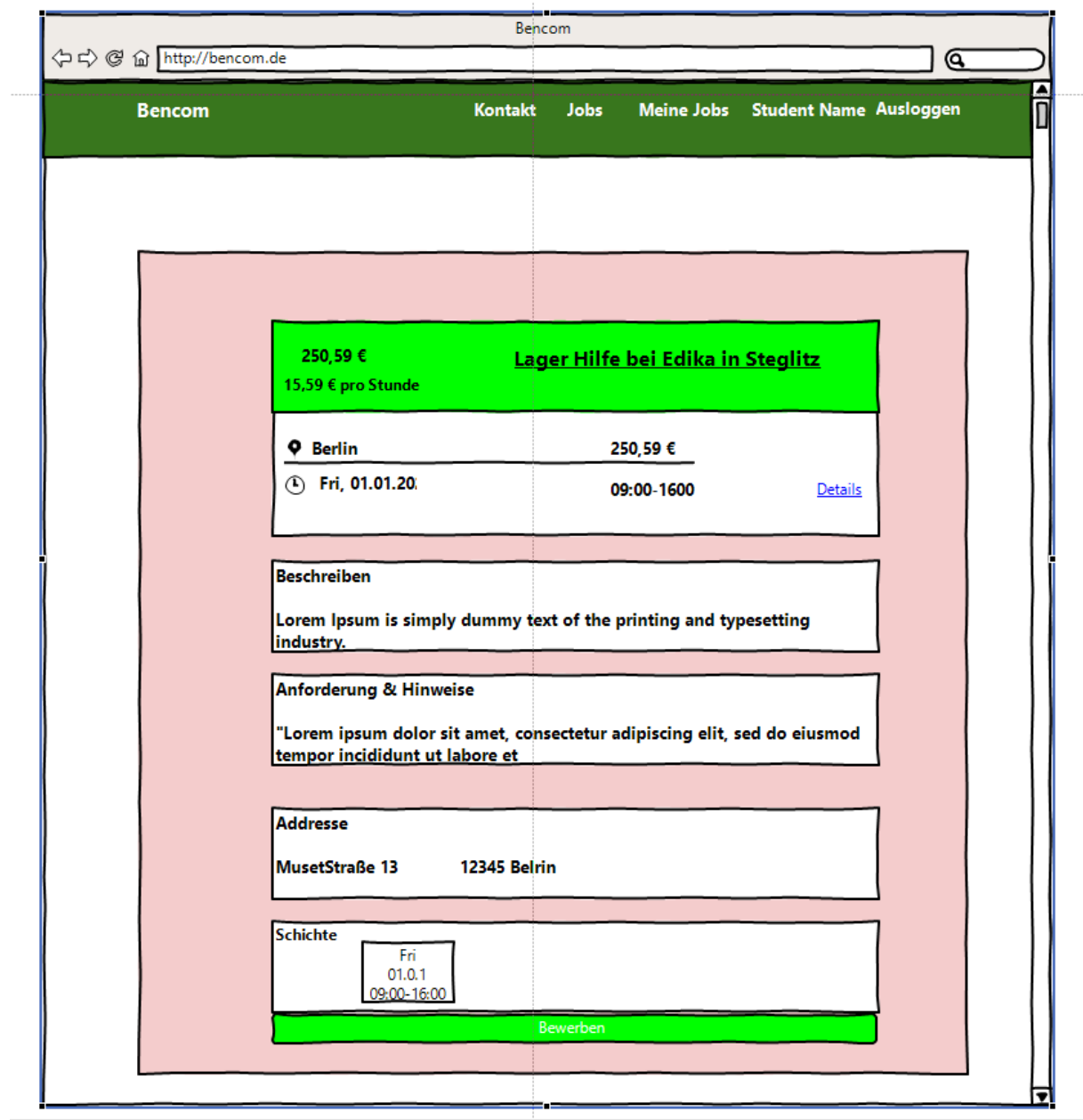


Abbildung 16: Offer-Details-Page-Student Prototyp

4.1.9 Meine-Jobs-Page-Student

Wie es in der „Abbildung 17“ gezeigt ist, sollen die Studenten die Zustände ihrer Bewerbungen genauso wie der Admin und die Firmen verfolgen können.

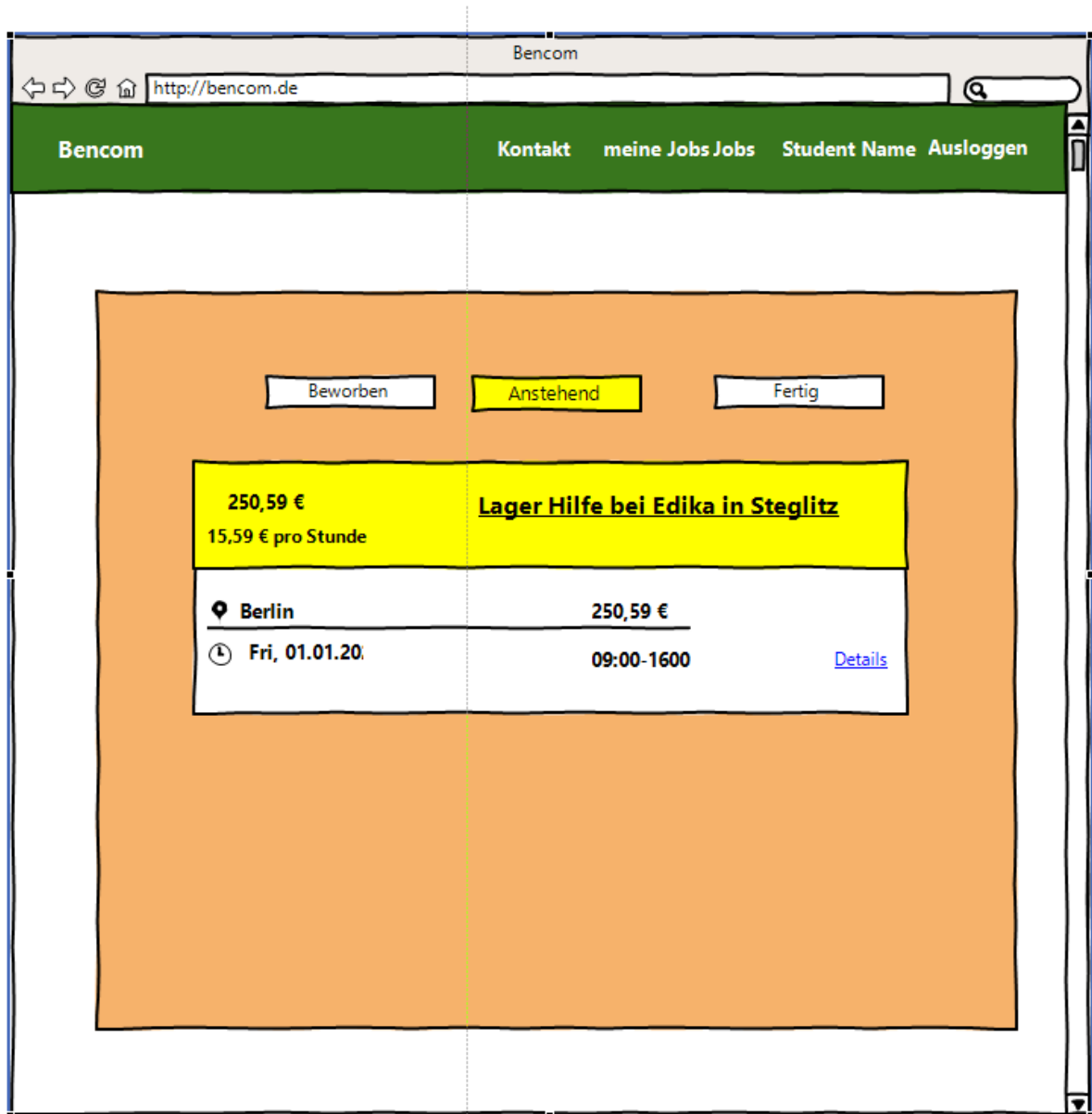


Abbildung 17: Meine Jobs-Page-Student Prototyp

4.1.10 Home-Page-Firma

Die Homepage Seite von den Firmen soll die Zustände der hinzugefügten Angebote, um die die Studenten sich beworben haben.

Darüber hinaus sollen die allen Angebote der Firma gezeigt werden.

Bei den anstehenden Bewerbungen soll auch ein „Fertig“ Button geben, mit dem kann die Firma nach dem erfolgreichen Einsatz den Zustand der Bewerbung zum „beendet“ ändern. (Siehe die Abbildung 18)

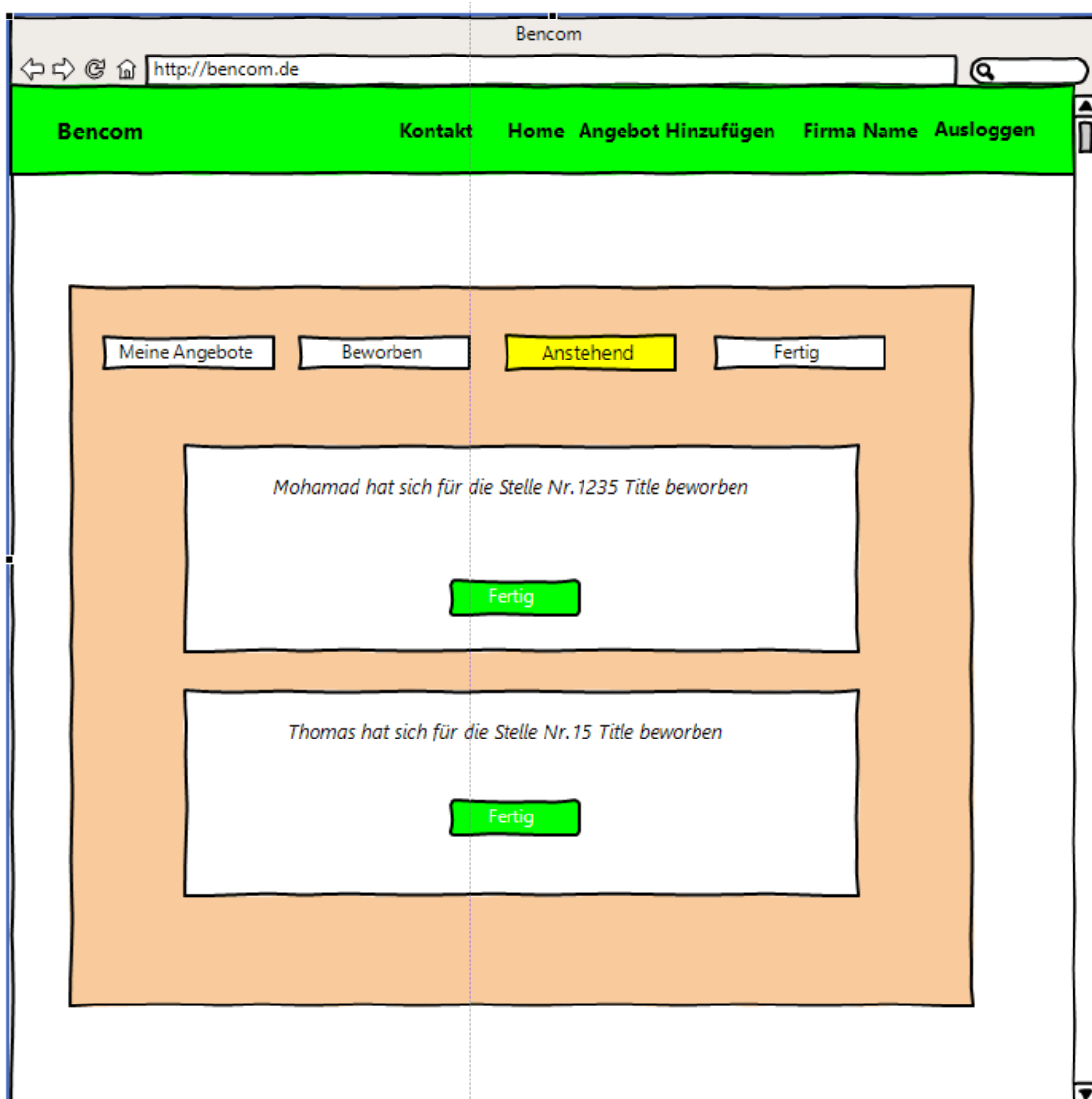


Abbildung 18: Homepage-Firme- Prototyp

4.1.11 Angebot-hinzufügen-Page-Firma

Es wurde ein Prototyp für die Seite „Angebot Hinzufügen“ erstellt, auf der die Firma die Informationen der neuen Stelle, die die Firma hinzufügen möchte, eingeben können soll.

(Siehe die Abbildung 19)

Bencom **Kontakt** **Home** **Angebot Hinzufügen** **Firma Name** **Ausloggen**

Title der Stelle

Wie Viele Tage? Pro Stunde €

Stadt Stadtteil

Wie viele Personen?

Beschreibung

Tag Wie viele Tage

Zeit-von Zeit-bis

Datum

Beschreibung

Hinweise & Anforderungen

Adresse

Straße Haus.Nr

Postleizahl Zusätzlich

Das Angebot Hinzufügen

Abbildung 19: Angebot-Hinzufügen-Firme- Prototype

4.2 ER-Diagramm

In der „Abbildung 20“ ist das ER-Diagramm zu sehen, das für die Datenbank erstellt wurde und zeigt, wie die Relationen zwischen den Tabellen sind.

Die Datenbank besteht aus vier Tabellen, und zwar „applied“, „student“, „company“ und „job“.

Jede Tabelle oder Entity hat eigenen Attributen und deren Datentyp und ob sie Primärschlüssel oder Fremdschlüssel, oder nicht.

Die Primärschlüssel sind mit dem gelben Schlüssel-Symbol und die Fremdschlüssel sind mit einer roten Route bezeichnet.

Die Relation zwischen den Tabellen „student“ und „applied“ ist eins und nur eins zu eins oder vielen, also, ein Student kann eine oder mehrere Bewerbungen haben.

Eine Firma kann ein oder mehreres Job und Bewerbungen haben.

Es kann sein, dass es eine oder mehrere Bewerbungen für jedes Angebot gibt. (Siehe die Abbildung 20)

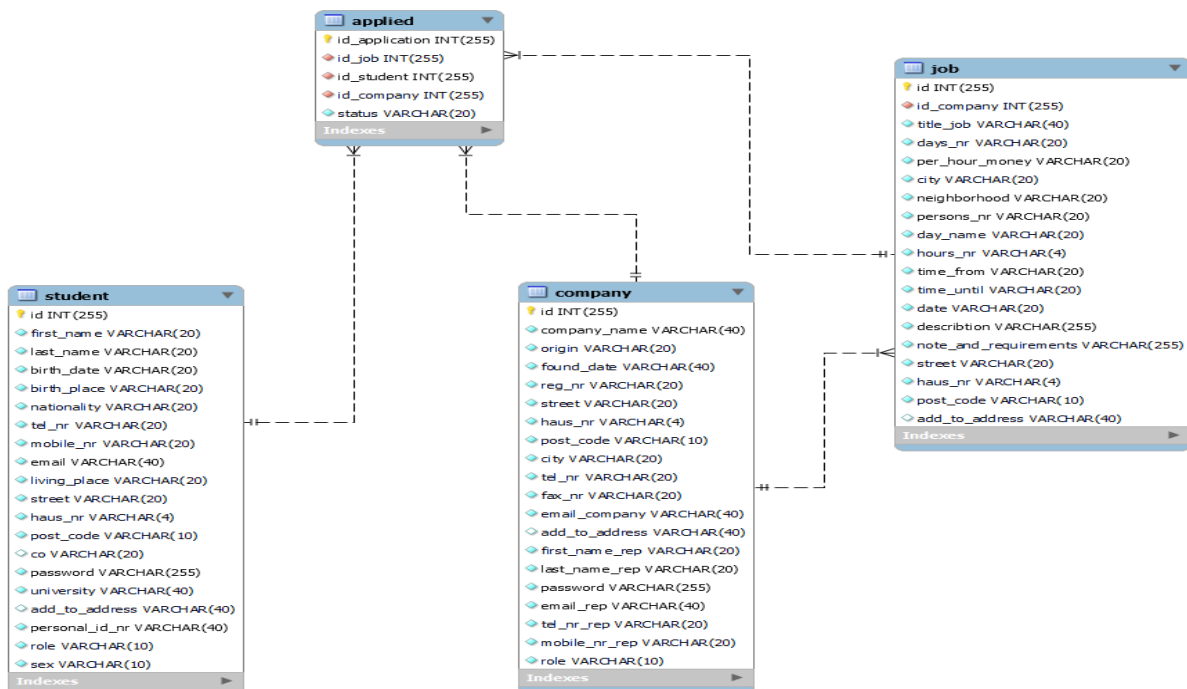


Abbildung 20: ER Diagramm für die Datenbank

4.4 Klassendiagramm

Das Klassendiagramm, das für diese Applikation erstellt wurde, besteht aus vier Klassen, und zwar „Student“, „Firma“, „Job“, und „Beworbenes Job“.

In der ersten Zeile steht der Name der Klasse, und in der Zweite Zeile stehen die Attribute. In der dritten Zeile liegen die Methoden.

Um die Sichtbarkeit zu bestimmen, ob sie Öffentlich oder privat ist, und zwar mit (+) für die Öffentlichkeit und mit (-) für die Privatsphäre.

Das Diagramm beschreibt die Relationen zwischen den Klassen.

Ein Student kann sich um ein oder mehrere Jobs bewerben. (eins zu eins oder vielen 1..*)

Eine Firma kann ein oder mehrere Angebote hinzufügen bzw. den Zustand eine oder mehrere Bewerbungen zu ändern. (eins zu eins oder vielen 1..*)

Der Admin kann den Zustand einer oder mehrere Bewerbungen ändern. (eins zu eins oder vielen 1..*)

(Siehe die Abbildung 21).

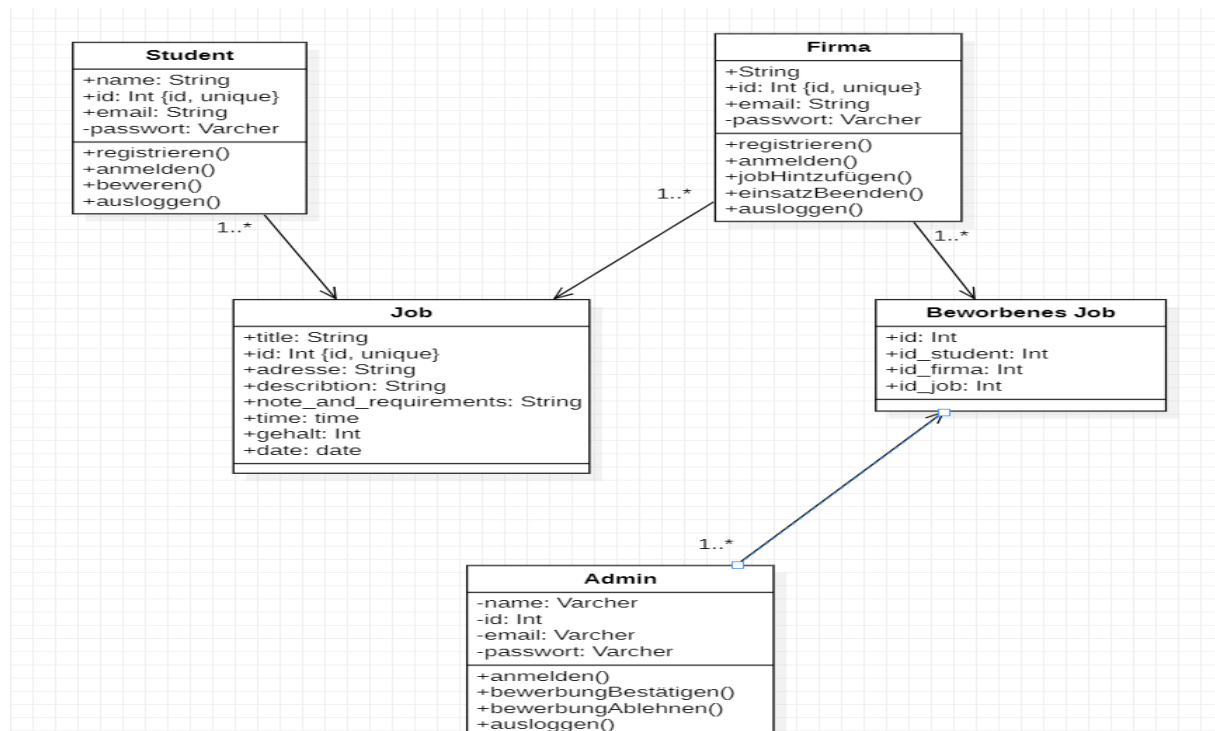


Abbildung 21: Klassendiagramm der Applikation

4.3 Aktivitätsdiagramme

Aktivitätsdiagramm wird verwendet, um den Ablauf bestimmter Anwendungsfälle der Benutzer darzustellen und die allen Aktivitäten in der Applikation zu modellieren.

Für Admin, Student und Firma sind drei Aktivitätsdiagramme in dieser Bachelorarbeit erstellt worden.

4.3.1 Student

Der Anwendungslauf des Studenten fängt mit der Registrierung und der Anmeldung an, und nach der erfolgreichen Anmeldung entweder sucht er nach passenden Angeboten oder verfolgt den Zustand seine Bewerbungen. Wenn er ein passendes Angebot findet, kann sich darum bewerben.

Die allen Anwendungsfälle sind in einem Diagramm dargestellt.

(Siehe die Abbildung 22)

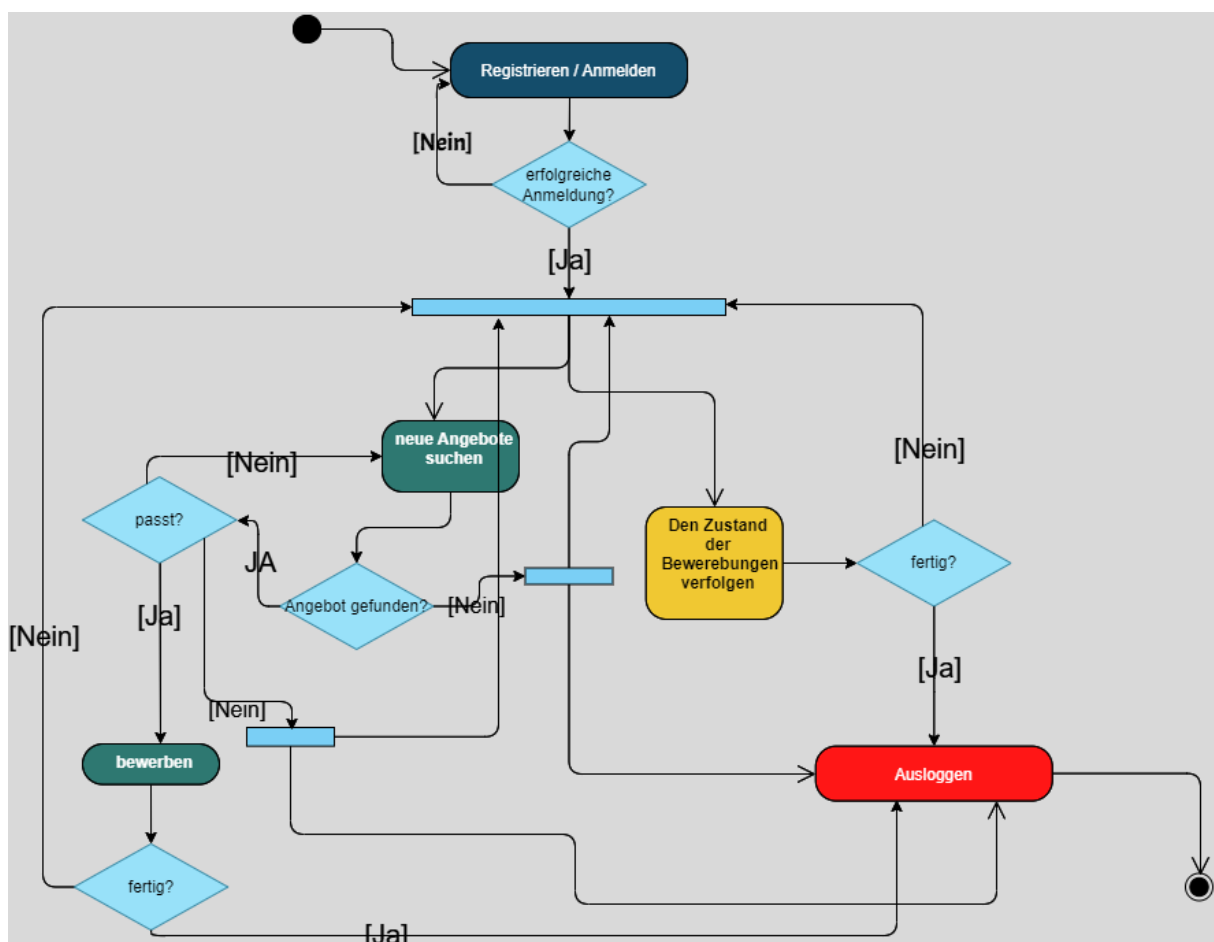


Abbildung 22: Aktivitätsdiagramm für Admin

4.3.2 Firma

Es wurde eine Aktivitätsdiagramm erstellt, die die allen Aktivitäten der Firmen in der App beschreibt.

Der Benutzer soll sich am Anfang registrieren oder anmelden, und dann hat er zwei Möglichkeiten, entweder den Zustand der Bewerbungen zu verfolgen oder ein neues Angebot hinzufügen.

Nach dem Verfolgen der Zustände der Bewerbungen, wenn es einen beendeten Einsatz gibt, kann er den Zustand der Bewerbung zum „beendet“ ändern.

Wenn der Benutzer mit der Benutzung der Applikation fertig ist, kann er sich ausloggen.

(Siehe die Abbildung 23)

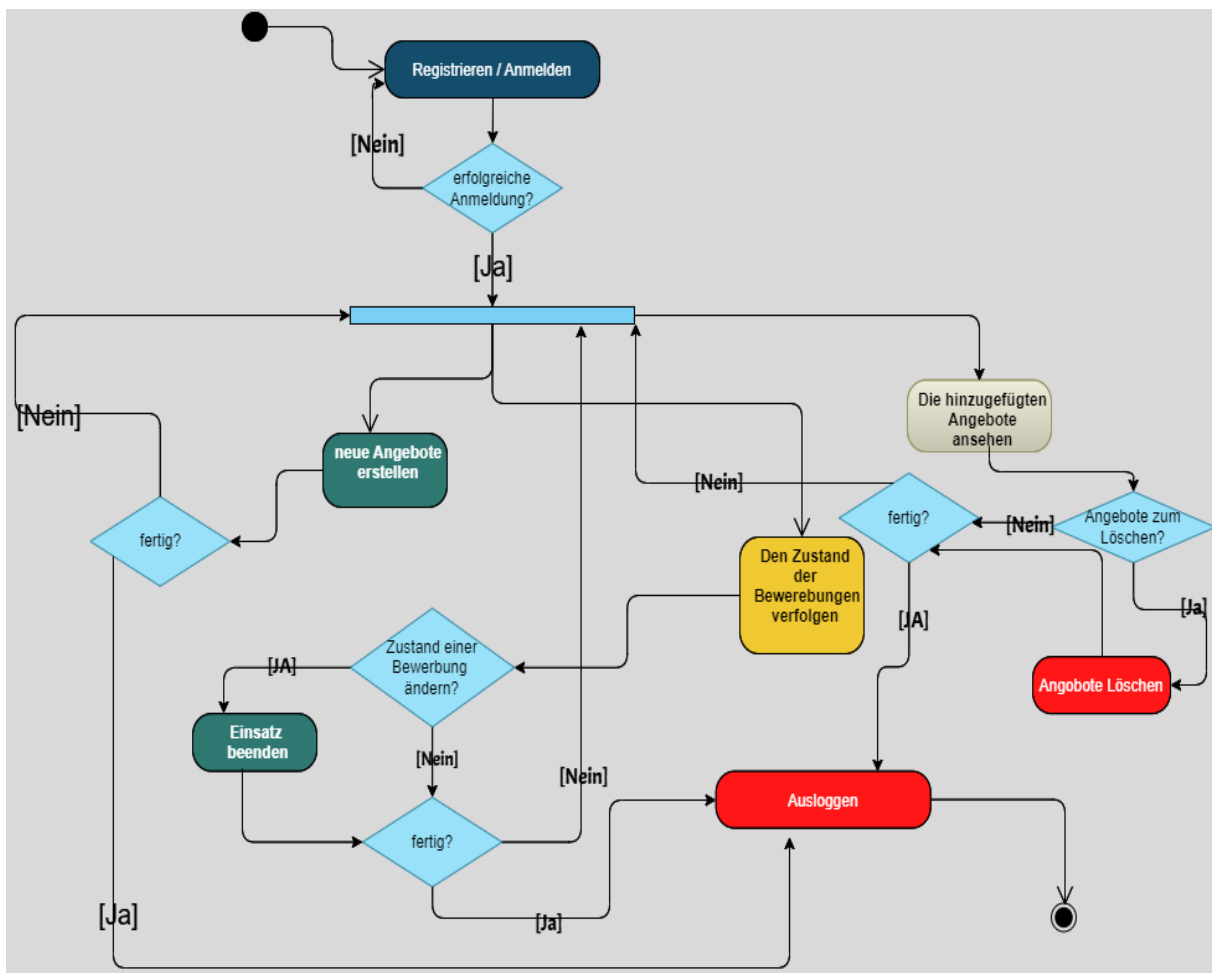


Abbildung 23: Aktivitätsdiagramm für Firma

4.3.3 Admin

Die letzte Aktivitätsdiagramm in dieser Arbeit ist für Admin.

Sie beschreibt die Anwendungsfälle von Admin in dieser Applikation.

Zum Anfang soll sich der Admin anmelden, dann ist er in der Lage, zwei Entscheidungen zu treffen, entweder die Zustände der Bewerbungen zu verfolgen oder zu prüfen, ob es neue Bewerbungen gibt oder nicht.

Wenn es neue Bewerbung/en gibt, kann er die Anfrage entweder bestätigen oder ablehnen, und wenn der Admin mit der App fertig ist, kann er sich ausloggen.

(Siehe die Abbildung 24)

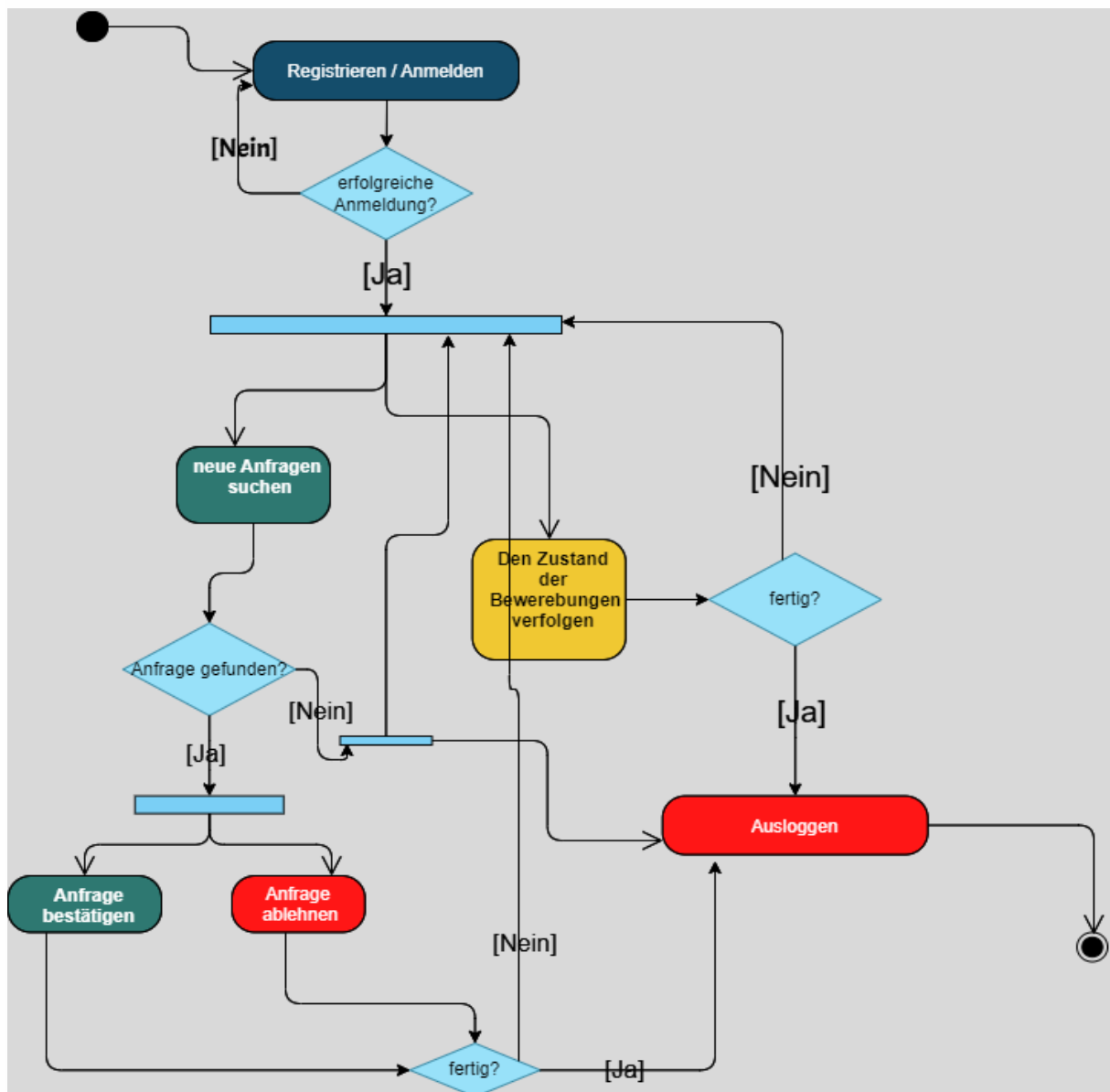


Abbildung 24: Aktivitätsdiagramm für Admin

Kapital 5

Implementierung

In diesem Kapitel wird das zuvor geplante Entwurf der Applikation implementiert, und Codeausschnitte des Quellcodes dargestellt.

Im Lauf der Implementierung wurde „Git“ verwendet.

Git ist ein Versionierung Open-Source-Tool, mit dem die Entwickler den gesamten Code verwalten bzw. in einer Datenbank abspeichern können.

Mit Git kann der Code mit den anderen Entwicklern geteilt, und von denen verfolgt werden.

In der „Abbildung 25“ wird gezeigt wie der Struktur der App, Backend, Frontend, und die Datenbank.

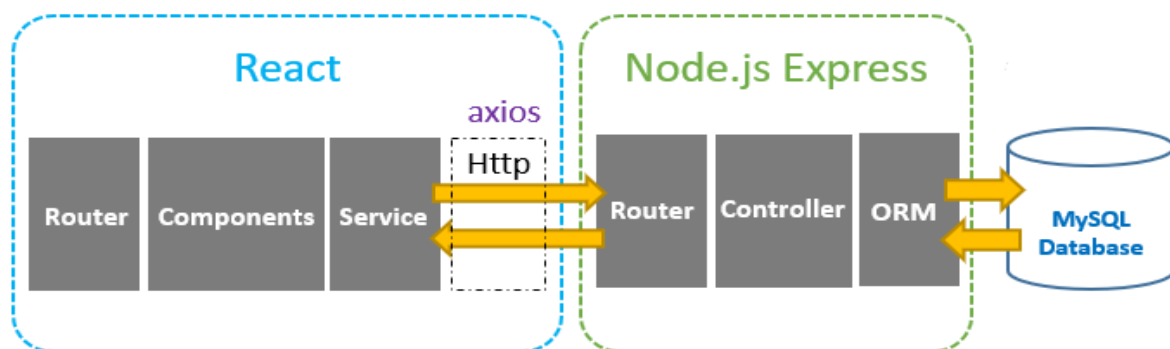


Abbildung 25: Struktur der Applikation [E1]

4.1 Frontend

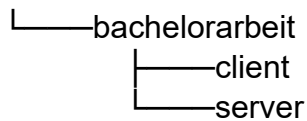
Als „Single-Page-App“ mit React.JS wurde das Frontend der Applikation implementiert.

Single-Page-App bedeutet, dass die Applikation aus nur einem einzigen HTML-Dokument, das den Code dynamisch nachladen, besteht.

Zum Anfang wurde das Development Environment von React.JS installiert und ausgeführt, indem bestimmte Befehle eingegeben werden.

```
npx create-react-app bachelorarbeit
cd bachelorarbeit
npm start
```

Der Projektstruktur sieht so wie der untere Baum aus. Ein Ordner „client“ für das Frontend und der andere „server“ für das Backend.



Die Quelldateien der Frontseite der Applikation sind in der folgenden Ordnerstruktur dargestellt. (Siehe die Abbildung 26)

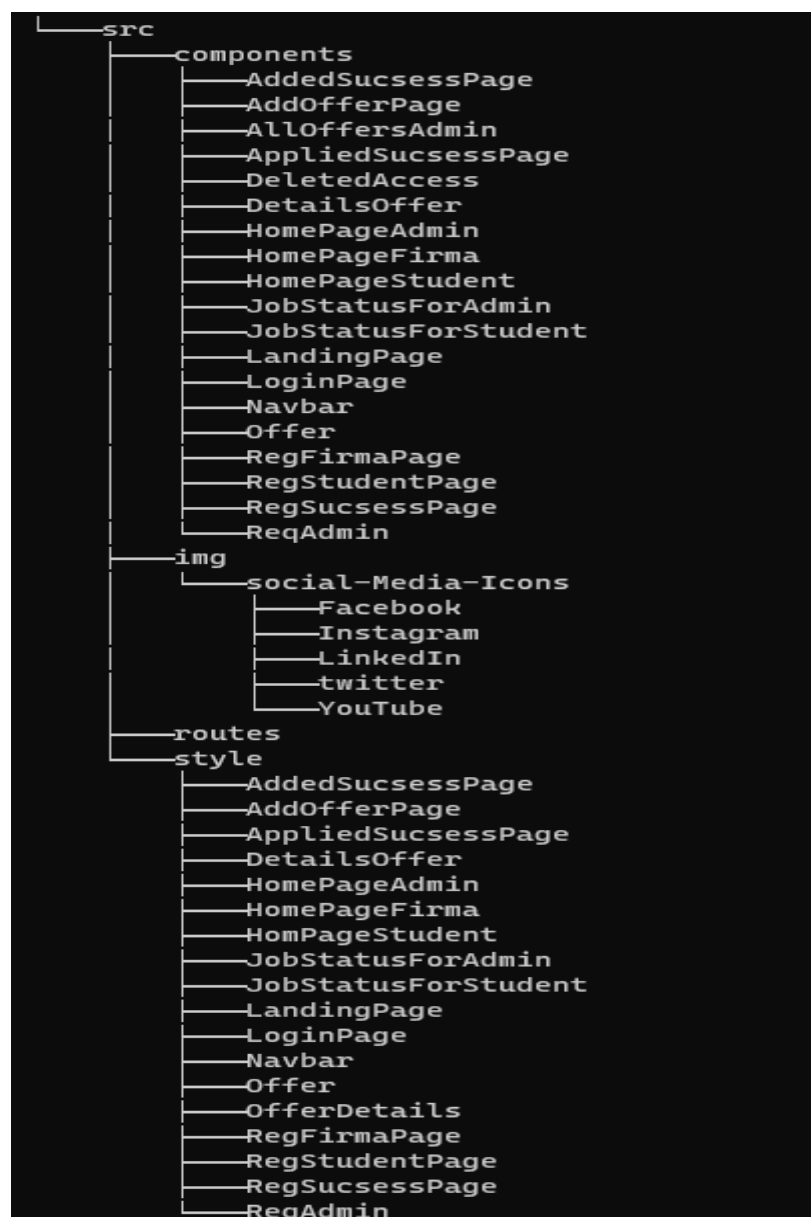


Abbildung 26: Der Verzeichnisstruktur der Frontendseite

Für Stylesheet wurde SASS die Seiten verwendet, indem wurde das „sass“ mit „npm“
`„$ npm install sass“` installiert.

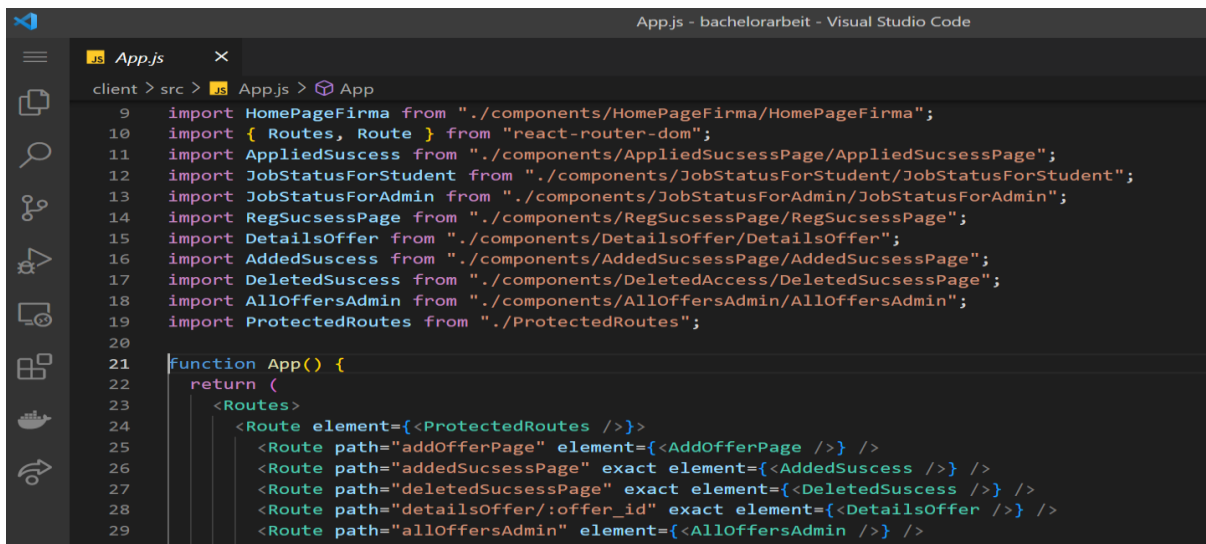
NPM ist ein Paketmanager, das für die JavaScript-Laufzeitumgebung Node.JS verwendet wird. [E2]

Es wurden Codeausschnitten vom Frontend als Beispiele genommen.

In der „Abbildung 27“ wird es gezeigt, wie die Komponenten importiert wurden, um die ausführen zu lassen und in Routes Komponente zu packen.

Routes Komponente enthält Route Komponente, welche ProtectedRoutes Komponente hat, die zum Schützen der Routen erstellt wurde.

Route packt die allen Routen der App die mit dem Path und die Komponente definiert, die im Browser geladen wird.



```
App.js - bachelorarbeit - Visual Studio Code
client > src > App.js > App
9   import HomePageFirma from "../components/HomePageFirma/HomePageFirma";
10  import { Routes, Route } from "react-router-dom";
11  import AppliedSuccessPage from "../components/AppliedSuccessPage/AppliedSuccessPage";
12  import JobStatusForStudent from "../components/JobStatusForStudent/JobStatusForStudent";
13  import JobStatusForAdmin from "../components/JobStatusForAdmin/JobStatusForAdmin";
14  import RegSuccessPage from "../components/RegSuccessPage/RegSuccessPage";
15  import DetailsOffer from "../components/DetailsOffer/DetailsOffer";
16  import AddedSuccess from "../components/AddedSuccessPage/AddedSuccessPage";
17  import DeletedSuccess from "../components/DeletedAccess/DeletedSuccessPage";
18  import AllOffersAdmin from "../components/AllOffersAdmin/AllOffersAdmin";
19  import ProtectedRoutes from "../ProtectedRoutes";
20
21  function App() {
22    return (
23      <Routes>
24        <Route element={<ProtectedRoutes />} />
25        <Route path="addOfferPage" element={<AddOfferPage />} />
26        <Route path="addedSuccessPage" exact element={<AddedSuccess />} />
27        <Route path="deletedSuccessPage" exact element={<DeletedSuccess />} />
28        <Route path="detailsOffer/:offer_id" exact element={<DetailsOffer />} />
29        <Route path="allOffersAdmin" element={<AllOffersAdmin />} />
30      </Routes>
    );
  }
  export default App;
```

Abbildung 27: Screenshot des Codeausschnittes von Routen und Importieren

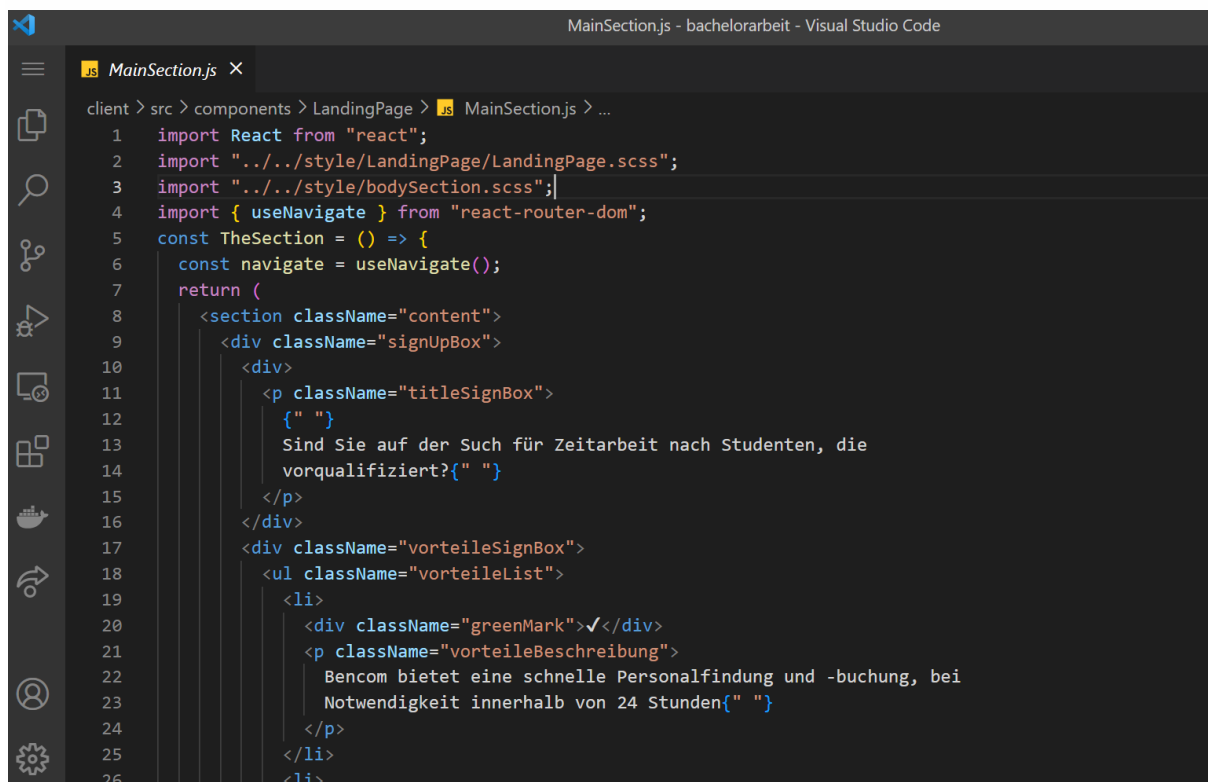
In der „Abbildung 28“ sieht ganz oben wie die Dateien mit dem Schlusswort „import“ in React.JS importiert werden.

In der Zeile 4 wird useNavigate-Hook-Funktion, mit der man navigieren kann.

Dann wurde die Arrow-Funktion-Komponente „theSection“ definiert, welche JSX und HTML5 zurückgibt und exportiert wurde.

JSX ist eine JavaScript-Erweiterung, welche bei der Entwicklung mit React.JS empfohlen ist.

Für jede Klasse in der React.JS-Applikation soll einzigen Namen, der nur einmal existiert darf, ansonsten wird ein Zusammenstoß mit dem Stylesheet treten.



```
client > src > components > LandingPage > MainSection.js > ...
1  import React from "react";
2  import "../../style/LandingPage/LandingPage.scss";
3  import "../../style/bodySection.scss";
4  import { useNavigate } from "react-router-dom";
5  const TheSection = () => {
6      const navigate = useNavigate();
7      return (
8          <section className="content">
9              <div className="signUpBox">
10                  <div>
11                      <p className="titleSignBox">
12                          { " "}
13                          Sind Sie auf der Such für Zeitarbeit nach Studenten, die
14                          vorqualifiziert?{ " "}
15                      </p>
16                  </div>
17                  <div className="vorteileSignBox">
18                      <ul className="vorteileList">
19                          <li>
20                              <div className="greenMark">✓</div>
21                              <p className="vorteileBeschreibung">
22                                  Bencom bietet eine schnelle Personalfindung und -buchung, bei
23                                  Notwendigkeit innerhalb von 24 Stunden{ " "}
24                              </p>
25                          </li>
26                          <li>
```

Abbildung 28: Screenshot des Codeausschnittes von HTML5

React.JS hat sehr hilfreiche Erweiterung und sie heißt useState-Hook, die den Entwicklern dabei helfen, den aktuellen Zustand in einer Komponente zu verfolgen.

Der „Abbildung 29“ zeigt ein Beispiel von der Verwendung der useState Erweiterung in der Arbeit.

In der Zeile 12 wurde ein useState-Hook für die Fehlermeldung bei der Validierung der Eingaben in einer Form definiert.

Der erste Wert „formErrors“ ist der aktuelle Zustand.

Und der Zweite Wert setFormErrors ist eine Funktion zum Aktualisieren der Zustand laut der Eingabe der Benutzer.

Zum Initialisieren der Wert von formErrors und ihr Datentyp, und hier wurde leerer String bestimmt.

```

11
12     const [formErrors, setFormErrors] = useState("");
13     const [isSubmit, setIsSubmit] = useState(false);
14

```

Abbildung 29: Screenshot des Codeausschnittes „useState-Hook“

Von der Zeile 13 bis 15 in der „Abbildung 30“ Variablen definiert, die die Werte die im lokalen Speicher des Browsers, die nach der erfolgreichen Anmeldung abgespeichert wurden, enthalten.

Mit der Hilfe des Objektes localStorage und ihrer getItem Methode, kann man die Werte des lokalen Speichers laut dem Schlüssel vom Browser zugreifen.

Ein andres React.JS-Hook wurde auch verwendet, und zwar useEffect.

Das useEffect-Hook ist zum Ausführen der Nebeneffekte in der Komponente.

Es kann zwei Argumente haben, eine ist erforderliche Funktion, und die andere optionale ist die Abhängigkeit.

In diesem Beispiel wurde die Abhängigkeit mit leerem Array bestimmt und das heißt, dass es nur beim ersten Rendern.

Und was wird in diesem Beispiel in useEffect-Hook ausgeführt, ist eine Get-HTTP-Anfrage, die an den Server geschickt wird, um eine Antwort mit Daten vom Server zu erhalten. Die Anfrage schickt auch Daten wie Token und ID, die in „params“ und „headres“ sind.

```

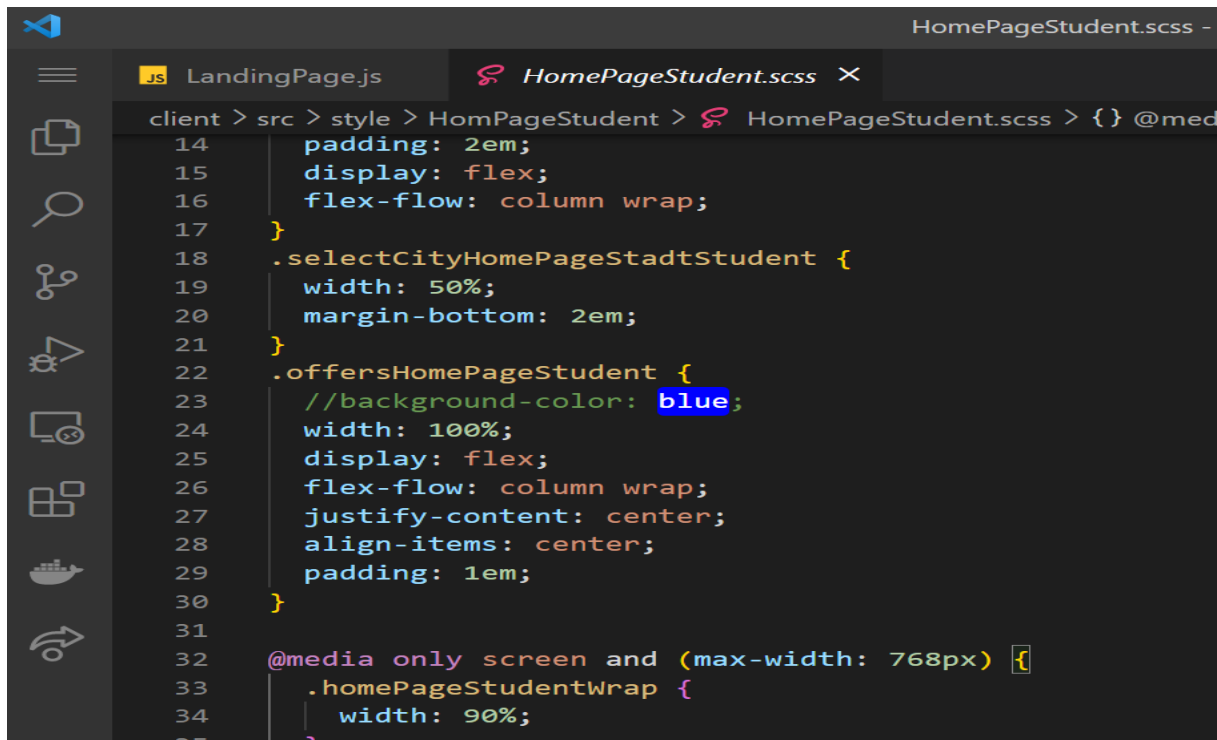
13     const userId = localStorage.getItem("userId");
14     const tokenUser = localStorage.getItem("token");
15     const user_role = localStorage.getItem("role");
16     useEffect(() => {
17         setLoading(true);
18
19         Axios.get(`http://localhost:4000/api/v1/offer/getDetails/${offer_id}`, {
20             params: { userId: userId },
21             headers: { "x-auth-token": `${tokenUser}`, role: `${user_role}` },
22         })
23         .then((res) => {
24             setOffersData(res.data);
25             setLoading(false);
26         })
27         .catch((err) => {
28             console.log(err);
29             setLoading(false);
30         });
31     }, []);
32

```

Abbildung 30: Screenshot des Codeausschnittes von „useEffect-Hook“

In „der Abbildung“ 31 ist ein SASS-Code-Abschnitt, und er es zeigt, wie Stylesheet mit dem Klassennamen gemacht wird, und es wurde "Media Queries" verwendet, um Responsives Webdesign laut der Größe des Bildschirmes zu entwickeln.

Es wird ausgeführt, wenn die Bedingung wahr ist.



```
client > src > style > HomPageStudent > HomePageStudent.scss > {} @med
14 padding: 2em;
15 display: flex;
16 flex-flow: column wrap;
17 }
18 .selectCityHomePageStadtStudent {
19 width: 50%;
20 margin-bottom: 2em;
21 }
22 .offersHomePageStudent {
23 //background-color: blue;
24 width: 100%;
25 display: flex;
26 flex-flow: column wrap;
27 justify-content: center;
28 align-items: center;
29 padding: 1em;
30 }
31
32 @media only screen and (max-width: 768px) {
33 .homePageStudentWrap {
34 width: 90%;
35 }
```

Abbildung 31: Screenshot des Codeausschnittes von SASS

4.2 Backend

Als Server für das Backend wurde mit Node.JS bzw. Express.JS entwickelt.

Zuerst müssten Node.JS auf dem Rechner installiert werden.

Nachdem der „Server“ Ordner erstellt worden war, wurde die „**package.json**“ darunter mit dem Befehl „**\$ npm init**“ erstellt.

Die Datei „**package.json**“ enthält die Informationen über das Projekt wie den Namen, Versionen, Schreiber usw.

In der „Abbildung 32“ wird der Ordnerstruktur der Quelldateien der Backendseite der Applikation dargestellt.

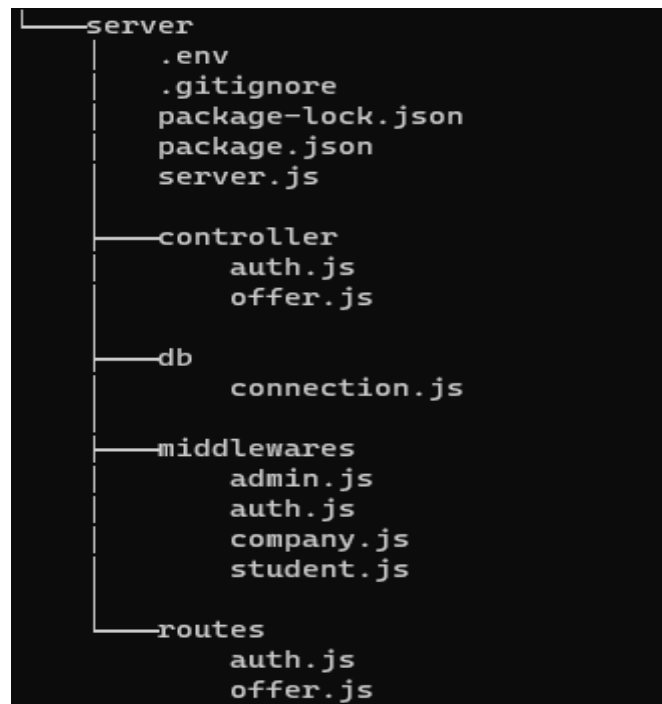


Abbildung 32: Der Verzeichnisstruktur der Backendseite

Der Inhalt der Server.JS-Datei wird in der „Abbildung 33“ gezeigt.

Zuerst wurde in der Zeil „Express“ importiert und in Variabel „app“ definiert, um den Das Routing zu erstellen.

Routing basiert sich auf die Endpunkte (URLs), die vom Client abgerufen wird.

Mit der Funktion „**app.method(path, handler)**“ werden die Routen in Express.JS definiert.

Method könnte als eine von den HTTP-Methoden „CRUD“ ausgeführt werden.

In der folgenden Tabelle wird die CRUD-Methode in Express.JS dargestellt.

CRUD	Express.JS
Create / Erstellen	Post
Read / Lesen	Get
Update / Verändern	Put
Delete / Löschen	Delete

Tabelle 1: Die CRUD-Operationen in Express.JS

Die „cors-Middleware“ gibt dem Server die Möglichkeit, die Anfragen von verschiedenen Domänen zu erhalten.

Mit „cors“ kann man für Sicherheitsgründe einen oder mehrere Domänen bestimmen, damit sie die Anfragen an den Server schicken darf.

Die Anfragen und Antworten werden als JSON-Datei geschickt und erhalten.

In der Zeile 6 wurde die „morgan-Middleware“ importiert, um bei der Entwicklung zu debuggen bzw. Log-Dateien in der Konsole zu erstellen.

Die Endpunkte wurden mit dem Path und einer Callback-Funktion, die von „routes-Ordner“ importiert werden, definiert.

Mit „listen-Methode“ wird der Server auf Port: 4000 ausgeführt.
(Siehe die Abbildung 33)

Die Routen wurden unter „routes“ Ordner in zwei Dateien erstellt, und zwar eine für die Authentication ⁽³⁾ und die andere für die Operationen mit den Angeboten in der Applikation.

In der „Abbildung 34“ sieht man ein Beispiel von Routen, die in dieser Arbeit erstellt wurden, und hat als Parameter Path, eine Middleware für Authorization, und die Funktion, die vom „Controller“ importiert, und nach der erfolgreichen Authorization ausgeführt wird.

```
2   const express = require("express");
3   var morgan = require("morgan");
4   const app = express();
5   const PORT = 4000;
6   const cors = require("cors");
7
8
9   app.use(morgan("dev"));
10
11  // cors als Middleware
12  app.use(cors());
13
14  require("dotenv").config();
15
16  // Bodyparser
17  app.use(express.json());
18  app.use(express.urlencoded({ extended: false }));
19
20  // Endpoints
21  app.use("/api/v1/auth", require("./routes/auth"));
22  app.use("/api/v1/offer", require("./routes/offer"));
23
24  // server start
25  app.listen(PORT, () => {
26    console.log(`The app listening on Port ${PORT}`);
27  });
```

Abbildung 33: Codeabschnitt von der server.js Datei im Backend

⁽³⁾ Es wird damit geprüft, wer der Benutzer ist.

Abbildung 34: Codeabschnitt von Routen der offer.js Datei

Das JWT wird nach der Erfolgreichen Anmeldung erstellt und vom Server an den Client geschickt.

In der „Abbildung 36“ wird gezeigt, wie die Autorisierungscode von Token läuft und das Token vom Client geprüft wird.

```
server > middlewares > JS auth.js > <unknown> > exports
1  "use strict"
2  const JWT = require("jsonwebtoken");
3
4  /**
5   * @desc      Auth middleware
6   */
7  module.exports = (req, res, next) => {
8      const token = req.header("x-auth-token");
9      //console.log(token);
10     if (!token)
11         return res.status(401).json({
12             message: "Access denied. No token provided.",
13         });
14
15     try {
16         const decoded = JWT.verify(token, "jwtSecret");
17
18         req.user = decoded;
19
20         next();
21     } catch (ex) {
22         res.status(400).json({
23             message: "Invalid token",
24         });
25     }
26 };
```

Abbildung 36: Der Verzeichnisstruktur der Backendseite

Es wird zuerst geprüft, ob der ein Token vom Browser erhalten oder nicht, und wenn ja, wird das Token mit der Signatur validiert.

Wenn es bestätigt wurde, wird die Funktion weiter durchgeführt, ansonsten wird der Server einen Fehler mit dem Statuts Code: 400 ⁽⁴⁾ an den Client schicken werden.

Es steht in der „Abbildung 37“ ein Beispiel von einem Token, das in der App erstellt wurde. Auf der linken Seite ist verschlüsselt und auf der rechten unverschlüsselt

⁽⁴⁾ Der Status Code: 400 „**Bad Request**“ bedeutet, dass der Server die Anfrage nicht behandeln kann oder will.

Encoded

PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6Nywicm9sZSI6ImFkbWluIiwibmFtZSI6IkpvaGFtYWQgQWxpIiwiaWF0IjoxNjYyOTI2OTc3fQ.BfLBwcZ8dNpQ3TdSbznZJhWsG-OpCSr30xHjWnE-WB4
```

Decoded

EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

PAYLOAD: DATA

```
{
  "id": 7,
  "role": "admin",
  "name": "Mohamad Ali",
  "iat": 1662926977
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) ☒ secret base64 encoded
```

Signature Verified

SHARE JWT

Abbildung 37: Screenshot vom Token Encoded-Decoded

Es wurde wie bereits genannt eine Middleware zur Prüfung der Rolle des Benutzers, da jeder Benutzer laut seiner Rolle beschränkte Operationen ausführen lassen.

Wie es in der „Abbildung 38“ zu sehen ist, steht ein Abschnittscode von Middleware, die es prüft, ob die Rolle des Benutzers „company“ ist, und wenn ja, dann hat er den Zugriff darauf, die Operation weiter auszurufen.

```
server > middlewares > JS company.js > <unknown> > <unknown> exports
1  "use strict"
2
3
4  /**
5   * @desc    Auth middleware
6   */
7  module.exports = (req, res, next) => {
8    if (req.header("role") !== "company")
9      return res.status(403).json({
10       message: "Access denied.",
11     });
12    next();
13  };

```

Abbildung 38: Codeabschnitt von Rollenautorisierung in company.JS Datei

Im Folgenden werden Die CRUD-Operationen-Funktionen, die Im Backend entwickelt wurde, in Abschnittscode gezeigt.

In der folgenden Tabelle ist Die CRUD-Operationen in MySQL:

CRUD	MySQL
Create / Lesen	INSERT
Read / Lesen	SELECT
Update / Veränderung	UPDATE
Delete / Löschen	DELETE

Tabelle 2: Die CRUD-Operationen in MySQL

In der "connection-Datei" wurde eine Funktion erstellt, mit der der Server mit dem Datenbankserver verbindet, um Anfrage zu schicken und Antworten zu erhalten.

Die „createPool-Methode“ enthält die Zugangsinformationen wie den Namen der Datenbank, Hostname und Passwort. (Siehe die Abbildung 39)

```
server > db > Js connection.js > ...
  1  const mysql = require("mysql");
  2
  3  const db = mysql.createPool({
  4    user: "root",
  5    host: "localhost",
  6    password: "",
  7    database: "bencom",
  8  });
  9  module.exports = db;
```

Abbildung 39: Codeabschnitt von der Erstellung der Verbindung mit der Datenbank

Die CRUD-Operationen:

Codeausschnitte von CRUD-Methode mit Express.JS, die in dieser Arbeit entwickelt wurden, werden als Beispiele in Folgendes gezeigt.

Der Client schickt HTTP-CRUD-Anfragen mit Daten in deren „headers, body, und params“, die der Server braucht um die Operation auszuführen.

1.Create / Erstellen:

Wie in der „Abbildung 40“ zu sehen ist, gibt es eine Post-Methode, die dabei ausgeführt wird, wenn ein Student sich um ein Angebot bewirbt.

Der Server bekommt im Body der Anfrage die Daten, die an den Datenbankserver und applied-Tabelle geschickt werden, wie id_job, id_student usw.

Danach wird eine Query-Methode, die im ersten Argument einen MySQL-Daten einfügen-Befehl, im zweiten in Array die Daten, und im dritten eine callback-Funktion.

Die Daten werden in die Tabelle „applied“ gepostet werden, wenn es keinen Fehler gibt. Ansonsten schickt der Server ein Error mit Status Code 200.

```
82 module.exports.applayOffer = (req, res) => {
83   const id_job = req.body.id_job;
84   const id_student = req.body.id_student;
85   const id_company = req.body.id_company;
86   const status = req.body.status;
87
88   db.query(
89     "INSERT INTO applied (id_job,id_student, id_company ,status) VALUES (?,?,,?)",
90     [id_job, id_student, id_company, status],
91     (err, result) => {
92       if (err) {
93         console.log(err);
94         return res.status(400).json({ message: "error" });
95       } else {
96         return res.status(200).send("Values Inserted");
97       }
98     }
99   );
100 }
```

Abbildung 40: Codeabschnitt von applayOffer-Funktion-POST

2.Read / Lesen:

Wenn der Admin die Seite „Alle Angebote“ besucht, wird eine Get-Anfrage an den Server geschickt, um die allen Angebote in der Tabelle „job“ abgerufen, und der Server sickt eine Abfrage an den Datenbankserver, und schickt die Antwort weiter an Client. (Siehe die Abbildung 41)

```
55 module.exports.getAllOffers = (req, res) => {
56   db.query("SELECT * FROM job", (err, result) => {
57     if (err) {
58       console.log(err);
59       return res.status(400).json({ message: "error" });
60     } else {
61       console.log(result);
62       return res.status(200).send(result);
63     }
64   });
65 };
66
```

Abbildung 41: Codeabschnitt von getAllOffers-Funktion-GET

3.Update / Veränderung:

Wenn der Admin eine Bewerbung bestätigt hätte, soll der Zustand der Bewerbung von "applied" zum „anstehend“ geändert werden.

Es wird eine PUT-Methode ausgeführt, die eine Callback-Funktion „approveReq“ ausführt.

Der Server bekommt das ID von der Bewerbung mit Body der von Client PUT-Anfrage geschickt wurde.

Dann einen UPDATE-Befehl, um den Zustand zum „upComming“ laut dem ID der Bewerbung zu aktualisieren. (Siehe die Abbildung 42)

```
136 module.exports.approveReq = (req, res) => {
137   const id_application = req.query.id_application;
138   const appliedStatus = "upComming";
139   db.query(
140     "UPDATE applied SET status = ? WHERE id_application = ?",
141     [appliedStatus, id_application],
142     (err, result) => {
143       if (err) {
144         console.log(err);
145         return res.status(400).json({ message: "error" });
146       } else {
147         console.log(result);
148         return res.status(200).send(result);
149       }
150     }
151   );
152 };
```

Abbildung 42: Codeabschnitt von aoorivReq-Funktion-PUT

4.Delete / Löschen:

Die Firma kann auch eine Anzeige löschen. Es wird eine DELETE-Anfrage geschickt von Client an den Server und dann wird einen DELETE-Befehl an den Datenbankserver geschickt werden. (Siehe die Abbildung 43)

```
353 module.exports.delteOffer =(req,res) =>{
354   const id_job = req.query.id_job;
355   console.log(id_job);
356   db.query("DELETE FROM job WHERE id = ? ", [id_job], (err, result) => {
357     if (err) {
358       console.log(err);
359       return res.status(400).json({ message: "error" });
360     } else {
361       console.log(result);
362       return res.status(200).send(result);
363     }
364   });
365 };
```

Abbildung 43: Codeabschnitt von deleteOffer-Fumktion-DELETE

MySQL-Code

Mit dem Befehl `CREATE DATABASE bencom;` wurde die Datenbank „bencom“ erstellt.

Es wurde vier Tabellen erstellt, und in der „Abbildung 44“ wird gezeigt, wie die Tabelle „company“ erstellt wurde.

Die Namen jeder Spalte und ihr Datentyp, und ob sie NULL darf sein oder nicht, wurden bestimmt.

```
27
28 CREATE TABLE `company` (
29     `id` int(255) NOT NULL AUTO_INCREMENT,
30     `company_name` varchar(40) NOT NULL,
31     `origin` varchar(20) NOT NULL,
32     `found_date` varchar(40) NOT NULL,
33     `reg_nr` varchar(20) NOT NULL,
34     `street` varchar(20) NOT NULL,
35     `haus_nr` varchar(4) NOT NULL,
36     `post_code` varchar(10) NOT NULL,
37     `city` varchar(20) NOT NULL,
38     `tel_nr` varchar(20) NOT NULL,
39     `fax_nr` varchar(20) NOT NULL,
40     `email_company` varchar(40) NOT NULL,
41     `add_to_address` varchar(40),
42     `first_name_rep` varchar(20) NOT NULL,
43     `last_name_rep` varchar(20) NOT NULL,
44     `password` varchar(255) NOT NULL,
45     `email_rep` varchar(40) NOT NULL,
46     `tel_nr_rep` varchar(20) NOT NULL,
47     `mobile_nr_rep` varchar(20) NOT NULL,
48     `role` varchar(10) NOT NULL,
49     PRIMARY KEY (`id`)
50 );
51
```

Abbildung 44: Codeabschnitt von Erstellung einer Tabelle in MySQL

Da es eine Relation mit der Tabelle „company“ und „applied“ gibt, und zwar ein Primärschlüssel mit einem Fremdschlüssel, kann das Angebot nicht gelöscht werden, bis die Relation bei dem Fremdschlüssel in der Tabelle von **“RESTRICT“** zum **“CASCADE“** geändert wird, damit die Zeile in der untergeordneten Tabelle gelöscht oder verändert werden kann, wenn die Zeile in der übergeordneten Tabelle gelöscht oder verändert würde. (Siehe die Abbildung 45)

```
84 ALTER TABLE `job` ADD CONSTRAINT `job_fk0` FOREIGN KEY (`id_company`) REFERENCES `company` (`id`);
85
86 ALTER TABLE `applied` ADD CONSTRAINT `applied_fk0` FOREIGN KEY (`id_job`) REFERENCES `job` (`id`);
87
88 ALTER TABLE `applied` ADD CONSTRAINT `applied_fk1` FOREIGN KEY (`id_student`) REFERENCES `student` (`id`);
89
90 ALTER TABLE `applied` ADD CONSTRAINT `applied_fk2` FOREIGN KEY (`id_company`) REFERENCES `company` (`id`);
91
92
93 ALTER TABLE `applied` DROP FOREIGN KEY `applied_fk2`; ALTER TABLE `applied` ADD CONSTRAINT `applied_fk2` FOREIGN KEY (`id_company`) REFERENCES `company` (`id`) ON DELETE CASCADE;
94
95
96
```

Abbildung 45: Codeabschnitt von MySQL-Befehl ADD CONSTRAINT

4.3 Frontend-Seiten

Im Folgenden werden Screenshots von den Frontend-Seiten gezeigt, die in dieser Arbeit entwickelt wurden.

1.Landing-Page

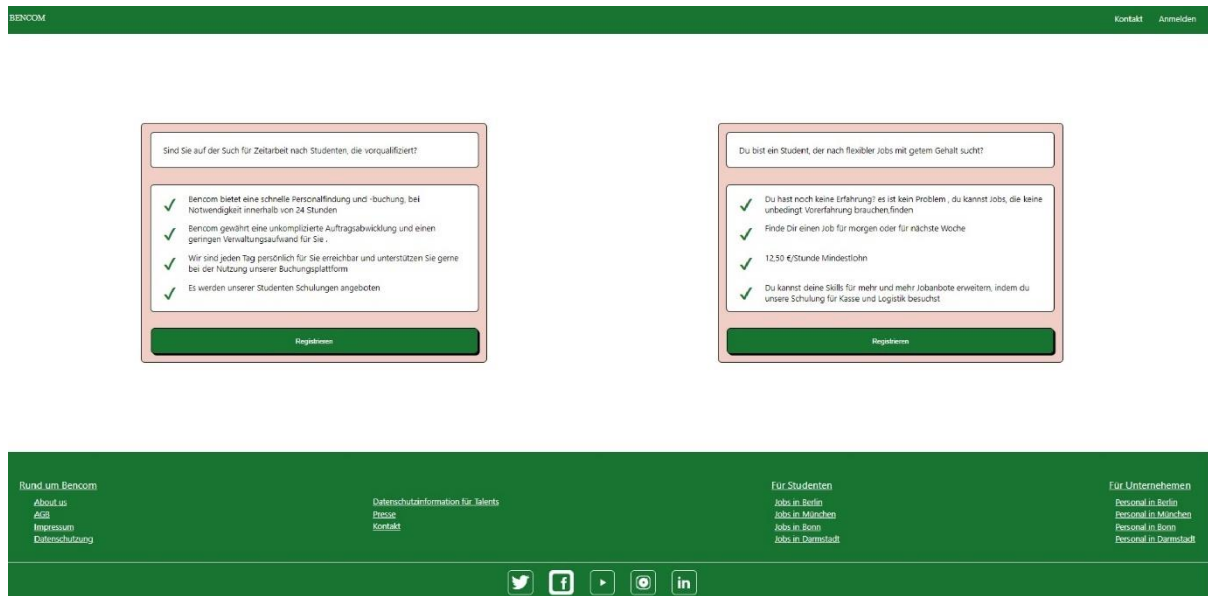


Abbildung 46: Screenshot von Landing-Page

2.Anmeldung-Page

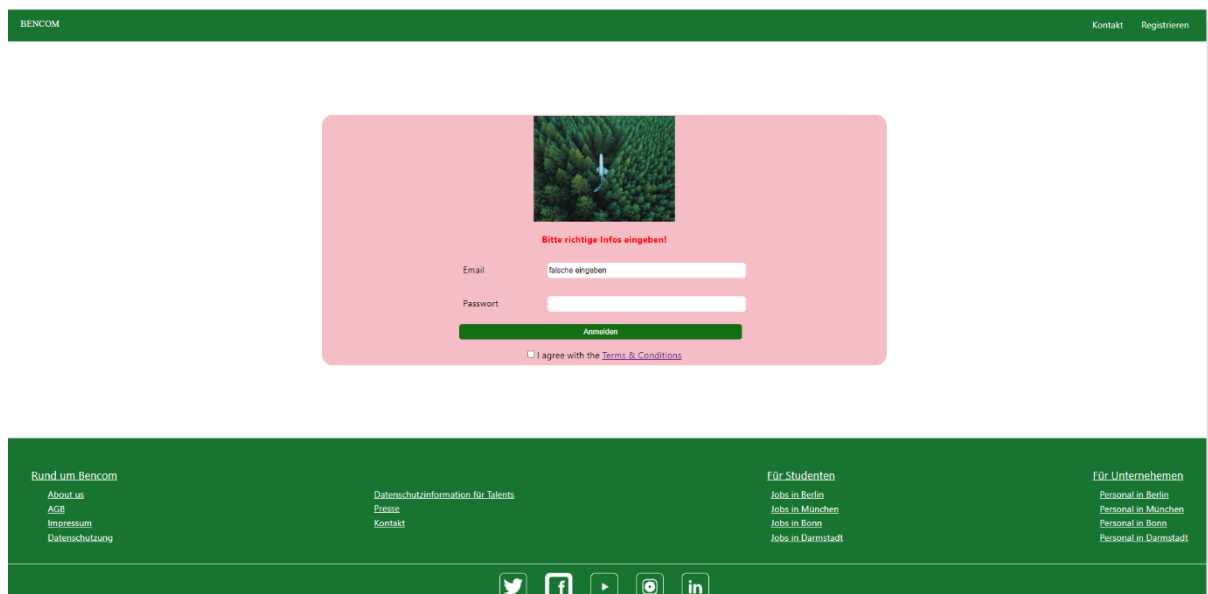


Abbildung 47: Screenshot von Anmeldung-Page

3.Registrierung-Page-Firma

Registrierung Als Unternehmen

Firma:
Gründungsdatum:

Adresse

Straße:
Tel.Nr.:
Haus.Nr.:
Email:
Stadt:
Fau.Nr.:
Postleitzahl:
Zusätzlich:

Vertreten durch

Vorname:
Email:
Passwort:
Tel.Nr.:
Name:
Re-Email:
Re-Passwort:
Handy.Nr.:

Registrieren
☐ I agree with the [Terms & Conditions](#)

Abbildung 48: Screenshot von Registrierung-Page-Firma

4. Registrierung-Page-Student

Registrierung Als Student

Vorname:
Geb.datum:
Nationalität:
Tel.Nr.:
Email:
Wohnort:
Passwort:
Haus.Nr.:
C/O:
Hochschule:
Name:
Geburtsort:
Geschlecht:
Handy.Nr.:
Re-Email:
Straße:
Re-Passwort:
Postleitzahl:
Zusatz zur Adresse:
Ausweis.Nr.:

Registrieren
☐ I agree with the [Terms & Conditions](#)

Rund um Bencom
[About Us](#)
[AGB](#)
[Impressum](#)
[Datenschutz](#)

Datenschutzhinweise für Talents
[Presse](#)
[Kontakt](#)

Für Studenten
[Jobs in Berlin](#)
[Jobs in München](#)
[Jobs in Bonn](#)
[Jobs in Darmstadt](#)

Für Unternehmen
[Personal in Berlin](#)
[Personal in München](#)
[Personal in Bonn](#)
[Personal in Darmstadt](#)

Abbildung 49: Screenshot von Registrierung-Page-Student

5. Homepage-Admin



Abbildung 50: Screenshot von Homepage-Admin

6. Alle-Angebote-Seite-Admin

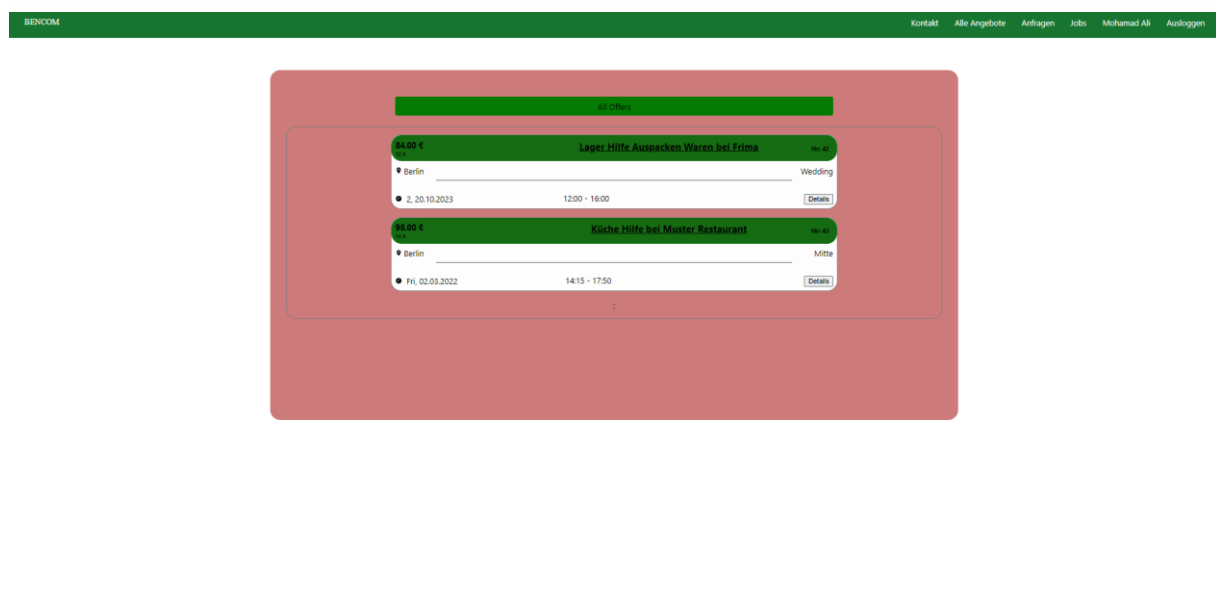


Abbildung 51: Screenshot von Alle-Angebote-Seite-Admin

7. Jobs-Page-Admin

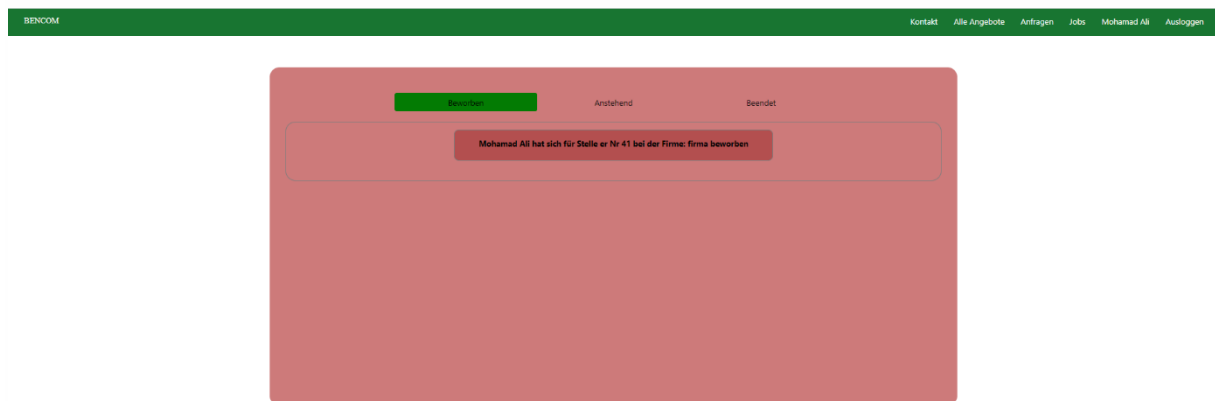


Abbildung 52: Screenshot von Jobs-Page-Admin

8. Homepage-Student



Abbildung 53: Screenshot von Homepage-Student

8. Meine-Jobs-Page-Student

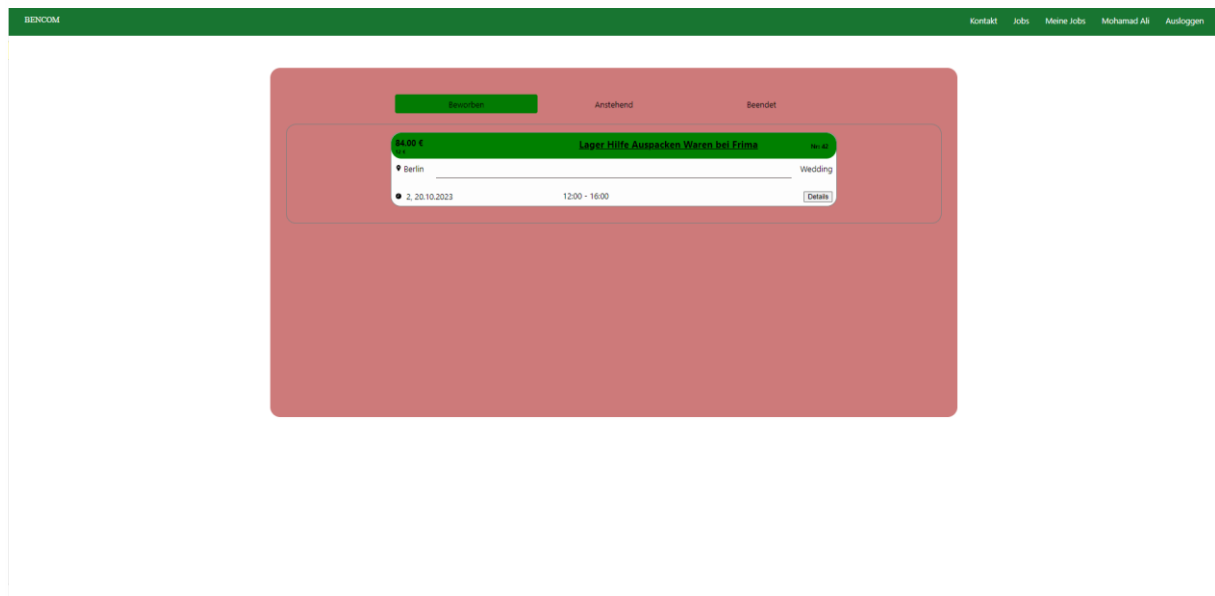


Abbildung 54: Screenshot von Meine-Jobs-Page-Student

8. Meine-Jobs-Page-Firma

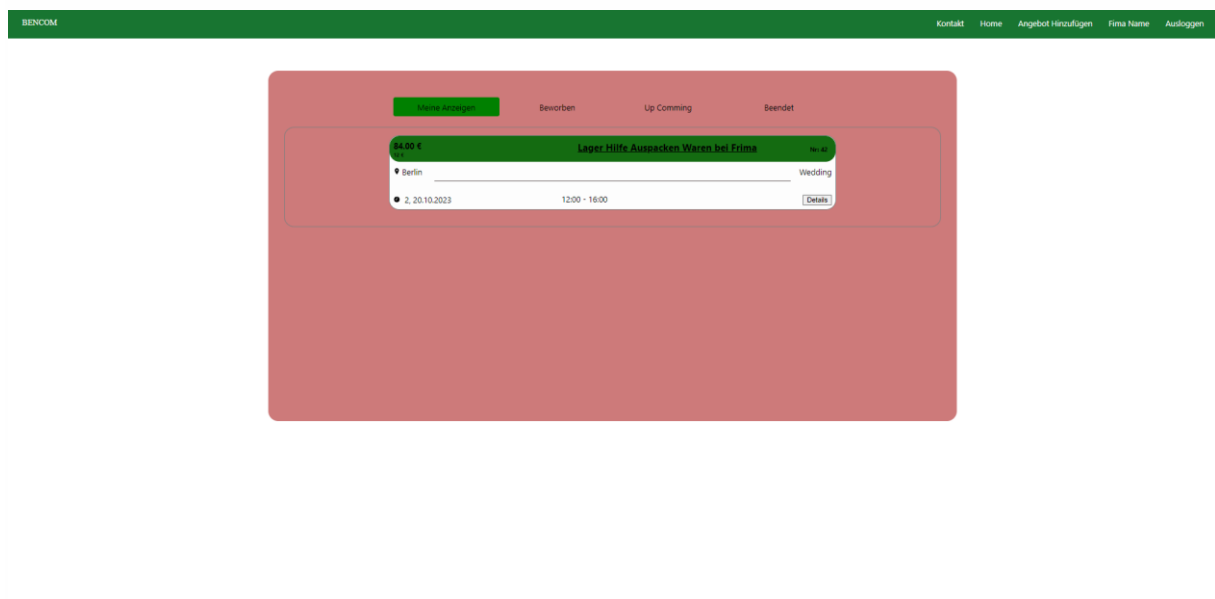


Abbildung 55: Screenshot von Meine-Jobs-Page-Firma

9. Angebote-Hinzufügen-Page-Firma

The screenshot shows a web form for adding a new offer. The form is titled 'Angebote-Hinzufügen' and is set against a pink background. It includes the following fields and sections:

- Header:** A green navigation bar with the logo 'BENCOM' on the left and links for 'Kontakt', 'Home', 'Angebot Hinzufügen', 'Firma Name', and 'Ausloggen' on the right.
- Title der Stelle:** A text input field.
- Wie viele Tage?:** A text input field.
- € Pro Stunde:** A text input field.
- Stadt:** A text input field.
- Stadtteil:** A text input field.
- Wie viele Personen?:** A text input field.
- Schicht Hinzufügen:** A section with the following fields:
 - Tag:** A text input field.
 - Wie viele Stunden?:** A text input field.
 - Zeit-von:** A text input field.
 - Zeit-bis:** A text input field.
 - Datum:** A text input field.
- Beschreibung:** A large text area for the offer description.
- Hinweise & Anforderungen:** A large text area for additional notes and requirements.
- Adresse:** A section with the following fields:
 - Straße:** A text input field.
 - HausNr:** A text input field.
 - Postleitzahl:** A text input field.
 - Zusätzlich:** A text input field.
- Submit Button:** A green button labeled 'Das Angebot Hinzufügen' at the bottom of the form.

Abbildung 56: Screenshot von Angebote-Hinzufügen-Page-Firma

10. Erfolgreiche-Registrierung-Meldung-Seite

The screenshot shows a confirmation page for successful registration. The page has a green navigation bar with the logo 'BENCOM' on the left and links for 'Kontakt' and 'Anmelden' on the right. The main content area is a light gray box with the following elements:

- Message:** A green text box with the message 'Glückwunsch! Du Hast Dich erfolgreich registriert!'.
- Submit Button:** A green button labeled 'Anmelden' at the bottom of the message box.

Abbildung 57: Screenshot von Erfolgreiche-Registrierung-Meldung-Seite

Kapital 6

Test

Nach der Implementierung der Bencom-Applikation werden in diesem Kapitel drei Anwendungsfälle manuell getestet. Und es wird in einer Tabelle die erfüllten und nicht erfüllten Kriterien gezeigt werden.

6.1 Anwendungsfälle-Manuell-Testen

Im Folgenden werden Drei Anwendungsfälle in der Applikation nach erfolgreicher Anmeldung getestet werden.

6.1.1 Angebot Hinzufügen

Nachdem die Firma sich erfolgreich angemeldet, wurde das Formular von der Angebotshinzufügen-Seite (Siehe die Abbildung 56) ausgefüllt.

Die Eingaben wurden zuerst validiert, bevor die POST-Anfrage an den Server geschickt werden.

Die Hinzufügung des Angebotes wurde erfolgreich durchgeführt, und der Benutzer zur Seite "addedSuccesPage" weitergeleitet, wo eine Meldung gezeigt wird, dass, das Angebot hinzugefügt wurde.

Wenn ein Blick auf die Tabelle "job" geworfen wird, sieht man die Daten in einer Zeile. (Siehe die Abbildung 58)

id	id_company	title_job	days_per	per_hour_money	city	neighborhood	persons_per	day_name	hours_per	time_from	time_until	date	description	note_and_requirements	street	house_no	post_code	add_to_address
1	1	Lager HIL bei Firma Name	3	15	Berlin	Berlin	1	Fri	8	08.00	18.30	23.09.2022	Nur mit Sicherheitskarte einstraten Bitte Maske L.	Pause 30 Minuten	arbeits-Muster-Straße 8	12345		notia

Abbildung 58: Screenshot von der Tabelle „job“ in der Datenbank

6.1.2 Bewerben

Nachdem ein Angebot von einer Firma hinzugefügt wurde, wird eine Komponente von der Anzeige in „homePageStudent“, „homePageFirma“, und „allOfferAdmin“ Seiten gezeigt. (Siehe die Abbildungen 51, 53, 55)

Wenn der Student auf den Link „Details“ in der Angebot-Komponente klickt, wird der Student zur „detailsOffer-Seite“ weitergeleitet, wo die ganzen Informationen über das Angebot gezeigt werden, und es ein Bewerben-Button gibt, um sich darum zu bewerben.

Die Bewerbung-Operation wurde erfolgreich durchgeführt, und die Daten wurden in der Tabelle „applied“ mit „applied-Status“ gesetzt. (Siehe die Abbildung 58)

Danach wurde eine Komponente Anfrage mit der Info der Bewerbung in die „homePageAdmin“ gelandet, damit der Admin die Entscheidung trifft, die Bewerbung entweder zu bestätigen oder abzulehnen. (Siehe die Abbildung 52)

Es wurde eine Komponente des beworbenen Jobs in Beworben-Spalte in der „Meine-Jobs-Student-Page“ gezeigt, damit der Student den Zustand der Bewerbung verfolgt. (Siehe die Abbildung 54)

Eine Komponente wurde auch des beworbenen Jobs in Beworben-Spalte in der „homePageFirma“ gezeigt, damit die Firma den Zustand der Bewerbung auch verfolgt. (Siehe die Abbildung 55)

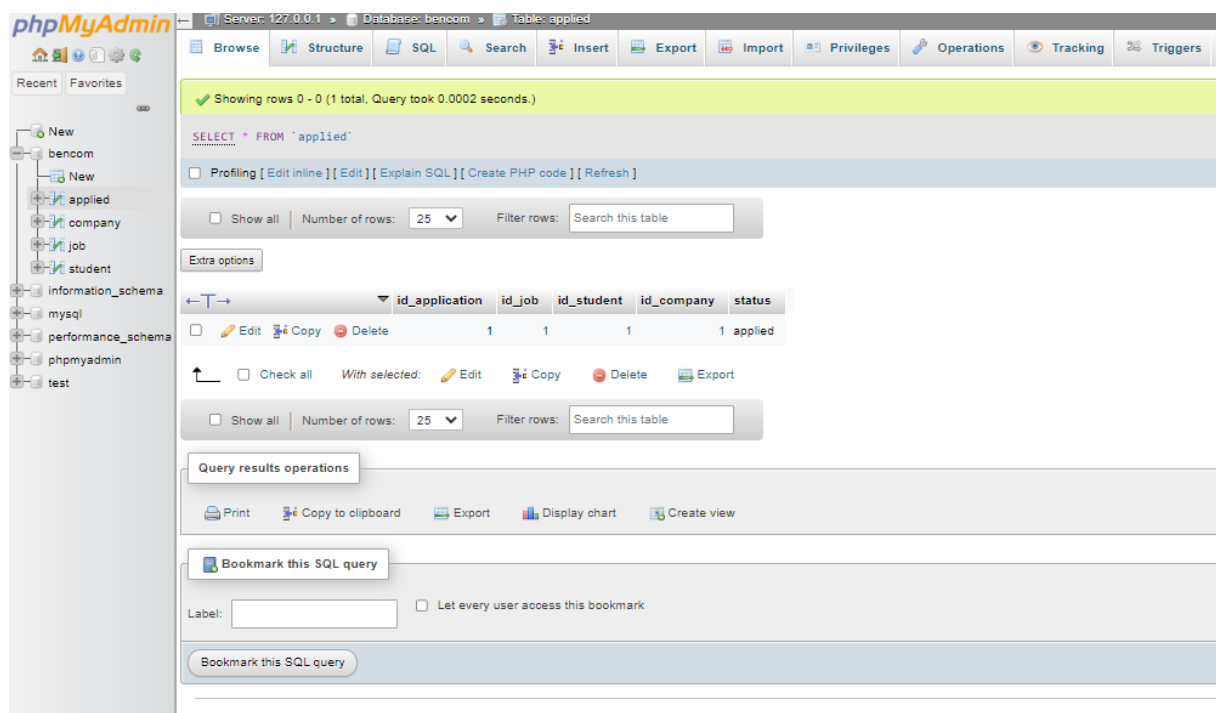


Abbildung 59: Screenshot von der „Tabelle applied“ Status: applied

6.1.3 Bewerbung-Bestätigen-Ablehnen

Wenn der Admin die Bewerbung ablehnen oder bestätigen möchte, klickt er auf dem angemessenen Button. (Siehe die Abbildung 50)

Die Beiden Operationen wurden getestet, und erfolgreich ausgeführt.

Die Bewerbungsdaten wurden erfolgreich in der Tabelle „applied“ erfolgreich abgespeichert. (Siehe die Abbildung 60)

Es wurde eine Komponente der anstehenden Bewerbung in Anstehend-Spalte in der „Meine-Jobs-Student-Page“ gezeigt, damit der Student den Zustand der Bewerbung verfolgt. (Siehe die Abbildung 54)

Es wurde auch eine Komponente der anstehenden Bewerbung in Anstehend-Spalte in der „homePageFirma“ gezeigt, damit die Firma den Zustand der Bewerbung auch verfolgt und den Einsatz beenden. (Siehe die Abbildung 55)

Genau so wurden die Komponenten in der „Jobs-Admin-Seite“ in der Spalte „Anstehend“ gesetzt. (Siehe die Abbildung 52)

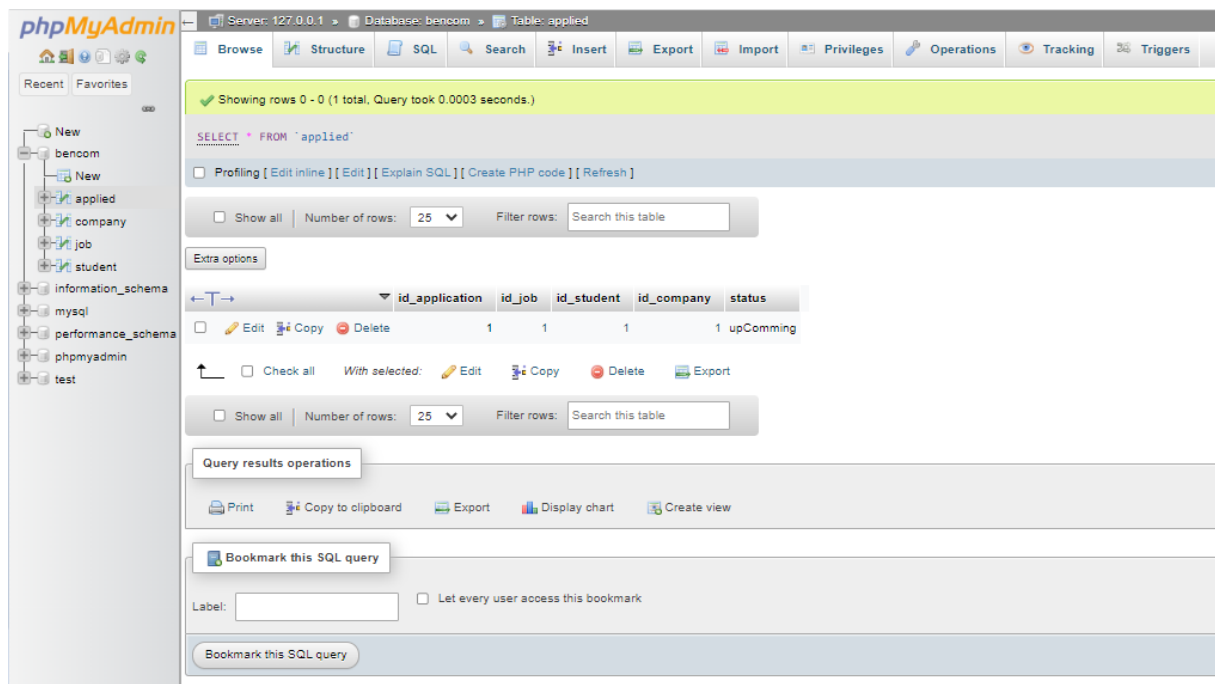


Abbildung 60: Screenshot von Tabelle „applied“ Status: Anstehend

6.2 Übersicht der erfüllten Kriterien

In der „Tabelle 3“ wird dargestellt, welche Muss bzw. Soll -Kriterien erfüllt worden sind.

Von 1 bis 20 Muss-Kriterien und 20 ist soll-Kriterien.

Nr.	Kriterium	erfüllt	teilweise erfüllt	nicht erfüllt	Kommentar
1	Die Applikation soll die Möglichkeit den Nutzern bieten, sich zu registrieren.	x			
2	Bei der Registrierung wird die Daten vom Client mit Axios (HTTP-POST-Request, der die Eingegeben Daten trägt) an den Server geschickt.	x			
3	Der Server bei der Registrierung empfängt die Anfrage vom Client bzw. wird die Daten an die Datenbank senden, und das eingegebene Password wird auch verschlüsselt im Backend, bevor es an die Datenbank geschickt werden.	x			
4	Es soll drei Rollen der Benutzer für die Applikation geben, und zwar "Admin, Student, und Firma".	x			
5	Bei der Anmeldung der Benutzer nach der Validierung die Eingegeben Daten werden geprüft, ob die eingegeben E-Mail und Passwort vorhanden und richtig sind.	x			
6	Bei der Anmeldung wird eine HTTP- GET-Anfrage vom Client an den Backend-Server gesendet, und dann	x			

	die Antwort wird an den Client zurückgesendet.				
7	Wenn die Daten richtig sind, wird der Benutzer in der Lage sein, die Applikation zu benutzen.	x			
8	Nach der erfolgreichen Anmeldung wird der Server ein Key mit JWT zur Autorisierung für den Benutzer erstellt.	x			
9	Die Daten der Benutzer: Name, ID, Rolle, und das JWT-Key im lokalen Speicher abgespeichert, um die im Frontend bzw. bei den Funktionen zu verwenden.	x			
10	Der Client als Firma soll die Möglichkeit haben, Angeboten zu posten.	x			
11	Die hinzugefügten von den Unternehmen Angebote sollen im Frontend angezeigt werden.	x			
12	Der Client als Student soll sich um die Angebote bewerben können.	x			
13	Nach der Bewerbung bekommt der Admin die Bewerbungen, und soll der Admin die Möglichkeit haben, zu bestätigen bzw. zu ablehnen.	x			
14	Nach der Bestätigung wird der Status der Bewerbung zum "Anstehend" in der Datenbank geändert.	x			
15	Der Zustand „beworben, anstehend, fertig“ der	x			

	Bewerbung wird im Frontend gezeigt.				
16	Der Client als Firma soll in der Lage sein, den Status der Bewerbung nach einem erfolgreichen Einsatz zum „Fertig“ ändern.	x			
17	Bei jeder Funktion soll das Key des Nutzers geprüft, um den Zugang zu den Privilegien zu haben.	x			
18	Die Nutzer sollen auch die Möglichkeit haben, auszuloggen.	x			
19	Es wird Routing für die Seiten der Applikation verwendet, also das Zuordnen einer URL in der Anwendung zu einer bestimmten Funktion.	x			
20	Beim Ausloggen sollen die Daten, die Im lokalen Speicher des Browsers sind, gelöscht werden.	x			
21	Die Firma soll in der Lage sein, die Angebote zu löschen	x			

Tabelle 3: Muss-Soll Kriterien erfüllen

Kapital 7

Ausblick

Das gesamte Projekt wird in diesem Kapitel zusammengefasst. Es wird festgelegt, wie das Projekt entstanden ist und geschafft wurde.

Darüber hinaus werden die Optimierungen erläutert, die zukünftig sowohl an dem Frontend als auch an dem Backend vorgenommen werden könnten.

7.1 Zusammenfassung

Es wurde in dieser Arbeit eine Single-Web-Applikation mit React.JS, Node.JS, Express.JS und MySQL für die Studentenjobvermittlung entworfen und entwickelt, die einen flexiblen Buchungsprozess den Studenten und Arbeitgebern bietet.

Zuerst wurden die Ziele der Bencom-Applikation und der Lauf der Benutzung bestimmt. Dann wurde ein gezeichneter Prototyp mit WireframeSketcher-App und dann ein grafischer Prototyp mit Adobe XD erstellt, da sie bei der Entwicklung des Frontends hilfreich sind.

Eine React.JS-Applikation wurde auf dem Rechner in einem Ordner erstellt, und wurden die geforderten Bibliotheken und Tools installiert.

Als nächster Schritt wurde das Frontend implementiert, und zwar wurden die Seiten mit HTML5, JSX, und JavaScript entwickelt. Die Benutzeroberfläche ist Responsive für Desktop, Tablet, und Handys entwickelt worden.

Danach wurden die Daten mit deren Typen bestimmt, die für die App gebracht werden und dann wurde die MySQL-Datenbank mit deren Tabellen erstellt.

Nachdem die Datenbank erstellt worden war, wurde der Server im Backend mit Node.JS und Express.JS aufgebaut und mit der Datenbank und Frontend verbunden.

Das Frontend kommuniziert mit dem Server, der auch mit dem Client und der Datenbank kommuniziert.

Es wurden Abbildungen für die Prototypen, Screenshots von den Seiten, und Diagramme (ER und Aktivität) in dieser Arbeit gezeigt.

Nach der erfolgreichen Implementierung wurde die Applikation getestet, und das Ergebnis war wie erwartet.

Das Endergebnis ist, dass die allen Müssen-Sollen Kriterien erfüllt worden sind.

7.2 Optimierung

In dieser Arbeit wurden die nur die Hauptsachen implementiert. Die Applikation ist noch nicht in einem optimalen Zustand, und könnte noch verbessert und weiterentwickelt werden.

Im Frontend könnten die folgenden Features entwickelt werden:

Es könnte ein schönes eigenartiges Logo entwerfen und erstellt.

Die Validierung der Eingegeben Daten bei der Registrierung könnte verbessert werden, indem es zum Beispiel keine Ziffern oder besondere Zeichen bei Namen akzeptiert werden.

Für mehr Sicherheit würde ein komplexes Passwort gefördert, das mindestens ein Sonderzeichen, eine Ziffer, und ein Großbuchstabe hätte.

Die eingegebenen Namen und E-Mail könnten bei der Registrierung zum Großbuchstabe Am Anfang umformuliert werden, bevor sie an den Server geschickt würden, und in der Datenbank abgespeichert würden.

Bei Postleitzahl könnten nur fünf Ziffern eingegeben werden. Es sollte ein E-Mail-Server erstellt werden, damit der Benutzer eine E-Mail nach der erfolgreichen Registrierung erhält, um die E-Mail zu verifizieren, und auch beim Passwort zurücksetzen zu verwenden.

Es könnte auch Benachrichtigung-Feature bei neuen Angeboten, Bewerbungen, Bestätigung, und Einsatz beenden entwickelt werden, damit die Benutzer die Neuigkeiten immer verfolgen könnten. Es könnte sein, dass, der Student in verschiedene Städte besuchen, und es wäre gut, wenn die Studenten die Möglichkeit haben, nach Angeboten laut der Stadt zu suchen.

Jeder Benutzer könnte eigene Konto-Seite haben, wo die Benutzer ihre persönlichen Daten aktualisieren könnten.

Die Bewerber könnten ihre Bewerbungen absagen.

Im Backend könnte noch Refresh-Token verwendet werden, um ein neues Token vom Authorization-Server nach dessen Ablauf anzufordern.

Quellenverzeichnis

- [B1] User Guide – WireframeSketcher: Abgerufen am 08.09.2022 von: <https://wireframesketcher.com/help/help.html>
- [B2] Hauptseite von Adobe XD: Abgerufen am 08.09.2022 von: <https://www.adobe.com/products/xd.html>
- [B3] Screenshot von der Plattform von Adobe XD: Abgerufen am 08.09.2022 von: <https://stacksol.blogspot.com/2018/09/adobe-xd-free-ui-and-ux-design-tool-you.html>
- [B4] VS-Code – Dokumentation-Seite: Abgerufen am 08.09.2022 von: <https://code.visualstudio.com/docs>
- [B5] JavaScript - Dokumentation Seite: Abgerufen am 08.09.2022 von: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
- [B6] React.JS-Dokumentation-Seite: Abgerufen am 08.09.2022 von: <https://reactjs.org/docs/getting-started.html>
- [B7] SASS - Dokumentation -Seite: Abgerufen am 08.09.2022 von: <https://sass-lang.com/documentation/>
- [B8] Wie SASS funktioniert: Abgerufen am 08.09.2022 von: <https://kulturbanause.de/blog/einstieg-in-sass-funktionsweise-und-ueberblick/>
- [B9] Node.JS - Dokumentation Seite: Abgerufen am 08.09.2022 von: https://docs.airplane.dev/tasks/node?qclid=Cj0KCQjwjlKYBhC6ARIsAGEds-JVqdB1y7oVvLI4mGHo8i0uyQkcOg4zTkLn6yVDQnkFmVbpfYeF5sEaAngBEALw_wcB
- [B10] Die Architektur von Node.JS: Abgerufen am 08.09.2022 von: <https://litslink.com/blog/node-js-architecture-from-a-to-z>
- [B11] Hauptseite von Express.JS: Abgerufen am 08.09.2022 von: <https://expressjs.com/>
- [B12] Middleware-Aufrufen in Express.JS: Abgerufen am 08.09.2022 von: <https://devopedia.org/express-js>
- [B13] Was ist MySQL? Abgerufen am 08.09.2022 von: <https://www.bigdata-insider.de/was-ist-mysql-a-614184/>

- [B14] Ein Beispiel von relationaler Beziehung: Abgerufen am 08.09.2022 von:
<https://maxcluster.de/blog/datenbankverwaltung-webseiten-mysql>
- [C1] Axios-Dokumentation-Seite: Abgerufen am 08.09.2022 von:
<https://axios-http.com/docs/intro>
- [E1] NPM-Wikipedia: Abgerufen am 08.09.2022 von:
<https://www.bezkoder.com/react-node-express-mysql/>
- [E2] NPM-Wikipedia: Abgerufen am 08.09.2022 von:
[https://de.wikipedia.org/wiki/Npm \(Software\)](https://de.wikipedia.org/wiki/Npm_(Software))
- [E3] JWT-Überblick: Abgerufen am 08.09.2022 von: <https://www.security-insider.de/was-ist-json-web-token-jwt-a-1094265/>

Abbildungsverzeichnis

ABBILDUNG 1: DIE ANSICHTEN IN WIREFRAMESKETCHER [B1]	4
ABBILDUNG 2: SCREENSHOT VON DER PLATTFORM VON ADOBE XD [B3]	5
ABBILDUNG 3: WIE DIE KOMPONENTE IN REACT.JS GEBAUT WIRD.	8
ABBILDUNG 4: WIE SASS FUNKTIONIERT. [B8]	9
ABBILDUNG 5: DIE ARCHITEKTUR VON NODE.JS [B10]	10
ABBILDUNG 6: MIDDLEWARE-AUFRUFEN IN EXPRESS.JS [B10]	11
ABBILDUNG 7: EIN BEISPIEL VON RELATIONALER BEZIEHUNG. [B13]	13
ABBILDUNG 8: USE CASE DIAGRAMM VON "BENCOM" APPLIKATION.....	14
ABBILDUNG 9: LANDING-PAGE PROTOTYP	18
ABBILDUNG 10: REGISTRIERUNG-FIRMA-PAGE PROTOTYP.....	19
ABBILDUNG 11: ANMELDUNG-PAGE-PROTOTYP.....	20
ABBILDUNG 12: HOME-PAGE-ADMIN-PROTOTYP DES BENUTZERS.....	21
ABBILDUNG 13: ALLE ANGEBOTE-PAGE-ADMIN PROTOTYP.....	22
ABBILDUNG 14: JOBS-PAGE-ADMIN PROTOTYPE.....	23
ABBILDUNG 15: HOME-PAGE-STUDENT PROTOTYP.....	24
ABBILDUNG 16: OFFER-DETAILS-PAGE-STUDENT PROTOTYP	25
ABBILDUNG 17: MEINE JOBS-PAGE-STUDENT PROTOTYP.....	26
ABBILDUNG 18: HOMEPAGE-FIRME- PROTOTYP.....	27
ABBILDUNG 19: ANGEBOT-HINZUFÜGEN-FIRME- PROTOTYPE.....	28
ABBILDUNG 20: ER DIAGRAMM FÜR DIE DATENBANK.....	29
ABBILDUNG 21: KLASSENDIAGRAMM DER APPLIKATION	30
ABBILDUNG 22: AKTIVITÄTSDIAGRAMM FÜR ADMIN.....	31
ABBILDUNG 23: AKTIVITÄTSDIAGRAMM FÜR FIRMA	32
ABBILDUNG 24: AKTIVITÄTSDIAGRAMM FÜR ADMIN.....	33
ABBILDUNG 25: STRUKTUR DER APPLIKATION [E1].....	34
ABBILDUNG 26: DER VERZEICHNISSTRUKTUR DER FRONTENDSEITE.....	35
ABBILDUNG 27: SCREENSHOT DES CODEAUSSCHNITTES VON ROUTEN UND IMPORTIEREN	36
ABBILDUNG 28: SCREENSHOT DES CODEAUSSCHNITTES VON HTML5.....	37
ABBILDUNG 29: SCREENSHOT DES CODEAUSSCHNITTES „USESTATE-HOOK“	38
ABBILDUNG 30: SCREENSHOT DES CODEAUSSCHNITTES VON „USEEFFECT-HOOK“	38

ABBILDUNG 31: SCREENSHOT DES CODEAUSSCHNITTES VON SASS.....	39
ABBILDUNG 32: DER VERZEICHNISSTRUKTUR DER BACKENDSEITE.....	40
ABBILDUNG 33: CODEABSCHNITT VON DER SERVER.JS DATEI IM BACKEND	41
ABBILDUNG 34: CODEABSCHNITT VON ROUTEN DER OFFER.JS DATEI	42
ABBILDUNG 35: SCREENSHOT VOM LOKALEN SPEICHERN DES BROWSERS	42
ABBILDUNG 36: DER VERZEICHNISSTRUKTUR DER BACKENDSEITE.....	43
ABBILDUNG 37: SCREENSHOT VOM TOKEN ENCODED–DECODED	44
ABBILDUNG 38: CODEABSCHNITT VON ROLLENAUTORISIERUNG IN COMPANY.JS DATEI	44
ABBILDUNG 39: CODEABSCHNITT VON DER ERSTELLUNG DER VERBINDUNG MIT DER DATENBANK	45
ABBILDUNG 40: CODEABSCHNITT VON APPLAYOFFER–FUNKTION–POST	46
ABBILDUNG 41: CODEABSCHNITT VON GETALLOFFERS–FUNKTION–GET	46
ABBILDUNG 42: CODEABSCHNITT VON AORIVREQ–FUNKTION–PUT	47
ABBILDUNG 43: CODEABSCHNITT VON DELETEOFFER–FUNKTION–DELETE.....	47
ABBILDUNG 44: CODEABSCHNITT VON ERSTELLUNG EINER TABELLE IN MySQL.....	48
ABBILDUNG 45: CODEABSCHNITT VON MySQL–BEFEHL ADD CONSTRAINT	48
ABBILDUNG 46: SCREENSHOT VON LANDING–PAGE	49
ABBILDUNG 47: SCREENSHOT VON ANMELDUNG–PAGE	49
ABBILDUNG 48: SCREENSHOT VON REGISTRIERUNG–PAGE–FIRMA.....	50
ABBILDUNG 49: SCREENSHOT VON REGISTRIERUNG–PAGE–STUDENT.....	50
ABBILDUNG 50: SCREENSHOT VON HOMEPAGE–ADMIN.....	51
ABBILDUNG 51: SCREENSHOT VON ALLE–ANGEBOTE–SEITE–ADMIN	51
ABBILDUNG 52: SCREENSHOT VON JOBS–PAGE–ADMIN	52
ABBILDUNG 53: SCREENSHOT VON HOMEPAGE–STUDENT	52
ABBILDUNG 54: SCREENSHOT VON MEINE–JOBS–PAGE–STUDENT.....	53
ABBILDUNG 55: SCREENSHOT VON MEINE–JOBS–PAGE–FIRMA.....	53
ABBILDUNG 56: SCREENSHOT VON ANGEBOTE–HINZUFÜGEN–PAGE–FIRMA.....	54
ABBILDUNG 57: SCREENSHOT VON ERFOLGREICHE–REGISTRIERUNG–MELDUNG–SEITE.....	54
ABBILDUNG 58: SCREENSHOT VON DER TABELLE „JOB“ IN DER DATENBANK.....	55
ABBILDUNG 59: SCREENSHOT VON DER „TABELLE APPLIED“ STATUS: APPLIED	56
ABBILDUNG 60: SCREENSHOT VON TABELLE “APPLIED” STATUS: ANSTEHEND	57

Tabellenverzeichnis

TABELLE 1: DIE CRUD-OPERATIONEN IN EXPRESS.JS DIE „CORS-MIDDLEWARE“ GIT DEM SERVER DIE MÖGLICHKEIT, DIE ANFRAGEN VON VERSCHIEDNE DOMAINE ZU ERHALTEN.....	40
TABELLE 2: DIE CRUD-OPERATIONEN IN MYSQL.....	45
TABELLE 3: MUSS-SOLL KRITERIEN ERFÜLLEN	60