

Bachelorarbeit

Entwurf und Implementierung einer Webanwendung für die Jobsuche mit Unterstützung durch Testautomatisierung

zur Erlangung des akademischen Grades

Bachelor of Science (B.Sc.)

vorgelegt dem

Fachbereich Mathematik, Naturwissenschaften und Informatik

der Technischen Hochschule Mittelhessen

Yvan Richnel Tchiengue

im September 2023

Referent: Prof. Dr. Axel Schumann

Korreferent: Herr Manuel Groh

Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Arbeit selbstständig und unter ausschließlicher Verwendung der angegebenen Literatur und Hilfsmittel erstellt zu haben.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

>Ort, Datum< >Unterschrift<

Inhalt

Inhalt	5
Abbildungen	6
Tabellen	7
Listings	8
Abkürzungen und Formelzeichen	9
1 Einleitung	10
1.1 Motivation und Problemstellung	10
1.2 Zielsetzung	11
1.3 Aufbau der Arbeit	12
2 Stand der Wissenschaft und Technik	13
3 Grundlagen	14
3.1 Entwicklungswerkzeuge	14
3.1.1 git	14
3.1.2 GitHub	14
3.1.3 WebStorm	15
3.1.4 NodeJS	16
3.1.5 NPM	17
3.2 verwendete Technologien zur Webentwicklung	17
3.2.1 Angular	17
3.2.2 Html	18
3.2.3 TypeScript	19
3.2.4 JavaScript	19
3.2.5 CSS	20
3.3 Konzepte der Webentwicklung	20
3.3.1 HTTP	21
3.3.2 JSON	21
3.3.3 API	22
3.4 Webanwendung	23
3.4.1 wie funktioniert eine Webanwendung?	23
3.5 Mobilanwendung	24
3.5.1 Funktionsweise einer mobilen Anwendung	25
3.6 Unterschied zwischen Web- und Mobilanwendung	26
4 Methodik	28
4.1 Anforderungsanalyse	28
4.1.1 Analyse funktionaler Anforderungen	28

4.1.2	Analyse nicht-funktionaler Anforderungen	29
4.1.2.1	Sicherheit	29
4.1.2.2	Leistung	30
4.1.2.3	Benutzerfreundlichkeit	30
4.2	Schritte zur Entwicklung einer Webanwendung	31
4.2.1	Analyse der Bedürfnisse und Ziele	31
4.2.2	Design der Anwendung	31
4.2.3	Back-End-Entwicklung	32
4.2.4	Front-End-Entwicklung	32
4.2.5	Durchführung der Tests	33
4.2.6	Deployment der Webanwendung	33
4.3	Kommunikation zwischen Front-End und Back-End	34
4.4	Testautomatisierung	35
5	Bearbeitungsphase	36
5.1	Verwendete Bibliotheken	36
5.1.1	jwt	36
5.1.2	Express	36
5.1.3	Bcrypt	37
5.1.4	Multer	37
5.2	Implementierung der wichtigsten Funktionen	38
5.2.1	Konfiguration des Express-Servers und Erstellen der Datenbank	38
5.2.2	Detaillierte Vorgehensweise für die Erstellung eines Benutzerkontos auf der Plattform	40
5.2.3	Implementation der Anmeldung bei einem bestehenden Benutzerkonto	44
5.2.4	Dashboard	47
5.2.5	Jobsuche	48
5.2.6	Speicherung von Dateien	50
5.2.7	Ausfüllen des Formulars und Veröffentlichung einer Stellensuche	53
5.2.8	Einreichen der Bewerbung auf ein Stellenangebot und deren Überprüfung durch den Arbeitgeber	54
6	Ergebnissbewertung	58
7	Zusammenfassung und Ausblick	60
Literatur		61

Abbildungen

3.1	Git Übersicht (aus [8])	14
3.2	GitGub Übersicht (aus [7])	15
3.3	WebStorm Übersicht (aus [18])	16
3.4	Nodejs Übersicht (aus [51])	17
3.5	NPM Übersicht (aus [9])	17
3.6	Angular Übersicht (aus [20])	18
3.7	Html Übersicht (aus [11])	19
3.8	TypeScript Übersicht (aus [37])	19
3.9	JavaScript Übersicht (aus [13])	20
3.10	Schematische Darstellung des Kommunikationsprozesses gemäß HTTP-Protokoll (aus [12])	21
3.11	JSON Übersicht (aus [3])	22
3.12	HOW API WORKS (aus [1])	22
3.13	The Flow of the Web Application (aus [15])	24
3.14	mobile application Übersicht (aus [26])	25
4.1	Web Application security against Cyber Attack (aus [34])	30
4.2	User Experience, User Interface (aus [38])	32
4.3	Back-End vs. Front-End Development (aus [33])	33
4.4	Communication between Frontend and Backend (aus [31])	35
5.1	Json Web Token Übersicht (aus [28])	36
5.2	Express Übersicht (aus [4])	37
5.3	Hashing Passwords with Bcrypt (aus [6])	37
5.4	multer Übersicht (aus [21])	38
5.5	Formular zur Kontoerstellung	42
5.6	Anmeldung bei einem bestehenden Benutzerkonto	45
5.7	Boardingboard für den Arbeitnehmer	47
5.8	Boardingboard für den Arbeitgeber	48
5.9	Anzeige von verfügbaren Stellenangeboten	48
5.10	Anzeige von verfügbaren Stellenangeboten	49
5.11	Implementierung der Sicherung wichtiger Dokumente in der Webanwendung	50
5.12	Ausfüllen des Formulars und Veröffentlichung einer Stellensuche	53
5.13	Einreichen der Bewerbung	55
5.14	Verwaltung von Bewerbungen	56

Tabellen

3.1 Unterschied zwischen Web- und Mobilanwendung.	27
4.1 Nicht-funktionaler Anforderungen.	31

Listings

5.1	Konfiguration des Express-Servers und Erstellen der Datenbank	38
5.2	Front-End Code des Formulars zur Erstellung eines Benutzerkontos	41
5.3	API für die Methode zur Erstellung eines Kontos	42
5.4	implementation of an API route for creating user accounts in our application	43
5.5	Code für die Anmeldung bei einem Benutzerkonto	44
5.6	API Route für die Verbindung mit einem Benutzerkonto.	45
5.7	Anzeige aller in der Datenbank verfügbaren Jobs	49
5.8	Anzeige aller in der Datenbank verfügbaren Jobs	50
5.9	Verwaltung des Uploads von Identitätsdokumenten	51
5.10	Umsetzung des Systems zur Veröffentlichung von Stellenangeboten	53
5.11	Methode zum Einreichen der Bewerbung	55
5.12	Abruf von mit Stellenangeboten verknüpften Bewerbungen	56
5.13	Herunterladen des Personalausweises eines Bewerbers	57

Abkürzungen und Formelzeichen

NPM	Node Package Manager
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
API	Application Programming Interface
CORS	Cross-Origin Resource Sharing
JWT	JSON Web Token
AWS	Amazon Web Services
URL	Uniform Ressource Locator
GUI	Graphical User Interface
IDE	Integrated Development Environment
OWASP	Open Web Application Security Project
XML	Extensible Markup Language
W3C	World Wide Web Consortium
UI/UX	User Interface / User Experience
USENIX	The Advanced Computing Systems Association

1 Einleitung

In einer sich ständig verändernden Welt ist die Arbeitssuche für viele Menschen auf der ganzen Welt zu einem wichtigen Anliegen geworden. Eine Stelle zu finden, die den eigenen Fähigkeiten, Interessen und beruflichen Zielen entspricht, ist eine Herausforderung, der sich viele Menschen stellen müssen. In diesem Zusammenhang spielen Jobsuchplattformen eine wesentliche Rolle, indem sie die Kontaktaufnahme zwischen Arbeitssuchenden und Arbeitgebern erleichtern.

Diese Bachelorarbeit konzentriert sich auf die Entwicklung einer Jobsuchplattform, ein leistungsstarkes Instrument, das den Prozess der Jobsuche für Einzelpersonen, die nach beruflichen Möglichkeiten suchen, erleichtern soll.

Kamerun ist ein aufstrebender Land mit einer steigenden Nachfrage nach Arbeitsplätzen. Mit einer jungen und dynamischen Bevölkerung ist es von entscheidender Bedeutung, innovative Lösungen anzubieten, um kamerunische Talente mit den verfügbaren Arbeitsmöglichkeiten zu verbinden. Diese Bachelorarbeit befasst sich speziell mit den Herausforderungen und Chancen, die mit der Schaffung einer Plattform für die Jobsuche in Kamerun verbunden sind, wobei die einzigartigen Merkmale des Landes berücksichtigt und geeignete Lösungen vorgeschlagen werden.

1.1 Motivation und Problemstellung

In Kamerun beobachten wir eine signifikante demografische Konzentration in zwei großen Städten, Yaoundé und Douala, die allein Millionen von Einwohnern beherbergen. Der schrittweise Ausbau des Internetzugangs in diesen Regionen hat die Qualität des täglichen Lebens zweifellos verbessert. Neben Yaoundé und Douala gibt es jedoch noch viele andere Städte, die über das riesige kamerunische Staatsgebiet von 475.445 km² verteilt sind und in denen der Rest der Bevölkerung lebt.

Es ist wichtig anzumerken, dass Einzelpersonen, die an Universitäten im Norden des Landes, wie der Universität Ngaoundéré, studiert und ihren Abschluss gemacht haben, Schwierigkeiten haben, einen Arbeitsplatz zu finden, obwohl es auch in anderen Teilen des Landes freie Stellen gibt. Dies ist vor allem auf den Mangel an digitalen Plattformen zurückzuführen, die Arbeitgeber und Arbeitssuchende miteinander verbinden. So kann es sein, dass es in Douala eine offene Stelle gibt, aber eine Person, die beispielsweise in Garoua wohnt, würde nicht von dieser Möglichkeit erfahren, da es sowohl umständlich als auch finanziell schwierig wäre, persönlich von Garoua nach Douala zu reisen, um sich zu bewerben.

Die Einrichtung einer zuverlässigen Online-Plattform für die Jobsuche würde es Einzelpersonen, unabhängig davon, ob sie im Norden oder im Süden des Landes leben, ermöglichen, sich auf Stellenangebote in jedem Ort zu bewerben. Diese Lö-

sung würde auch das Problem lösen, dass Menschen, die in derselben Stadt wohnen, in der es eine offene Stelle gibt, nichts davon wissen, da diese Möglichkeiten oft durch Mundpropaganda oder durch Aushänge an öffentlichen Orten bekannt gemacht werden.

Ich kenne Menschen in Kamerun, die auf der Suche nach einer Arbeitsstelle wiederholt von Tür zu Tür gehen mussten, in der Hoffnung, ein Unternehmen zu finden, das Arbeitskräfte sucht, aber ohne Erfolg. Gleichzeitig war ein kamerunischer Unternehmer gezwungen, sein Unternehmen für einige Zeit zu schließen, weil es an qualifizierten und geeigneten Arbeitskräften mangelte.

Die Einrichtung einer robusten digitalen Plattform, die eine effektive Verbindung zwischen Arbeitgebern und Arbeitssuchenden ermöglicht, ist daher von entscheidender Bedeutung, um diese Probleme zu beheben und ein besseres Gleichgewicht zwischen Angebot und Nachfrage nach qualifizierten Arbeitsplätzen im ganzen Land zu fördern.

1.2 Zielsetzung

Im Rahmen der Thesis über die Schaffung einer Plattform für die Stellensuche ist das Hauptproblem, das es zu lösen gilt, das bestehende Ungleichgewicht auf dem kamerunischen Arbeitsmarkt. Dieses Ungleichgewicht äußert sich in einer hohen Anzahl qualifizierter Arbeitssuchender, die nach beruflichen Möglichkeiten suchen, während es für Arbeitgeber schwierig ist, die richtigen Talente für ihre spezifischen Bedürfnisse zu finden. Dieses Problem führt zu einer Lücke zwischen Angebot und Nachfrage auf dem kamerunischen Arbeitsmarkt, was hohe Arbeitslosigkeit und eine unzureichende Nutzung von Qualifikationen zur Folge hat.

Die Arbeitssuche ist für Bewerber oft ein komplexer und zeitaufwändiger Prozess mit vielen Herausforderungen wie der Suche nach geeigneten Möglichkeiten, der Übereinstimmung mit den Anforderungen der Arbeitgeber, der effektiven Präsentation von Kompetenzen und dem Zugang zu relevanten Informationen über verfügbare Beschäftigungsmöglichkeiten. Darüber hinaus stehen Arbeitgeber vor der Herausforderung, ein breites Spektrum an qualifizierten Bewerbern zu erreichen, deren Eignung für die offenen Stellen zu bewerten und den Einstellungsprozess effektiv zu steuern.

Mit der Schaffung einer auf Kamerun zugeschnittenen Plattform für die Stellensuche soll dieses Problem gelöst werden, indem eine effiziente Kontaktaufnahme zwischen Arbeitssuchenden und Arbeitgebern erleichtert wird. Die Plattform wird einen zentralen Bereich bereitstellen, in dem Bewerber ihre Fähigkeiten präsentieren, nach relevanten Möglichkeiten suchen und sich auf Stellenangebote bewerben können, während Arbeitgeber gezielt Stellenangebote veröffentlichen, nach qualifizierten Bewerbern suchen und den Einstellungsprozess rationeller verwalten können.

1.3 Aufbau der Arbeit

Der Prozess der Implementierung einer Webanwendung für die Stellensuche ist dicht und multidisziplinär. Um dem Leser ein klares und strukturiertes Verständnis dieses Prozesses zu vermitteln, ist diese Arbeit in mehrere Hauptabschnitte unterteilt.

Zunächst wird **der Stand von Wissenschaft und Technik** ermittelt. Hier wird der aktuelle Kontext gesetzt, wobei die Entwicklung und die Herausforderungen im Bereich der Webanwendungen beleuchtet werden.

Anschließend wird, um ein solides Fundament zu schaffen, in **die Grundlagen** eingetaucht. Dieser Abschnitt ist keineswegs als monolithisch zu betrachten, sondern untersucht viele verschiedene Aspekte. Von **den Entwicklungswerkzeugen** bis zu **den verwendeten Webentwicklungstechnologien** wird das technische Fundament für die Studie gelegt. In Fortsetzung dieser Erkundung wird ein Augenmerk auf die grundlegenden **Konzepte der Webentwicklung** gelegt, die unter dem Titel Konzepte der Webentwicklung vorgestellt werden. Was wäre eine solche Studie ohne einen gründlichen Vergleich zwischen **Web-Anwendung und mobiler Anwendung**? Dabei werden **die Unterschiede zwischen einer Web- und Mobilanwendung** hervorgehoben.

Mit dieser Grundlage wurde erkannt, wie wichtig es ist, zu verstehen, wie diese Arbeit durchgeführt wurde, daher der Abschnitt über **die Methodik**. Von **der Anforderungsanalyse**, mit der die Bedürfnisse des Projekts ermittelt wurden, bis zu den Entwicklungsschritten einer Webanwendung wird jeder Eckpfeiler des Erstellungsprozesses enthüllt. Auch **die Kommunikation zwischen Front- und Backend** wird analysiert, wobei die Bedeutung der Interaktion zwischen Benutzer und Daten hervorgehoben wird. Ein wesentlicher Aspekt eines jeden Technologieprojekts, **das Testen**, wird nicht vergessen. Der Ansatz der Testautomatisierung wird detailliert beschrieben und es wird gezeigt, wie diese Tests durchgeführt und optimiert wurden.

Nachdem diese strengen Schritte durchlaufen wurden, werden die Ergebnisse der Bemühungen, die Früchte der harten Arbeit und die konkreten Auswirkungen der Testautomatisierung im Abschnitt **Bearbeitung und Ergebnisse** enthüllt.

Letztendlich werden **die Zusammenfassung und der Ausblick** realisiert.

2 Stand der Wissenschaft und Technik

Es ist wichtig zu erwähnen, dass die Forschung im Bereich der Entwicklung von Webanwendungen, insbesondere im Zusammenhang mit der Arbeitssuche, intensiv war. Es wurden zahlreiche Forschungsarbeiten und Artikel veröffentlicht, die eine solide Grundlage an Wissen und Techniken für diese Art der Entwicklung bieten. Zu den Schlüsselaspekten dieses Themas gehören nutzerzentriertes Design, Zugänglichkeit, Sicherheit, Effizienz und Zuverlässigkeit.

Die Testautomatisierung spielt bei der Entwicklung moderner Webanwendungen eine entscheidende Rolle und ist eine zunehmend gängige Praxis, um die Effizienz und Zuverlässigkeit von Software zu steigern. Die Testautomatisierung ist entscheidend, um sicherzustellen, dass die Anwendung wie geplant funktioniert, und ermöglicht es, Regressionen und Fehler schnell und zuverlässig zu identifizieren.[16]

Im Hinblick auf die Sicherheit veröffentlicht das Open Web Application Security Project (OWASP) regelmäßig eine Liste der zehn wichtigsten Sicherheitslücken in Webanwendungen, die eine wertvolle Ressource für Entwickler von Webanwendungen darstellt. Sicherheit ist für Anwendungen zur Stellensuche besonders wichtig, da sie häufig sensible Daten wie persönliche Informationen und Karrieredetails der Nutzer verarbeiten.[24]

Die Entwicklung einer Webanwendung für die Stellensuche, unterstützt durch automatisiertes Testen, ist ein etabliertes und sich ständig weiterentwickelndes Forschungsgebiet. Die bestehende Forschungsarbeit bietet eine reiche Informationsquelle über bewährte Verfahren, Herausforderungen und aktuelle Trends in diesem Bereich.

3 Grundlagen

3.1 Entwicklungswerkzeuge

3.1.1 git

Git ist ein verteiltes Versionskontrollsystem, das 2005 von Linus Torvalds, dem Schöpfer von Linux, entwickelt wurde[29]. Es wurde entwickelt, um alles, von kleinen bis zu sehr großen Projekten, schnell und effizient zu verwalten. Git ist leicht zu erlernen und hat einen winzigen Speicherfußabdruck bei blitzschneller Leistung.

Die Versionskontrolle ist ein System, das den Verlauf von Änderungen an einer Datei oder einer Gruppe von Dateien über die Zeit hinweg aufzeichnet, sodass spezifische Versionen später abgerufen werden können. Mit Git besitzt jeder Entwickler eine vollständige lokale Kopie des Projektverlaufs, was eine außergewöhnliche Flexibilität ermöglicht und verschiedene Arbeitsmethoden unterstützt.



Abb. 3.1: Git Übersicht (aus [8])

3.1.2 GitHub

GitHub, Inc. ist ein professioneller Dienst, der cloudbasierte Softwareentwicklung und Versionskontrolle über Git anbietet. Dieser Dienst ermöglicht es Entwicklern, ihre Codebase sicher zu speichern und zu überwachen. GitHub hat seinen Sitz in Kalifornien und seine wachsende Popularität hat die Aufmerksamkeit der Technologieriesen auf sich gezogen und dazu geführt, dass es 2018 von Microsoft übernommen wurde. Diese Übernahme festigte die Position von GitHub als eine der einflussreichsten Plattformen in der Welt der Softwareentwicklung. Die Plattform ist eine

beliebte Wahl für das Hosting von Open-Source-Softwareprojekten. Im Januar 2023 gab GitHub bekannt, dass seine Plattform von mehr als 100 Millionen Entwicklern genutzt wird und die Anzahl der Repositorys 372 Millionen übersteigt.[43]



Abb. 3.2: GitHub Übersicht (aus [7])

Eines der Alleinstellungsmerkmale von GitHub ist die Förderung der Open-Source-Entwicklung. Die Plattform ermutigt Entwickler, ihre Projekte zu veröffentlichen, und fördert so den Wissensaustausch, die Zusammenarbeit zwischen Unternehmen und die Entstehung von Gemeinschaften rund um bestimmte Projekte.

3.1.3 WebStorm

In der dynamischen Welt der Webentwicklung ist der Bedarf an leistungsstarken und spezifisch angepassten Werkzeugen größer denn je. Das von JetBrains entwickelte WebStorm erfüllt diesen Bedarf, indem es sich als IDE präsentiert, die sich hauptsächlich auf moderne Webentwicklungssprachen wie JavaScript, TypeScript, HTML5, Node.js, Bootstrap, Angular/AngularJS konzentriert.[49]



Abb. 3.3: WebStorm Übersicht (aus [18])

3.1.4 NodeJS

Historisch gesehen wurde JavaScript hauptsächlich als clientseitige Programmiersprache betrachtet, die dazu verwendet wurde, Webseiten mit Interaktivität zu versieren. Mit dem Aufkommen von Node.js hat sich die Rolle von JavaScript jedoch erheblich erweitert, was einen wichtigen Wandel in der Landschaft der Webentwicklung darstellt.

Node.js, das 2009 von Ryan Dahl eingeführt wurde, ist eine Open-Source-Plattform, mit der serverseitiger JavaScript-Code ausgeführt werden kann. Dank seiner ereignisbasierten Architektur ist Node.js besonders effizient bei der Verwaltung von Echtzeit-Webanwendungen, API-Diensten und Systemen mit hoher Zugriffskonkurrenz. Diese Plattform bietet Entwicklern die Freiheit, Kommandozeilen-Tools und serverseitige Skripte in JavaScript zu schreiben. Sie wird häufig genutzt, um dynamische Webinhalte zu produzieren, bevor die Seite an den Browser des Nutzers weitergeleitet wird.[47]



Abb. 3.4: Nodejs Übersicht (aus [51])

3.1.5 NPM

In der Programmierung ist die Verwaltung von Abhängigkeiten ein entscheidender Schritt, um die Modularität, Wiederverwendbarkeit und Wartbarkeit des Codes zu gewährleisten. Für die Programmiersprache JavaScript wird diese Verwaltung hauptsächlich von NPM gesteuert, einem Werkzeug, das für Entwickler auf der ganzen Welt unverzichtbar geworden ist. NPM ist ein Paketmanager für die Programmiersprache JavaScript, der von npm, Inc. verwaltet wird. Er ist der Standard-paketmanager für die JavaScript-Laufzeitumgebung, Node.js.[48]

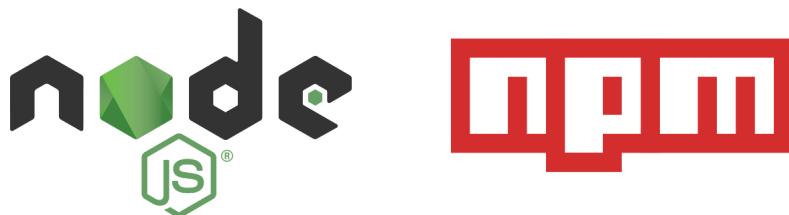


Abb. 3.5: NPM Übersicht (aus [9])

3.2 verwendete Technologien zur Webentwicklung

3.2.1 Angular

Angular ist ein leistungsfähiges, robustes und zunehmend beliebtes Frontend-Framework, das von Google entwickelt wurde. Es wurde 2010 unter dem Namen AngularJS eingeführt, erfuhr mit Version 2 im Jahr 2016 eine wichtige Weiterentwicklung und hat seitdem einen exponentiellen Wachstumspfad eingeschlagen, der

es zu einer beliebten Wahl unter Entwicklern für die Erstellung umfangreicher Webanwendungen gemacht hat.[2]

Eine der herausragenden Eigenschaften von Angular liegt in seiner komponentenbasierten Architektur, welche die Förderung der Modularität, Wiederverwendbarkeit und Wartbarkeit des Codes ermöglicht. Durch diese modulare Architektur haben Entwickler die Möglichkeit, Anwendungen inkrementell aufzubauen, wodurch die Verwaltung umfangreicher Entwicklungsprojekte erleichtert wird.[23]



Abb. 3.6: Angular Übersicht (aus [20])

3.2.2 Html

HTML ist die grundlegende Säule jeder Webentwicklung. Sie wurde Anfang der 1990er Jahre eingeführt und dient als Skelett für alle Webseiten. Sie ermöglicht es Entwicklern, Inhalte zu strukturieren und Informationen organisiert und verständlich darzustellen.[27]

HTML5, die neueste Version von HTML, die 2014 eingeführt wurde, brachte eine Reihe von Verbesserungen und neuen Funktionen mit sich, die zur Vereinfachung und Standardisierung der Webentwicklung beigetragen haben. Zu diesen Neuerungen gehören semantische Elemente wie `<article>`, `<section>` und `<nav>`, die es Entwicklern ermöglichen, aussagekräftigere und zugänglichere Webseitenstrukturen zu erstellen.[25]

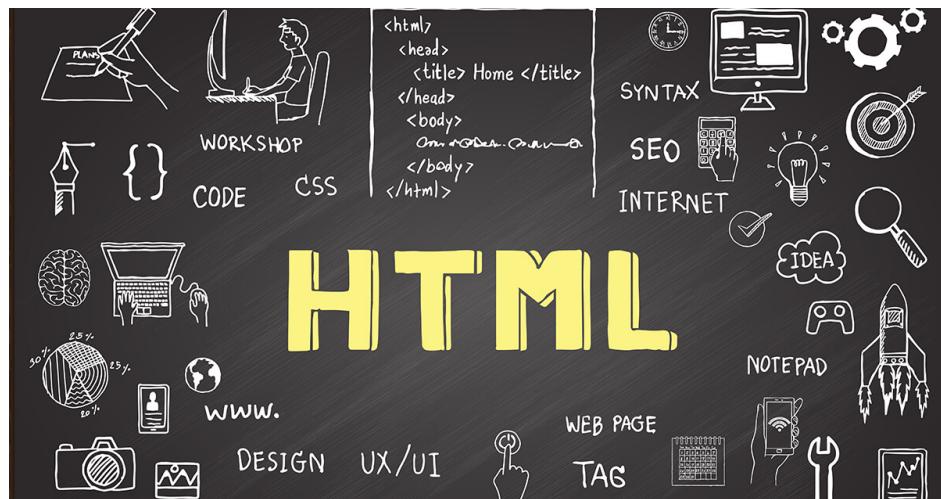


Abb. 3.7: Html Übersicht (aus [11])

3.2.3 TypeScript

TypeScript ist eine höhere Programmiersprache, die 2012 von Microsoft eingeführt wurde. Es ist eine typisierte Obermenge von JavaScript, was bedeutet, dass es JavaScript um Funktionen erweitert, einschließlich eines statischen Typensystems. Alle gültigen JavaScript-Programme sind auch gültige TypeScript-Programme, was die Übernahme von TypeScript für bestehende JavaScript-Entwickler einfacher macht.[36]

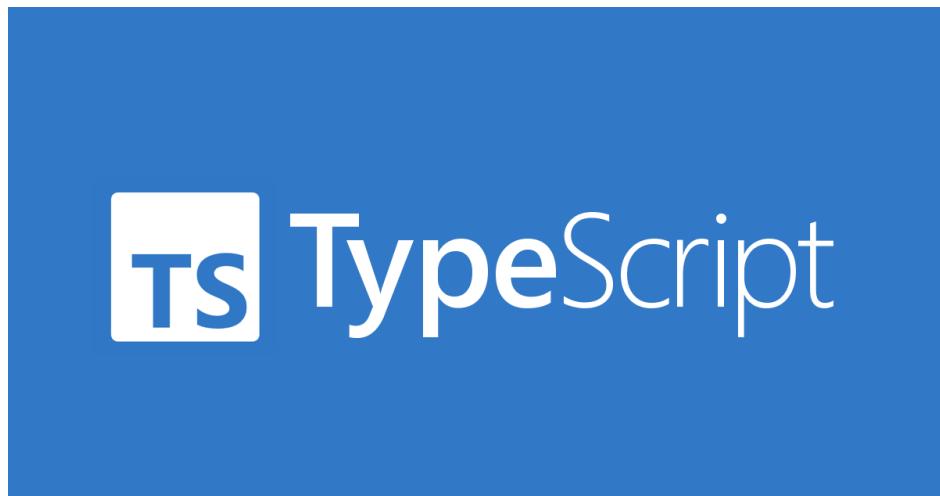


Abb. 3.8: TypeScript Übersicht (aus [37])

3.2.4 JavaScript

JavaScript ist eine Programmiersprache, die 1995 von Brendan Eich entwickelt wurde. Sie wurde ursprünglich für den Netscape Navigator entwickelt, um dynamische Interaktionen in Webseiten zu ermöglichen, und entwickelte sich zur am weitesten verbreiteten Programmiersprache für die clientseitige Webentwicklung. Im Laufe der Jahre hat sich JavaScript weiterentwickelt und mit der Einführung von Plattformen wie Node.js auch auf die serverseitige Entwicklung ausgedehnt.[14]



Abb. 3.9: JavaScript Übersicht (aus [13])

Trotz vieler Kritikpunkte hat JavaScript den Test der Zeit bestanden und ist nach wie vor eine der beliebtesten und meistgenutzten Programmiersprachen der Welt[5]. Seine Allgegenwärtigkeit in der Webentwicklung, seine ständige Weiterentwicklung und seine Flexibilität machen es zu einem wichtigen Thema für jede Studie über Softwareentwicklung.

3.2.5 CSS

CSS ist eine Stylesheet-Sprache, die zur Beschreibung des Layouts eines in HTML oder XML geschriebenen Dokuments verwendet wird. CSS wurde vom World Wide Web Consortium (W3C) entwickelt und 1996 zum ersten Mal veröffentlicht. Es gehört neben HTML und JavaScript zu den Grundpfeilern der Webentwicklung.[50]

3.3 Konzepte der Webentwicklung

Die schnelle und stetige Entwicklung der digitalen Welt hat die Art und Weise, wie wir mit Informationen und Diensten miteinander interagieren, verändert. Im Zentrum dieses Wandels steht die Webentwicklung, eine Disziplin, die die Architektur des Internets, wie wir es heute kennen, prägt und belebt.

3.3.1 HTTP

Zu Beginn der digitalen Revolution, als die Grundlagen für das Internet geschaffen wurden, wurde schnell klar, dass ein universelles System benötigt wurde, mit dem Maschinen miteinander kommunizieren können. In diesem Zusammenhang entstand das HTTP, das schnell zum Standard für die Webkommunikation wurde.

HTTP ist ein Protokoll der Anwendungsschicht, das festlegt, wie Nachrichten formatiert und über das Web übertragen werden und wie Webserver und Browser auf diese Nachrichten reagieren sollen. Es funktioniert als Anfrage-Antwort-Verfahren zwischen dem Client (normalerweise ein Webbrowser) und dem Server, was für das Modell des Webs, wie wir es kennen, von entscheidender Bedeutung war. Die Entwicklung des HTTP-Protokolls wurde 1989 von Tim Berners-Lee begonnen und führte zu einem ersten Dokument, das das Verhalten eines Clients und eines Servers durch die erste Version von HTTP definierte, die als Version 0.9 bezeichnet wurde. Diese Version wurde später weiterentwickelt und führte schließlich zur Mainstream-Version 1.0.[44]

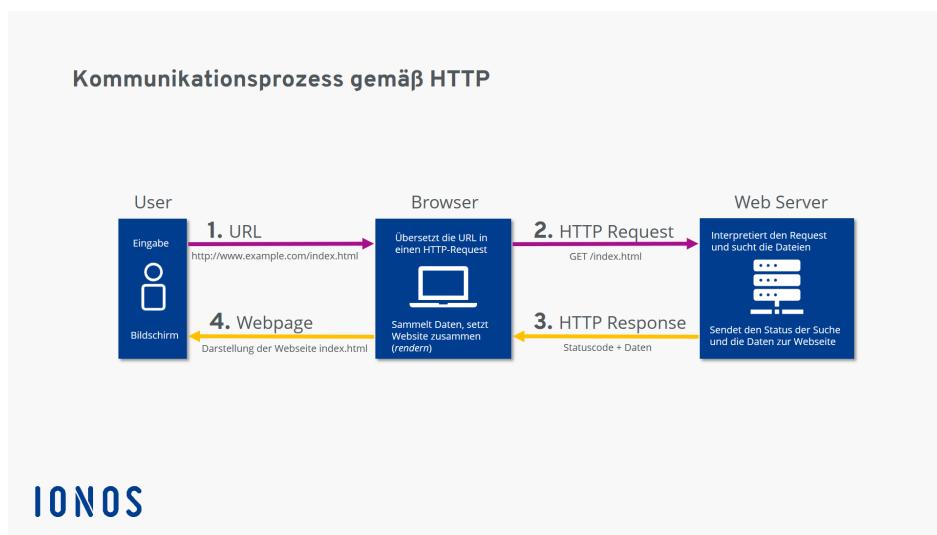


Abb. 3.10: Schematische Darstellung des Kommunikationsprozesses gemäß HTTP-Protokoll (aus [12])

3.3.2 JSON

Als die digitale Welt immer komplexer wurde und die Notwendigkeit, Informationen zwischen heterogenen Systemen auszutauschen, vorherrschte, wurde der Bedarf an einem schlanken, leicht lesbaren und allgemein akzeptierten Format für den Datenaustausch offensichtlich. Hier kam JSON ins Spiel, das sich schnell zum De-facto-Standard für die Übertragung strukturierter Daten über das Web entwickelte.

JSON ist ein offenes Standard-Dateiformat und ein Datenaustauschformat, das menschenlesbaren Text verwendet, um Datenobjekte, die aus Attribut-Wert-Paaren und Tabellen bestehen, zu speichern und zu übertragen. Obwohl JSON seine Ursprünge in der Programmiersprache JavaScript hat, handelt es sich um ein sprachunabhängiges Datenformat. Dateien, die dieses Format verwenden, nehmen die Erweiterung

.json an. Douglas Crockford war der erste, der das JSON-Format Anfang des 21. Jahrhunderts definierte. Im April 2001 übermittelten sie mit Hilfe von Chip Morningstar die allererste Nachricht in JSON.[45]



Abb. 3.11: JSON Übersicht (aus [3])

3.3.3 API

Im Zeitalter der digitalen Vernetzung ist die Notwendigkeit, eine reibungslose und zuverlässige Kommunikation zwischen Anwendungen, Plattformen und Diensten herzustellen, wichtiger denn je. Hier kommt das Konzept der API ins Spiel, das eine entscheidende Rolle bei der Erleichterung dieser Interaktionen spielt.

Eine API ist ein Satz von Regeln und Spezifikationen, über den eine Anwendung oder Software mit einer anderen kommunizieren kann. Sie definiert die Methoden und Datenstrukturen, die Entwickler verwenden können, um zu interagieren und auf Funktionen oder Daten in einer externen Software zuzugreifen. Im Gegensatz zu einer Benutzeroberfläche, die die Interaktion zwischen einem Computer und einem Benutzer erleichtert, sorgt eine API für die Kommunikation zwischen verschiedenen Maschinen oder Softwareprogrammen. Sie ist für die Verwendung durch Programmierer und nicht direkt durch Endbenutzer gedacht. APIs dienen in erster Linie dazu, die interne Komplexität eines Systems zu verschleiern, indem sie nur die für den Programmierer relevanten Elemente offenlegen und ihre Konsistenz auch dann gewährleisten, wenn interne Details geändert werden.[40]

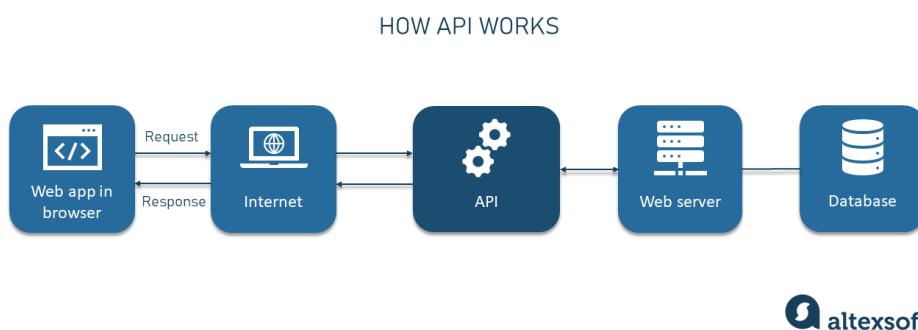


Abb. 3.12: HOW API WORKS (aus [1])

3.4 Webanwendung

Eine Webanwendung ist im Wesentlichen ein Computerprogramm, das einen Webbrowser als Benutzerschnittstelle verwendet. Im Gegensatz zu herkömmlicher Software, die eine spezielle Installation auf dem Computer des Nutzers erfordert, liegt eine Webanwendung auf einem entfernten Server und ist über das Internet zugänglich. Sie wird mithilfe von standardisierten Sprachen wie HTML, CSS und JavaScript erstellt.[15]

3.4.1 wie funktioniert eine Webanwendung?

Die Funktionsweise einer Webanwendung ist zwar hinter den Kulissen komplex, lässt sich aber zum besseren Verständnis in einige grundlegende Schritte unterteilen. Eine Webanwendung basiert auf einem Client-Server-Modell, bei dem der Client (in der Regel ein Webbrowser) Ressourcen oder Dienste von einem Server anfordert und der Server entsprechend antwortet. Hier eine detaillierte Beschreibung des Prozesses :

1. Anfrage des Kunden

- Alles beginnt, wenn der Nutzer eine URL in den Browser eingibt oder auf einen Link klickt. In diesem Moment sendet der Browser eine Anfrage an den Webserver, auf dem die Anwendung gehostet wird.
- Diese Anfrage wird normalerweise mit dem HTTP-Protokoll (oder seiner sicheren Version, HTTPS) formuliert

2. Verarbeitung der Anfrage durch den Server

- Sobald die Anfrage eingegangen ist, wird sie vom Server verarbeitet. Wenn die Anwendung dynamisch ist (wie die meisten modernen Anwendungen), muss der Server möglicherweise mit einer Datenbank interagieren, um Informationen abzurufen oder zu speichern.

3. Antwort des Servers

- Nachdem der Server die Anfrage verarbeitet hat, sendet er eine Antwort. Diese Antwort besteht in der Regel aus HTML-, CSS- und JavaScript-Dateien sowie eventuellen Multimediateilen wie Bildern oder Videos.
- Der Server verwendet auch das HTTP-Protokoll, um diese Antwort zu senden.

4. Anzeige des Browsers

- Sobald der Browser die Antwort des Servers erhalten hat, beginnt er, sie zu verarbeiten. Er interpretiert den HTML-, CSS- und JavaScript-Code, um die Webseite zu erstellen und anzuzeigen.
- JavaScript kann verwendet werden, um der Seite Interaktivität hinzuzufügen, z. B. Animationen oder Reaktionen auf Benutzeraktionen

5. Zusätzliche Interaktionen

- Sobald die Seite geladen ist, führt jede neue Interaktion, die nicht lokal (d. h. direkt im Browser) verarbeitet werden kann, zu neuen Anfragen an

den Server. Wenn z. B. ein Beitrag in einem sozialen Netzwerk geliked wird, wird eine neue Anfrage gesendet, um diese Aktion zu registrieren.

6. Session und Cookies

- Webanwendungen verwenden häufig Cookies, um den Überblick über die Nutzer und ihre Interaktionen zu behalten. Ein Cookie ist ein kleines Stück Daten, das im Browser des Nutzers gespeichert wird.
- Session ermöglichen es den Servern, Informationen über den Nutzer während einer Interaktionsperiode zu speichern. Sie können z. B. verwendet werden, um herauszufinden, ob ein Nutzer angemeldet ist und welche Einstellungen er hat.

Die Funktionsweise einer Webanwendung beruht auf einem ständigen Zyklus von Anfragen und Antworten zwischen dem Client (Browser) und dem Server. Moderne Technologien haben es ermöglicht, diesen Prozess zu beschleunigen und flüssiger zu gestalten und so immer reichhaltigere und reaktionsschnellere Nutzererfahrungen zu bieten.

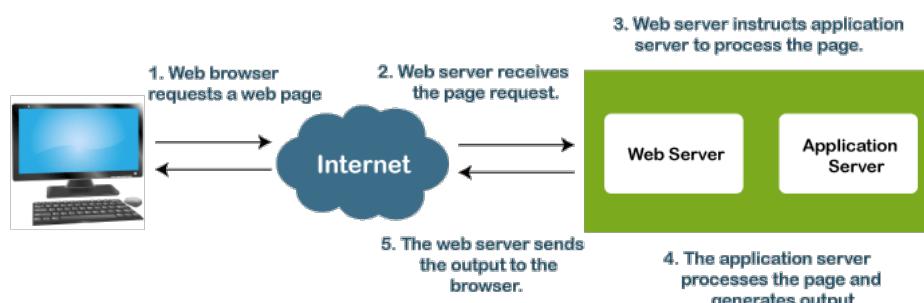


Abb. 3.13: The Flow of the Web Application (aus [15])

3.5 Mobilanwendung

Mobile Anwendungen, oft einfach als Apps bezeichnet, sind Software, die für den Betrieb auf mobilen Geräten wie Smartphones oder Tablets entwickelt wurde. Ihre Funktionsweise unterscheidet sich teilweise von der von Webanwendungen, obwohl es einige Ähnlichkeiten gibt.[32]

Bei der Konzeption von mobilen Anwendungen und ihrem Einsatz müssen die Besonderheiten der verschiedenen mobilen Betriebssysteme und die Fähigkeiten der Geräte selbst berücksichtigt werden.

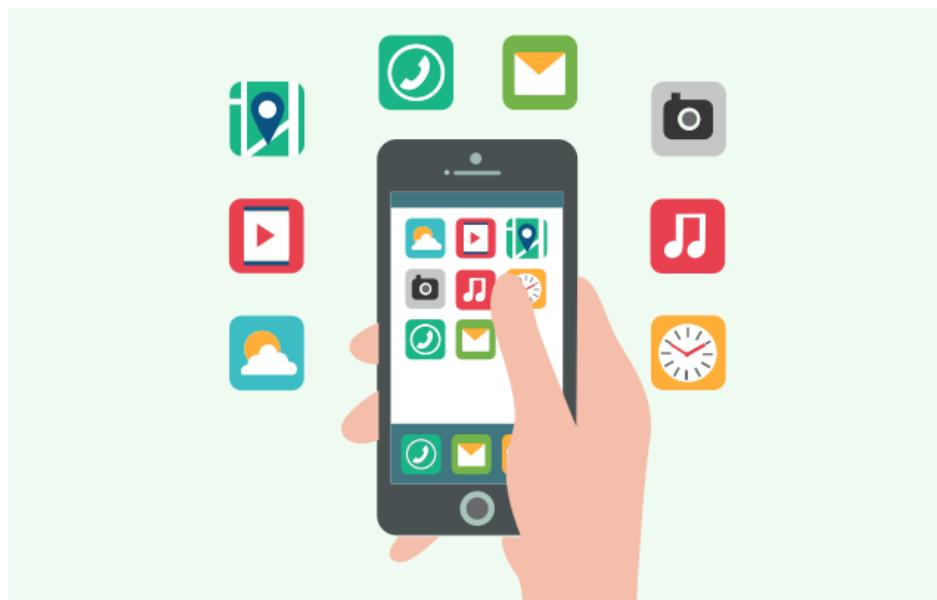


Abb. 3.14: mobile application Übersicht (aus [26])

3.5.1 Funktionsweise einer mobilen Anwendung

Hier ist eine detaillierte Beschreibung, wie eine mobile Anwendung funktioniert :

1. Installation

- Im Gegensatz zu Webanwendungen, die über einen Browser aufgerufen werden können, müssen mobile Anwendungen in der Regel auf dem Gerät installiert werden. Diese Installation erfolgt in der Regel über App-Stores, wie den App Store für iOS oder Google Play für Android.

2. Interaktionen mit dem Betriebssystem

- Mobile Anwendungen interagieren direkt mit dem Betriebssystem des Geräts (z. B. iOS, Android, Windows Mobile). Dadurch können sie auf bestimmte Funktionen des Geräts zugreifen, z. B. auf die Kamera, GPS, Kontakte etc.

3. Lokale Ausführung

- Nach der Installation kann die mobile Anwendung lokal auf dem Gerät ausgeführt werden, ohne dass eine ständige Verbindung zu einem entfernten Server erforderlich ist. Wesentliche Ressourcen (wie die Benutzeroberfläche, die grundlegende Logik usw.) werden während der Installation mit der Anwendung verpackt.

4. Speicherung von Daten

- **Lokale Speicherung:** Die Anwendung kann Daten direkt auf dem Gerät speichern und dabei lokale Datenbanken, Dateien oder freigegebene Voreinstellungen verwenden.
- **Remote-Speicherung:** Für Daten, auf die von mehreren Geräten aus zugegriffen werden soll oder die sicher gespeichert werden müssen, kommuniziert die Anwendung in der Regel über das Internet mit einem Remote-Server (ähnlich wie Webanwendungen)

5. Kommunikation mit Servern

- Obwohl eine mobile Anwendung lokal funktionieren kann, müssen viele Anwendungen mit entfernten Servern interagieren, um Updates abzurufen, Daten zu synchronisieren usw. Diese Kommunikation erfolgt in der Regel über APIs unter Verwendung von Protokollen wie HTTP/HTTPS

6. Updates

- Mobile Anwendungen erhalten über die App-Stores Aktualisierungen. Entwickler können Updates veröffentlichen, um Fehler zu beheben, Funktionen hinzuzufügen oder die Leistung zu verbessern. Die Nutzer laden diese Updates herunter und installieren sie, um die neueste Version zu erhalten

3.6 Unterschied zwischen Web- und Mobilanwendung

Mobile Anwendungen und Webanwendungen haben unterschiedliche Merkmale, auch wenn sie manchmal ähnliche Funktionen bieten können. Hier sind die wichtigsten Unterschiede zwischen den beiden :

Eigenschaften	Webanwendungen	Mobilanwendungen
Plattform und Zugang	Auf sie kann über einen Webbrower (wie Chrome, Firefox, Safari) zugegriffen werden, sie müssen weder heruntergeladen noch installiert werden[35]	Sie werden entwickelt, um auf mobilen Geräten wie Smartphones und Tablets installiert und ausgeführt zu werden.[35]
Entwicklung und Kompatibilität	Sie werden in der Regel so entwickelt, dass sie mit mehreren Browsern kompatibel sind.	Sie erfordern eine spezielle Entwicklung für jedes Betriebssystem (z. B. iOS, Android).
Updates	Updates sind für den Nutzer in der Regel transparent. Sobald eine Aktualisierung auf dem Server bereitgestellt wird, haben die Nutzer beim nächsten Besuch der Webanwendung sofort Zugriff darauf.	Updates müssen über App-Stores heruntergeladen und installiert werden, was ein Eingreifen des Nutzers erfordert.
Kosten und Wartung	Sie bieten in der Regel niedrigere Anfangskosten und geringere Wartungskosten, da nur eine einzige Codebasis zu verwalten ist.	Die Entwicklungskosten können höher sein, vor allem wenn man auf mehrere Plattformen abzielt. Außerdem kann die Wartung für jede Plattform einen separaten Aufwand erfordern.

Tab. 3.1: Unterschied zwischen Web- und Mobilanwendung.

4 Methodik

4.1 Anforderungsanalyse

4.1.1 Analyse funktionaler Anforderungen

Im Rahmen dieses Projekts werden die folgenden Funktionen umgesetzt:

1. Erstellung eines Benutzerkontos

- Der Nutzer muss in der Lage sein, ein Anmeldeformular mit seinen persönlichen Informationen auszufüllen.
- Der Nutzer muss bei der Erstellung seines Kontos die Option *employe* oder *jobseeker* wählen können.
- Das System muss die vom Benutzer eingegebenen Informationen überprüfen und ein entsprechendes Benutzerkonto erstellen.

2. Anmeldung bei einem bestehenden Benutzerkonto

- Der Benutzer muss seinen Benutzernamen und sein Passwort eingeben können, um sich anzumelden.
- Das System muss den Benutzer authentifizieren und ihm den Zugriff auf sein Benutzerkonto ermöglichen.

3. Jobsuche

- Das System soll die Datenbank nach Stellenangeboten durchsuchen und sie dem Nutzer anzeigen.

4. Sich auf ein Stellenangebot bewerben

- Der *employe* sollte in der Lage sein, sich die Details einer Stellenanzeige anzusehen und zu entscheiden, ob er sich bewerben möchte.
- Das System muss dem Nutzer die Möglichkeit bieten, eine Bewerbung für die ausgewählte Stellenanzeige einzureichen.

5. Ergebnisse der Bewerbungen

- Der *employe* muss in der Lage sein, die vom *jobseeker* eingereichten Bewerbungen zu empfangen und zu bearbeiten.
- Der *jobseeker* muss die Ergebnisse seiner Bewerbungen (positiv oder negativ) einsehen können.

6. Speicherung von Dateien

- Der Benutzer muss verschiedene Dokumente (Aufenthaltstitel, Lebenslauf, Arbeitserlaubnis, Personalausweis) in der Datenbank speichern können.

- Die von dem Benutzer gespeicherte persönlichen Dokumente sollen dem Arbeitgeber bei der Bewertung von Bewerbungen zur Verfügung stehen.

7. Abmelden des Benutzers

- Der Benutzer muss die Möglichkeit haben, sich von seinem Konto abzumelden.

8. Stellenangebote durch den Arbeitgeber veröffentlichen

- Der Arbeitgeber muss die Möglichkeit haben, Stellenangebote zu posten.
- Die Veröffentlichung ist sofort auf der Plattform sichtbar.

9. Abfrage von Stellenangeboten

- Der Nutzer muss alle auf der Plattform verfügbaren Stellenangebote einsehen können.

10. Empfang von Bewerbungen durch den Arbeitgeber

- Der Arbeitgeber muss die Möglichkeit haben, Bewerbungen für seine Stellenangebote zu empfangen und einzusehen.
- Übersichtliche Schnittstelle, die alle eingegangenen Bewerbungen auflistet.
- Möglichkeit, jede Bewerbung zu öffnen, um Details und angehängte Dokumente zu sehen.

11. Tests

- Das System muss auf verschiedenen Ebenen getestet werden, um seine Zuverlässigkeit und Effizienz zu gewährleisten.

4.1.2 Analyse nicht-funktionaler Anforderungen

4.1.2.1 Sicherheit

Die Wahrung der Sicherheit spielt eine entscheidende Rolle für den Erfolg und die Akzeptanz der Anwendung durch die Nutzer. Diese Anforderung zielt darauf ab, die Integrität sensibler Daten und die Vertraulichkeit der persönlichen Informationen der Nutzer zu gewährleisten sowie Versuche des Eindringens oder der böswilligen Ausnutzung zu verhindern.

Die Gewährleistung der Sicherheit der Webanwendung ist von größter Bedeutung, um das Vertrauen der Nutzer zu erhalten. In einem Umfeld, in dem die Cyberkriminalität stetig zunimmt, achten die Nutzer verstärkt auf den Schutz ihrer persönlichen Daten. Durch die konsequente Erfüllung dieser Anforderung kann sich die Anwendung als zuverlässig und vertrauenswürdig positionieren und so die Nutzer dazu bewegen, ihre Informationen in aller Ruhe zu teilen, ohne eine Verletzung ihrer Privatsphäre befürchten zu müssen.



Abb. 4.1: Web Application security against Cyber Attack(aus [34])

4.1.2.2 Leistung

Die Leistung ist ein wesentlicher Aspekt bei der Entwicklung einer Webanwendung, da sie direkten Einfluss auf die Reaktionsfähigkeit, die Ausführungsgeschwindigkeit und die Gesamteffizienz der Anwendung hat. Eine leistungsfähige Anwendung ist unerlässlich, um eine reibungslose und zufriedenstellende Nutzererfahrung zu gewährleisten, und trägt damit direkt zur Zufriedenheit der Nutzer und zum Erfolg des Projekts bei.

Ein Schlüsselaspekt der Anwendungsleistung ist die Ladezeit der Seiten. Schnelle Ladezeiten sind entscheidend, um die Aufmerksamkeit der Nutzer zu wecken und sie zu ermutigen, die Anwendung weiter zu erkunden. Eine reaktionsschnelle Oberfläche ermöglicht es den Nutzern außerdem, die gesuchten Informationen oder Funktionen schnell zu finden, wodurch das Gesamterlebnis verbessert wird. Eine Anwendung, die prompt auf Nutzeraktionen wie Klicks oder Interaktionen mit Elementen der Benutzeroberfläche reagiert, erzeugt ein Gefühl von Flüssigkeit und Effizienz. Im Gegensatz dazu kann eine langsame, ruckelige Anwendung die Nutzer frustrieren und sie davon abhalten, die Anwendung weiter zu nutzen.

4.1.2.3 Benutzerfreundlichkeit

Das Ziel einer benutzerfreundlichen Webanwendung ist es, eine intuitive, angenehme und mühelose Benutzererfahrung zu bieten. Um dieses Ziel zu erreichen, stützt sie sich auf klare, gut organisierte und leicht verständliche Benutzeroberflächen. Wenn die Nutzer leicht mit der Anwendung interagieren können, sind sie eher geneigt, sie anzunehmen und regelmäßig zu nutzen, was zum Erfolg des Projekts beiträgt.

Die Benutzerfreundlichkeit spielt eine herausragende Rolle für die Zufriedenheit der Nutzer. Eine gut gestaltete Benutzeroberfläche, die eine flüssige Navigation und natürliche Interaktionen bietet, ermöglicht es den Nutzern, ihre Aufgaben effizienter und schneller zu erledigen.

Sicherheit	Die Sicherheit einer Webanwendung ist entscheidend für das Vertrauen und die Akzeptanz der Nutzer, insbesondere in einem Zeitalter wachsender Cyberkriminalität.
Leistung	Die Leistung einer Webanwendung, insbesondere schnelle Ladezeiten und Reaktionsfähigkeit, ist entscheidend für eine zufriedenstellende Nutzererfahrung und den Erfolg des Projekts.
Benutzerfreundlichkeit	Eine benutzerfreundliche Webanwendung bietet durch ihre intuitive und klar strukturierte Benutzeroberfläche eine angenehme Erfahrung, wodurch Nutzer sie gerne und effizient nutzen, was zum Erfolg des Projekts beiträgt.

Tab. 4.1: Nicht-funktionaler Anforderungen.

4.2 Schritte zur Entwicklung einer Webanwendung

4.2.1 Analyse der Bedürfnisse und Ziele

In diesem Schritt werden die Anforderungen und Ziele des Projekts sowie die Bedürfnisse der Endbenutzer eingehend analysiert. Dieser erste Schritt ist von entscheidender Bedeutung, um den Umfang und die Funktionen der Anwendung klar abzugrenzen und damit eine solide Grundlage für den gesamten Entwicklungsprozess zu schaffen. Sie bedeutet, dass die Anforderungen und Ziele des Projekts gesammelt, verstanden und genau definiert werden müssen, bevor die Entwicklung beginnt. Diese Phase ist von entscheidender Bedeutung, da sie den Rahmen für das Projekt vorgibt, die Entscheidungsfindung beim Design lenkt und sicherstellt, dass die entwickelte Anwendung die Erwartungen der Nutzer und die festgelegten Geschäftsziele erfüllt.

4.2.2 Design der Anwendung

Die Designphase ist bei der Erstellung einer Webanwendung von entscheidender Bedeutung. Ihr Ziel ist es, eine umfassende Vision der Anwendung zu entwickeln, indem ihre Architektur, Struktur, Benutzeroberfläche (UI) und Benutzererfahrung (UX) genau definiert werden. Als wesentlicher Bestandteil des Prozesses spielt diese Phase eine entscheidende Rolle für den Erfolg des Projekts, da sie dafür sorgt, dass die während der Analyse ermittelten Anforderungen und Ziele in einen konkreten und kohärenten Plan umgewandelt werden, der als Grundlage für die Entwicklung dient.

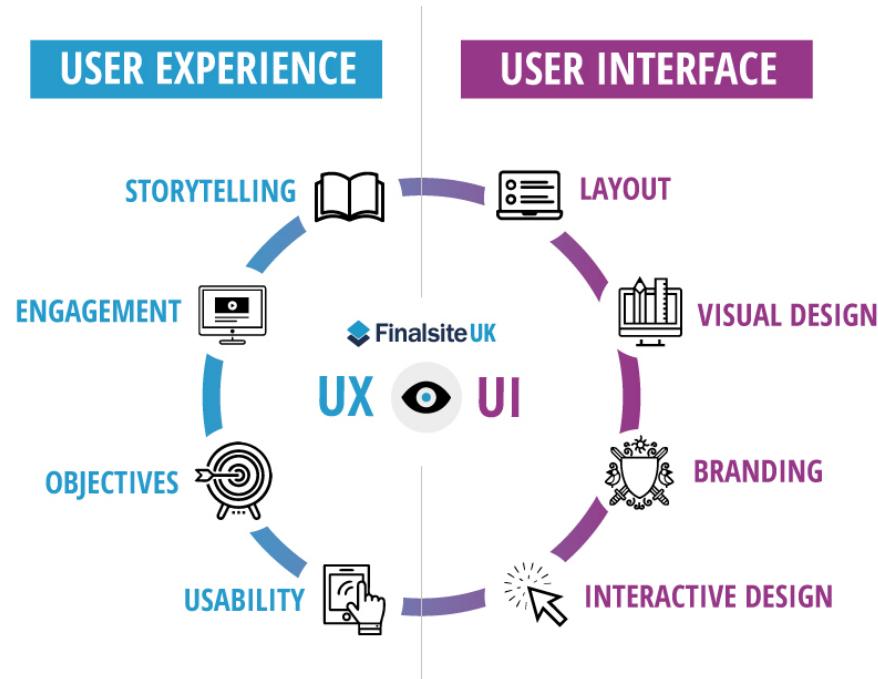


Abb. 4.2: User Experience, User Interface(aus [38])

4.2.3 Back-End-Entwicklung

Die Backend-Entwicklung ist die Phase bei der Erstellung einer Webanwendung, in der die für den Benutzer unsichtbaren, aber dennoch für das reibungslose Funktionieren der Anwendung wichtigen Prozesse gesteuert werden. Diese Phase betrifft den Server, die Datenbank und die Serveranwendungen (d. h. die API), die die Funktionen der Anwendung versorgen.

Die Backend-Entwicklung beinhaltet hauptsächlich die Erstellung einer API, die die Kommunikation zwischen dem Front-End der Anwendung und dem Server ermöglicht. Dies kann mithilfe einer Vielzahl von Programmiersprachen wie JavaScript (Node.js), geschehen. Die API empfängt Anfragen vom Frontend, interagiert bei Bedarf mit der Datenbank und gibt die angeforderten Daten an das Frontend zurück.[30]

Die Verwaltung der Datenbank ist ein weiterer wichtiger Aspekt der Backend-Entwicklung. Datenbanken speichern die für die Anwendung benötigten Informationen, wie z. B. Benutzerdetails, Nachrichten, Dateien usw.

4.2.4 Front-End-Entwicklung

Dieser Teil der Entwicklung befasst sich mit der Benutzeroberfläche und allem, was für den Benutzer sichtbar ist. Frontend-Entwickler verwenden Programmiersprachen wie HTML, CSS und JavaScript, um das Design, die Benutzeroberfläche und das interaktive Verhalten einer Webanwendung zu gestalten.

Die Frontend-Entwicklung stellt eine anspruchsvolle Aufgabe dar, die eine harmonische Verbindung von Programmier-, Design- und UX-Kompetenzen erfordert. Sie spielt eine entscheidende Rolle bei der Schaffung einer positiven Nutzererfahrung

und ist direkt verantwortlich für das Erscheinungsbild, das Gefühl und die Interaktivität einer Webanwendung.

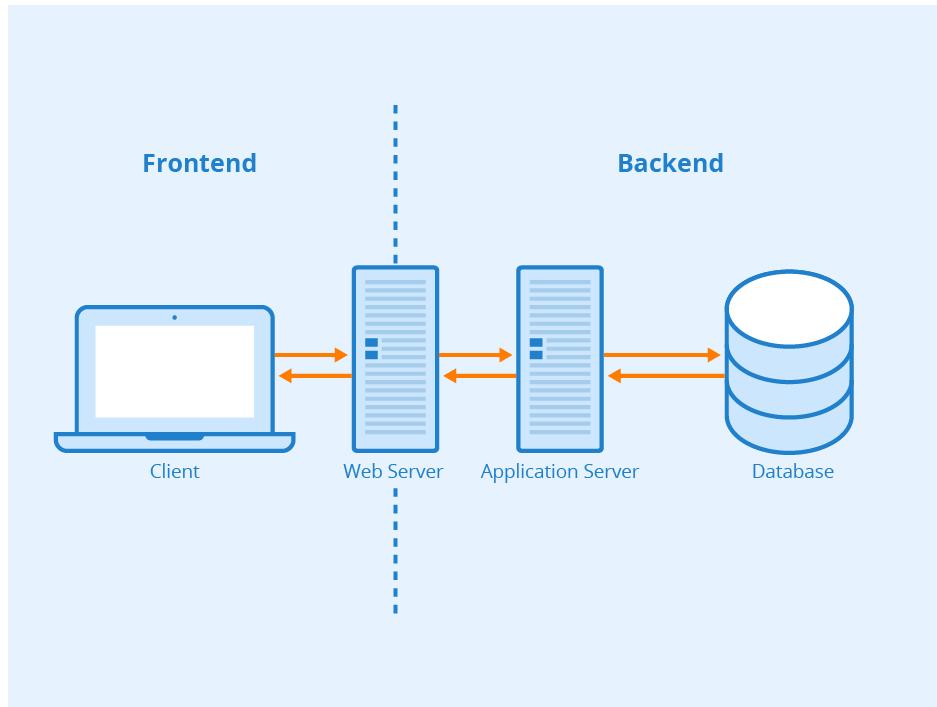


Abb. 4.3: Back-End vs. Front-End Development(aus [33])

4.2.5 Durchführung der Tests

Diese Phase stellt sicher, dass die Anwendung wie geplant funktioniert, und ermöglicht es, Fehler zu erkennen und zu beheben, bevor sie sich auf die Endbenutzer auswirken.

Tests finden in der Regel auf mehreren Ebenen statt:

- **Unit-Tests** zum Beispiel überprüfen, ob die einzelnen Komponenten der Anwendung richtig funktionieren. Sie konzentrieren sich auf isolierte Teile des Codes, um sicherzustellen, dass sie unter verschiedenen Bedingungen richtig funktionieren.[19]
- **Integrationstests** prüfen, ob die einzelnen Komponenten der Anwendung richtig zusammenarbeiten. Sie können z. B. testen, wie die verschiedenen Teile der Anwendung mit der Datenbank oder anderen externen Diensten interagieren.[17]
- **End-to-End-Tests (E2E)** simulieren die gesamte Benutzererfahrung, um sicherzustellen, dass der gesamte Interaktionsfluss wie erwartet funktioniert.

4.2.6 Deployment der Webanwendung

Zu diesem Zeitpunkt wird die Anwendung, die entworfen, entwickelt, getestet und freigegeben wurde, schließlich für die Endbenutzer im Internet verfügbar gemacht.

Eine der ersten Entscheidungen, die bei der Bereitstellung getroffen werden müssen, ist die Wahl des Hostings. Dabei kann es sich um einen dedizierten Server, einen gemeinsam genutzten Server oder eine Umgebung in der Cloud handeln. Jede Option hat ihre eigenen Vor- und Nachteile in Bezug auf Kosten, Leistung, Sicherheit und Flexibilität[10].

Hosting-Anbieter wie Amazon Web Services⁵, Google Cloud⁶ und Microsoft Azure³ werden häufig verwendet, um moderne Webanwendungen zu hosten.

Wenn der Server vorbereitet und konfiguriert wurde, wird dann der Code der Anwendung auf den Server übertragen. Dies kann manuell geschehen, aber es ist üblich, Werkzeuge für kontinuierliche Integration/continuous deployment (CI/CD) zu verwenden, um diesen Prozess zu automatisieren¹. Diese Tools, wie Jenkins² oder GitLab CI/CD³, stellen die Anwendung automatisch bereit, wenn der Quellcode geändert wird.

Als nächstes muss die Domäne der Anwendung konfiguriert werden. Dies bedeutet in der Regel, dass Sie einen Domainnamen⁴ von einem Registrar kaufen und diesen dann so konfigurieren, dass er auf die IP-Adresse des Servers verweist.

Es ist auch wichtig, ein SSL-Zertifikat einzurichten, um die Kommunikation zwischen der Anwendung und den Nutzern zu sichern. Dadurch werden die ausgetauschten Daten verschlüsselt, was das Abfangen durch böswillige Akteure erheblich erschwert.[39]

4.3 Kommunikation zwischen Front-End und Back-End

Die Kommunikation zwischen Frontend und Backend erfolgt in der Regel mithilfe von HTTP-Anfragen.[31] Dies funktioniert folgendermaßen:

- **Anfrage:** Wenn ein Benutzer eine Aktion im Frontend ausführt (z. B. auf eine Schaltfläche klickt, um ein Formular abzuschicken), kann dies eine HTTP-Anfrage auslösen, die an das Backend gesendet wird. Diese Anfrage kann unter anderem eine GET-Methode (zum Abrufen von Daten), POST-Methode (zum Senden von Daten), PUT-Methode (zum Aktualisieren von Daten) oder DELETE-Methode (zum Löschen von Daten) sein.
- **Verarbeitung:** Sobald das Backend die Anfrage erhalten hat, verarbeitet es sie entsprechend seiner Geschäftslogik. Dazu kann die Validierung der Daten, die Interaktion mit einer Datenbank zum Abrufen, Speichern oder Ändern von Daten, oder der Aufruf anderer Dienste usw. gehören.
- **Antwort:** Nachdem das Backend die Anfrage bearbeitet hat, sendet es eine Antwort an das Frontend zurück. Diese Antwort kann eine einfache Bestätigung, angeforderte Daten oder ein Fehler sein.
- **Anzeige:** Das Frontend verarbeitet dann die erhaltene Antwort und aktualisiert die Benutzeroberfläche entsprechend. Wenn der Benutzer z. B. Daten angefordert hat, können diese auf der Seite angezeigt werden. Wenn ein Fehler aufgetreten ist, kann eine Fehlermeldung angezeigt werden.

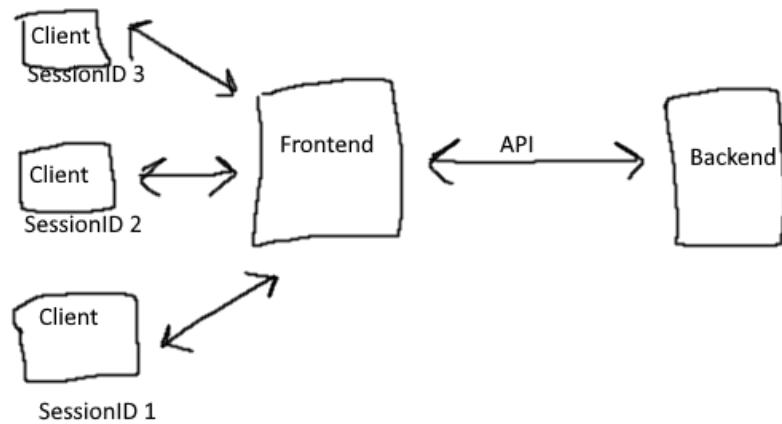


Abb. 4.4: Communication between Frontend and Backend (aus [31])

4.4 Testautomatisierung

5 Bearbeitungsphase

5.1 Verwendete Bibliotheken

5.1.1 jwt

JSON Web Token (JWT) ist ein Internetstandard, der verwendet wird, um Informationen zwischen Parteien auf sichere Weise zu übertragen. Diese Informationen werden in JSON kodiert. JWT wird häufig für die Authentifizierung und Autorisierung in Webanwendungen verwendet. [46] Es gibt drei Schlüsselkomponenten in einem JWT:

- **Header:** er enthält normalerweise den Typ des Tokens (JWT) und den verwendeten Signaturalgorithmus.
- **Payload:** sie enthält Aussagen über eine Entität (oft einen Benutzer) und zusätzliche Daten.
- **Signatur:** sie stellt sicher, dass die Nachricht auf dem Weg nicht verändert wurde

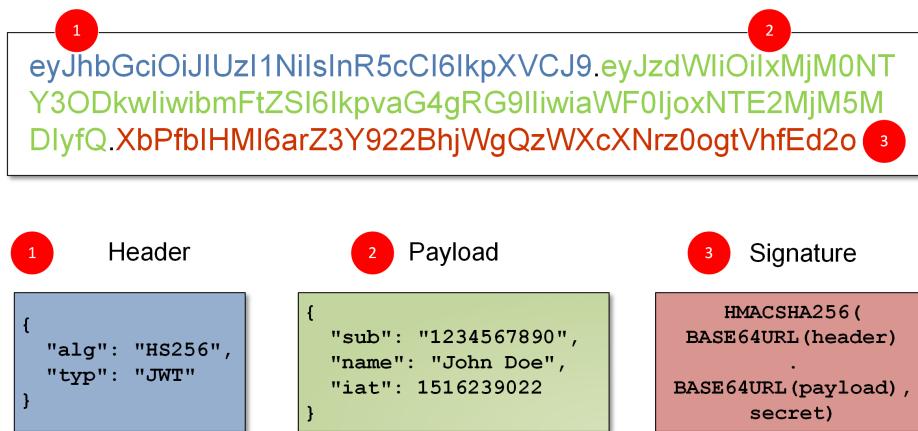


Abb. 5.1: Json Web Token Übersicht (aus [28])

5.1.2 Express

In einem sich ständig verändernden Ökosystem der Webentwicklung ist der Bedarf an effizienten und flexiblen Werkzeugen zur Entwicklung von Webanwendungen und APIs von entscheidender Bedeutung. Express.js, oft einfach nur Express genannt, präsentiert sich als eine Antwort auf diese Anforderung. Es handelt sich um ein Framework für serverseitige Webanwendungen, das speziell für Node.js entwickelt wurde, die beliebte Runtime, mit der serverseitiges JavaScript ausgeführt

werden kann. Sein ursprünglicher Schöpfer, TJ Holowaychuk, hat Express.js so konzipiert, dass es minimalistisch ist und dennoch die Möglichkeit bietet, über Plugins zahlreiche Funktionen hinzuzufügen.[42]



Abb. 5.2: Express Übersicht (aus [4])

5.1.3 Bcrypt

bcrypt ist eine Hashfunktion für Passwörter, die für ihre Robustheit und Sicherheit bekannt ist. Sie wurde von Niels Provos und David Mazières entworfen, basiert auf der Blowfish-Verschlüsselung und wurde erstmals 1999 auf der USENIX-Konferenz vorgestellt. Die bcrypt-Funktion wird als Standard-Passwort-Hash-Algorithmus für OpenBSD übernommen.[41]

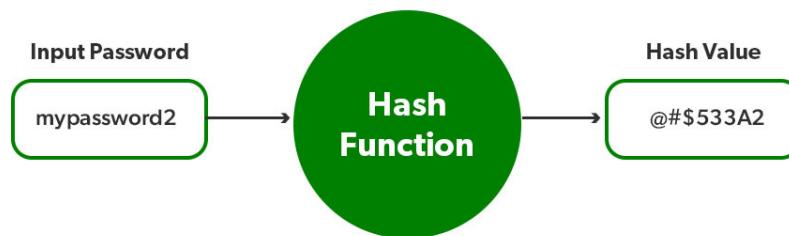


Abb. 5.3: Hashing Passwords with Bcrypt (aus [6])

5.1.4 Multer

Multer ist eine Middleware für Node.js, die speziell für den Umgang mit **Multipart/Form-data** entwickelt wurde, einem Format, das zum Herunterladen von Dateien verwendet wird. In der Webentwicklung ist es üblich, den Benutzern das Herunterladen von Dateien zu ermöglichen, seien es Bilder, Dokumente oder andere Arten von Binärdateien. Um solche Downloads in Node.js zu handhaben,

braucht man eine Lösung, die diese Art von eingehenden Daten effizient verarbeiten kann.[\[22\]](#) Multer bietet die folgenden Funktionen:

- **Dateispeicherung:** multer ermöglicht Flexibilität bei der Art und Weise, wie Dateien gespeichert werden. Dies kann im Dateisystem des Servers oder in Cloud-Speicherlösungen sein.
- **Dateimanipulation:** bevor eine Datei gespeichert wird, kann sie mit multer gefiltert, umbenannt oder auf andere Weise bearbeitet werden.
- **Validierung:** sie können Beschränkungen für Dateitypen und -größen festlegen und so sicherstellen, dass nur geeignete Dateien akzeptiert werden.



Abb. 5.4: multer Übersicht (aus [21])

5.2 Implementierung der wichtigsten Funktionen

5.2.1 Konfiguration des Express-Servers und Erstellen der Datenbank

Listing 5.1: Konfiguration des Express-Servers und Erstellen der Datenbank

```
1 const express = require('express');
2 const bodyParser = require('body-parser');
3 const bcrypt = require('bcrypt');
4 const jwt = require('jsonwebtoken');
5 const cors = require('cors');
6 const multer = require('multer');
7 var mysql = require('mysql');
8
9 const app = express();
10 const port = 3000;
11 app.use(bodyParser.json());
12 app.use(cors());
13 var con = mysql.createConnection({
14   host: "localhost",
15   user: "root",
16   password: "*****",
17   database: 'jobSearch'
18 });
19
```

```

20 con.query(`CREATE DATABASE IF NOT EXISTS JobSearch;`, (err,
21   result) => {
22     if (err) throw err;
23
24     con.changeUser({ database : 'JobSearch' }, function(err) {
25       if (err) throw err;
26
27       con.query(`CREATE TABLE IF NOT EXISTS Employer (
28         id INT AUTO_INCREMENT PRIMARY KEY,
29         namee VARCHAR(255),
30         adress VARCHAR(255),
31         email VARCHAR(255),
32         telephone VARCHAR(255)
33       )`, (err, result) => {
34         if (err) throw err;
35
36         con.query(`CREATE TABLE IF NOT EXISTS JobOffer (
37           id INT AUTO_INCREMENT PRIMARY KEY,
38           employerId INT,
39           title VARCHAR(255),
40           description TEXT,
41           salary FLOAT,
42           publicationDate TIMESTAMP DEFAULT CURRENT_TIMESTAMP
43
44           location VARCHAR(255),
45           typeContract VARCHAR(255),
46           businessSector VARCHAR(255),
47           FOREIGN KEY (employerId) REFERENCES Employer(id)
48         )`, (err, result) => {
49           if (err) throw err;
50
51           con.query(`CREATE TABLE IF NOT EXISTS JobSeeker (
52             id INT AUTO_INCREMENT PRIMARY KEY,
53             namee VARCHAR(255),
54             email VARCHAR(255),
55             password VARCHAR(255),
56             cv TEXT,
57             telephone VARCHAR(255),
58             motivationLetter TEXT
59           )`, (err, result) => {
60             if (err) throw err;
61             console.log("JobSeeker table created");
62           });
63         });
64       });
65     });
66   });

```

67 }) ;

Das Express-Framework, das aufgrund seiner Flexibilität und Einfachheit eine beliebte Wahl für Node.js-Webanwendungen ist, wird im Kern der Anwendung eingesetzt. Nach dem Import von express ist die erste Aktion die Initialisierung einer neuen Instanz von Express, die die Anwendung symbolisiert

Beim Übergang zum Aspekt der Datenverarbeitung ist die Integration von bodyParser von entscheidender Bedeutung. Diese Middleware wandelt den eingehenden Inhalt von Anfragen im JSON-Format in verwertbare JavaScript-Objekte um. Mithilfe von `app.use(bodyParser.json())`; wird jede Anfrage, die den Server passiert, analysiert und ihr Inhalt in einer handhabbaren Form zur Verfügung gestellt.

Das Modul **bcrypt** wird normalerweise für das Hashing von Passwörtern verwendet, um deren Sicherheit zu gewährleisten. **jsonwebtoken**, oder **jwt**, spielt eine Schlüsselrolle bei der Verwaltung von Authentifizierungs-Tokens, um sichere Sitzungen für die Benutzer aufrechtzuerhalten. **multer** wiederum ist eine gute Wahl, um das Herunterladen von Dateien zu verwalten. Und schließlich dient **mysql** dazu, eine Verbindung zu einer MySQL-Datenbank herzustellen, was die Interaktion zwischen unserer Anwendung und unseren persistenten Daten erleichtert.

Bei der Entwicklung dieser Anwendung für die Stellensuche wurde eine Datenbank mit mehreren Haupttabellen eingerichtet, darunter *Employer*, *JobOffer* und *JobSeeker*. Jede dieser Tabellen spielt eine spezifische Rolle und ermöglicht die Speicherung von Informationen über Arbeitgeber, Stellenangebote bzw. Bewerber. Eine bemerkenswerte Besonderheit ist die Verbindung zwischen den Tabellen *Employer* und *JobOffer*. Die Tabelle *JobOffer* enthält einen Fremdschlüssel, der auf die ID des Arbeitgebers verweist.

Die Architektur basiert auf dem Paradigma der relationalen Datenbank, das in modernen Anwendungen häufig verwendet wird. Im Zentrum dieses Ansatzes steht das Konzept des Schlüssels - sei es ein Primär- oder ein Fremdschlüssel. Jede Tabelle in dieser Struktur hat ein eindeutiges Feld, das als Identifikator dient und als Primärschlüssel bezeichnet wird. In diesem Fall ist es eine selbsterstellte ID. Darüber hinaus werden Fremdschlüssel verwendet, um Verbindungen zwischen den Tabellen herzustellen, wie das Feld *employerId*, das Stellenangebote mit einem bestimmten Arbeitgeber verknüpft.

Die Tabelle *Employer* ist für die Speicherung von Informationen über Arbeitgeber vorgesehen und umfasst Felder wie *Name*, *Adresse*, *E-Mail* und *Telefon*. Die Tabelle *JobOffer* speichert Stellenangebote. Sie weist eine interessante Besonderheit auf: Das Feld *publicationDate* ist so gestaltet, dass es automatisch das aktuelle Datum und die aktuelle Uhrzeit erfasst, wenn eine neue Stelle hinzugefügt wird. Die Tabelle *JobSeeker* schließlich ist den Bewerbern gewidmet und umfasst sowohl persönliche als auch berufliche Informationen.

5.2.2 Detaillierte Vorgehensweise für die Erstellung eines Benutzerkontos auf der Plattform

Die Erstellung eines Benutzerkontos ist ein entscheidender Schritt für jede Webanwendung, da sie den ersten Kontakt zwischen dem Benutzer und der Platt-

form herstellt. In dieser Anwendung wurde diese Funktion durch ein sorgfältiges Design und eine durchdachte Ergonomie hervorgehoben.

Listing 5.2: Front-End Code des Formulars zur Erstellung eines Benutzerkontos

```

1 <h2>Account Creation</h2>
2 <form (ngSubmit)="submit()" #loginForm="ngForm">
3   <span>Name</span>
4   <input [(ngModel)]="jobSeeker.namee" name="namee"
      placeholder="Name" required #nameField="ngModel">
5   <span class="error" style="color: red" *ngIf="nameField.
      errors?.['required'] && nameField.touched">Name is
      required</span>
6   <span>E-mail</span>
7   <input [(ngModel)]="jobSeeker.email" name="email"
      placeholder="Email" required #emailField="ngModel">
8   <span class="error" style="color: red" *ngIf="emailField.
      errors?.['required'] && emailField.touched">Email is
      required</span>
9   <span>Password</span>
10  <input [(ngModel)]="jobSeeker.password" name="password"
      placeholder="password" required minlength="8" #
      passwordField="ngModel">
11  <span class="error" style="color: red" *ngIf="passwordField.
      errors?.['required'] && passwordField.touched">password
      is required</span>
12  <span class="error" style="color: red" *ngIf="passwordField.
      errors?.['minlength'] && passwordField.touched">
      Password must be at least 8 characters</span>
13  <span>Type of Account</span>
14  <select [(ngModel)]="jobSeeker.type_of_account" name="
      type_of_account" class="sel" required #
      typeofaccountField="ngModel">
15    <option value="jobseeker">JobSeeker</option>
16    <option value="employe">Employe</option>
17  </select>
18  <span class="error" style="color: red" *ngIf="
      typeofaccountField.errors?.['required'] &&
      typeofaccountField.touched">Type of account is required
      </span>
19  <button type="submit" [disabled]="!loginForm.valid">Create
      an Account</button>
20 </form>
```

The screenshot shows a dark-themed 'ACCOUNT CREATION' form. At the top, it says 'ACCOUNT CREATION'. Below that is a white input field labeled 'Name' with the placeholder 'Name'. A red validation message 'Name is required' is displayed above the field. Next is an 'E-mail' field with placeholder 'Email', followed by a red validation message 'Email is required'. Then is a 'Password' field with placeholder 'password'. Below these fields is a dropdown menu labeled 'Type of Account'. At the bottom is a dark blue button labeled 'create an account'.

Abb. 5.5: Formular zur Kontoerstellung

Der Prozess der Kontoerstellung wurde so gestaltet, dass er sowohl funktional als auch ästhetisch ansprechend ist. Er dient als solides Fundament für eine erfolgreiche Interaktion zwischen dem Nutzer und unserer Plattform, sei es als Arbeitgeber oder als Arbeitssuchender.

1. Aufbau des Formulars

- **Wesentliche Felder:** Das Formular wurde entworfen, um grundlegende, aber wichtige Informationen zu sammeln: *Name, E-Mail und Passwort*. Diese Informationen sind wichtig, um eine eindeutige und sichere Identifizierung jedes Nutzers zu gewährleisten.
- **Auswahl des Kontotyps:** Das Besondere an unserer Anwendung ist, dass sie sowohl Arbeitgebern als auch Arbeitssuchenden dienen kann. Das Hinzufügen einer Option zur Auswahl des Kontotyps gewährleistet eine personalisierte Erfahrung für jede Nutzerkategorie.

2. Funktionen für die Validierung

- **Instant Feedback:** Um zu verhindern, dass etwas vergessen wird, werden neben nicht ausgefüllten Feldern rote Fehlermeldungen angezeigt, die den Benutzer dazu anleiten, das Formular korrekt auszufüllen. Dieser Mechanismus stellt nicht nur die Qualität der gesammelten Daten sicher, sondern erleichtert auch das Nutzererlebnis, indem er den Nutzer genau darauf hinweist, wo der Fehler liegt.

Sobald der Nutzer das Formular ausgefüllt hat, werden die Daten über eine API an das Backend weitergeleitet, um mit der Erstellung des Kontos fortzufahren.

Listing 5.3: API für die Methode zur Erstellung eines Kontos

```

1 createAccount(jobseeker: any): Observable<any> {
2   return this.http.post('http://localhost:3000/api/account-
3     creating', jobseeker);

```

Listing 5.4: implementation of an API route for creating user accounts in our application

```

1 app.post('/api/account-creating', async (req, res) => {
2     const user = req.body;
3     const salt = await bcrypt.genSalt(10);
4     user.password = await bcrypt.hash(user.password, salt);
5     let tableName;
6     if(user.type_of_account === 'jobseeker') {
7         tableName = 'JobSeeker';
8     } else if(user.type_of_account === 'employe') {
9         tableName = 'Employer';
10    } else {
11        res.status(400).json({ error: 'Invalid account type' });
12    }
13    const query = `INSERT INTO ${tableName} SET ?`;
14    con.query(query, user, (err, result) => {
15        if (err) {
16            res.status(500).json({ error: 'Error when creating an account' });
17        }
18        res.json({ success: true });
19    });
20 });
21 });
22 });

```

Dieses Stück Code in Listing 5.4 veranschaulicht die Implementierung einer API-Route für die Erstellung von Benutzerkonten in der Anwendung. Hier die wichtigsten Schritte:

- **Empfang der Daten:** wenn ein Nutzer das Formular zur Erstellung eines Kontos abschickt, werden die Daten über eine POST-Anfrage an das Backend gesendet. Die Daten werden dann extrahiert.
- **Sicherung des Passworts:** bevor das Passwort in der Datenbank gespeichert wird, muss es unbedingt gesichert werden. Hier verwenden wir die bcrypt-Bibliothek, um das Passwort zu zerhacken.
- **Auswahl der Speichertabelle:** unser System verwaltet zwei Arten von Konten: *jobseeker* (*Stellensuchender*) und *employe* (*Arbeitgeber*). Je nachdem, welche Art von Konto der Benutzer gewählt hat, bestimmen wir die entsprechende Tabelle, in die die Daten eingefügt werden sollen.
- **Einfügen in die Datenbank:** eine SQL-Abfrage wird vorbereitet, um die Daten in die entsprechende Tabelle einzufügen. Wenn die Abfrage aus irgend einem Grund fehlschlägt, wird eine Fehlerantwort zurückgegeben. Wenn die Abfrage erfolgreich war, wird eine Erfolgsmeldung an den Kunden zurückgeschickt, die bestätigt, dass das Konto erstellt wurde.

Dieser Ansatz gewährleistet sowohl die Sicherheit der Benutzerdaten als auch eine saubere und modulare Softwarearchitektur.

5.2.3 Implementation der Anmeldung bei einem bestehenden Benutzerkonto

Die Möglichkeit für die Nutzer, sich in ihr Konto einzuloggen, ist für jede interaktive Webanwendung von entscheidender Bedeutung. Sie ermöglicht den Nutzern nicht nur den Zugriff auf personalisierte Informationen, sondern stellt auch sicher, dass die Daten sicher sind und nur von autorisierten Parteien eingesehen werden können.

Listing 5.5: Code für die Anmeldung bei einem Benutzerkonto

```
1 <h2>Welcome to our website</h2>
2 <form (ngSubmit)="submit()" #loginForm="ngForm">
3   <span>E-mail</span>
4   <input [(ngModel)]="credentials.email" name="email"
      placeholder="Email" required #emailField="ngModel">
5   <span class="error" style="color: red" *ngIf="emailField.
      errors?.['required'] && emailField.touched">Email is
      required</span>
6
7   <span>Password</span>
8   <input [(ngModel)]="credentials.password" name="password"
      placeholder="Mot de passe" type="password" required
      minlength="8" #passwordField="ngModel">
9   <span class="error" style="color: red" *ngIf="passwordField.
      errors?.['required'] && passwordField.touched">Password
      is required</span>
10  <span class="error" style="color: red" *ngIf="passwordField.
      errors?.['minlength'] && passwordField.touched">
      Password must be at least 8 characters</span>
11
12  <button type="submit" [disabled]="!loginForm.valid || isSubmitted">log in</button>
13 </form>
14 <a routerLink="/account-creation">Create an account</a>
```

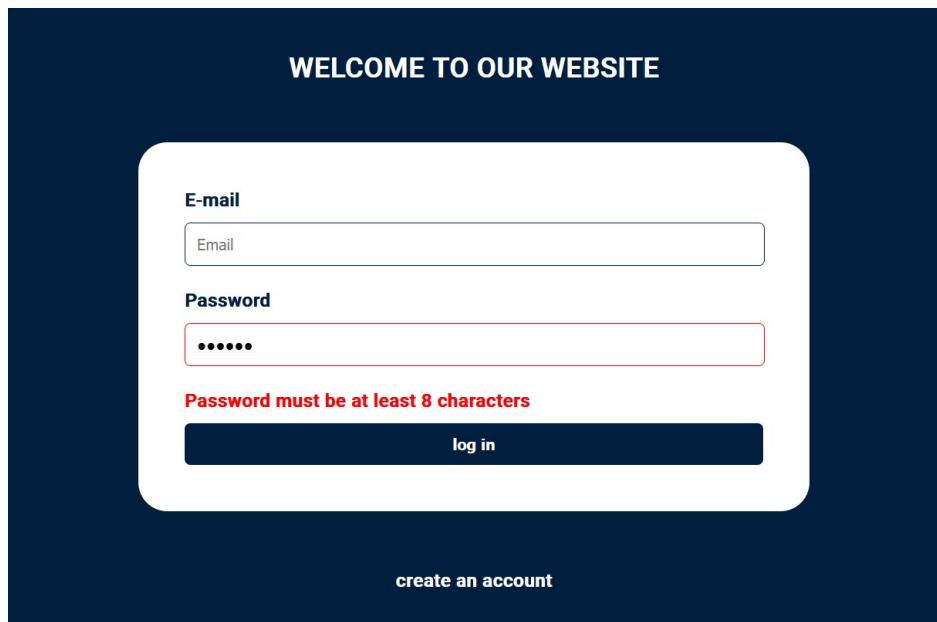


Abb. 5.6: Anmeldung bei einem bestehenden Benutzerkonto

Bei der Gestaltung des Anmeldeformulars in Listing 5.5 und Abb. 5.6 wurde besonders auf den visuellen Aspekt geachtet. Denn die Wahl der Farben ist für ein optimales Nutzererlebnis von entscheidender Bedeutung. Daher wird der Nutzer mit einem dunkelblauen Hintergrund, der einen eleganten Kontrast zum weißen Formular bildet, auf natürliche Weise geführt und seine Konzentration auf die wesentlichen Felder gelenkt. Um diese Erfahrung noch zu bereichern, wurde außerdem eine interaktive Nuance hinzugefügt: Wenn der Nutzer auf die Schaltfläche *log in* klickt, beobachtet er einen Farbwechsel zu Rosa. Dieses kleine, keineswegs unbedeutende Detail liefert ein sofortiges visuelles Feedback, das dem Nutzer signalisiert, dass seine Aktion beachtet wurde.

Andererseits wurden neben dem ästhetischen Aspekt auch Schlüsselemente integriert, die sowohl die Sicherheit als auch die Benutzererfahrung verbessern. So ist es zum Beispiel zwingend notwendig, dass jedes Feld des Formulars ausgefüllt wird, um eine korrekte Authentifizierung zu gewährleisten. Wenn ein Benutzer ein Feld nicht ausfüllt, erscheint sofort eine rote Fehlermeldung, die den Benutzer darauf hinweist. Um die Informationen der Nutzer noch sicherer zu machen, wurde außerdem ein Kriterium für die Zusammensetzung des Passworts festgelegt. Indem wir ein Minimum von 8 Zeichen vorschreiben, minimieren wir nicht nur die Risiken, die mit simplen Passwörtern verbunden sind, sondern es dient auch dazu, die Wichtigkeit von robusten Passwörtern in Erinnerung zu rufen.

Listing 5.6: API Route für die Verbindung mit einem Benutzerkonto.

```

1 app.post('/api/authentication', (req, res) => {
2   const { email, password } = req.body;
3   con.query('SELECT * FROM JobSeeker WHERE email = ?', [email], async (error, jobSeekerResults) => {
4     if (error) {
5       return res.status(500).json({ error: 'Database error' });
6     }
7     if (jobSeekerResults.length > 0) {

```

```

8      const user = jobSeekerResults[0];
9      const isMatch = await bcrypt.compare(password,
10         user.password);
11
12      if (!isMatch) {
13          return res.status(400).json({ error: 'Incorrect_email_address_or_password' });
14      }
15
16      const token = jwt.sign({ id: user.id, type: 'JobSeeker' }, 'tony', { expiresIn: '24h' });
17
18      return res.json({ token, userType: user.type_of_account, userID: user.id, userName: user.namee });
19
20  con.query('SELECT * FROM Employer WHERE email = ? ', [
21    email], async (error, employerResults) => {
22      if (error) {
23          return res.status(500).json({ error: 'Database_error' });
24      }
25
26      if (employerResults.length === 0) {
27          return res.status(400).json({ error: 'Incorrect_email_address_or_password' });
28      }
29
30      const user = employerResults[0];
31      const isMatch = await bcrypt.compare(password,
32         user.password);
33
34      if (!isMatch) {app.post('/api/
35        uploadTitleOfStayFiles', (req, res)=>{
36          console.log('test1reusssi');
37        })
38          return res.status(400).json({ error: 'Incorrect_email_address_or_password' });
39      }
40
41      const token = jwt.sign({ id: user.id, type: 'Employer' }, 'tony', { expiresIn: '24h' });
42
43      res.json({ token, userType: user.type_of_account,
44         userID: user.id, userName: user.namee });
45  });

```

Die Authentifizierungsroute `/api/authentifizierung` ist dafür gedacht, Anmeldeversuche von Benutzern zu verarbeiten, unabhängig davon, ob es sich um Arbeitssuchende *JobSeeker* oder Arbeitgeber *Employe* handelt.

Wenn ein Anmeldeantrag eingeht, versucht das System zunächst, die E-Mail-Adresse des Benutzers in der JobSeeker-Tabelle zu finden. Wird ein Treffer gefunden, wird das vom Benutzer angegebene Passwort mithilfe der bcrypt-Bibliothek mit dem in der Datenbank gespeicherten Hash-Passwort verglichen. Dadurch wird sichergestellt, dass selbst wenn die Datenbank kompromittiert wird, die tatsächlichen Passwörter der Benutzer geheim bleiben.

Wenn die Passwortprüfung erfolgreich ist, wird für den Benutzer ein JWT-Token erzeugt. Dieses Token wird verwendet, um den Benutzer angemeldet zu halten und seine zukünftigen Interaktionen mit der Anwendung zu sichern.

Für den Fall, dass die E-Mail nicht in der JobSeeker-Tabelle gefunden wird, wiederholt das System denselben Vorgang für die Arbeitgeber-Tabelle und stellt so sicher, dass sich Arbeitgeber auch mit ihren eigenen Anmeldeinformationen anmelden können.

5.2.4 Dashboard

Die Architektur des Dashboards wurde akribisch entworfen und berücksichtigt die unterschiedlichen Bedürfnisse der beiden Hauptnutzer der App: *Jobsucher* und *Arbeitgeber*. Für die Jobsuchenden war es vorrangig, das Dashboard so einfach wie möglich zu gestalten, um die Navigation zu den Stellenangeboten, die Bestätigung der gesendeten Bewerbungen, das Bearbeiten des Profils und das Abmelden zu erleichtern. Aus diesem Grund wurde jede Funktion sinnvoll im Header platziert, um einen direkten und schnellen Zugriff zu gewährleisten.

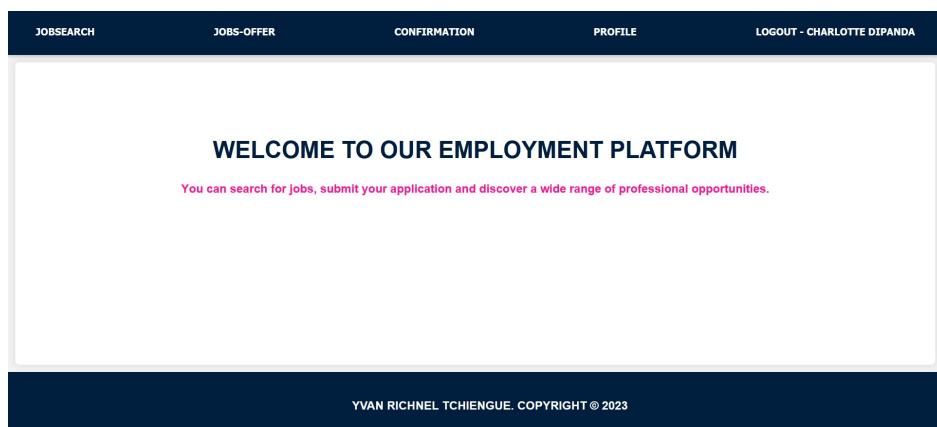


Abb. 5.7: Boardingboard für den Arbeitnehmer

Auf der anderen Seite weist die Perspektive der Arbeitgeber eine Nuance auf. Ihre Benutzeroberfläche ist eher darauf ausgerichtet, die eingereichten Stellenangebote zu verwalten, Stellenangebote zu veröffentlichen, die eingegangenen Bewerbungen anzusehen und sich natürlich abzumelden. Dieser Kontrast zwischen den beiden Dashboards verdeutlicht die Verpflichtung, die Nutzererfahrung an die spezifischen Bedürfnisse der Nutzer anzupassen.

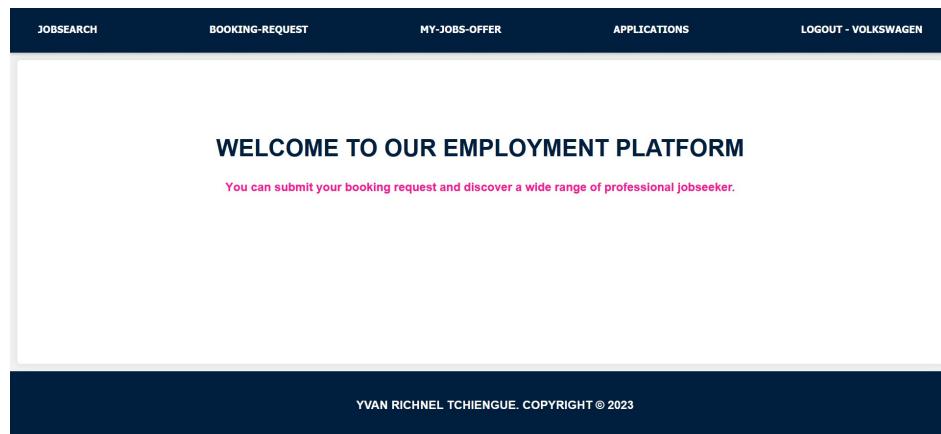


Abb. 5.8: Boardingboard für den Arbeitgeber

Besonders hervorzuheben ist, dass sich das Dashboard je nach Kontotyp anpasst und so sicherstellt, dass die Nutzer nur die Informationen sehen, die für ihre Rolle relevant sind. Ein Arbeitgeber muss z. B. keine Stellenangebote sehen, da seine Rolle darin besteht, Mitarbeiter einzustellen und nicht nach Stellen zu suchen.

Übergehend in den ästhetischen Bereich, zeichnet sich das Design der Anwendung durch Eleganz und Schlichtheit aus. Die Kombination aus einem dunkelblauen Header und Footer, die weißen Mat-Cards gegenübergestellt werden, schafft einen auffälligen visuellen Kontrast. Diese Farbpalette wurde sorgfältig ausgewählt, um eine optimale Lesbarkeit zu gewährleisten und gleichzeitig einer Informationsüberlastung vorzubeugen. Infolgedessen stellen wir sicher, dass die Navigation unabhängig von der Reise des Nutzers flüssig und intuitiv bleibt.

5.2.5 Jobsuche

Jede seriöse Personalvermittlungsplattform muss eine effiziente und intuitive Jobsuche anbieten. In diesem Sinne wurde in dieser Webanwendung ein Suchmechanismus entwickelt, der diese Kriterien erfüllt.

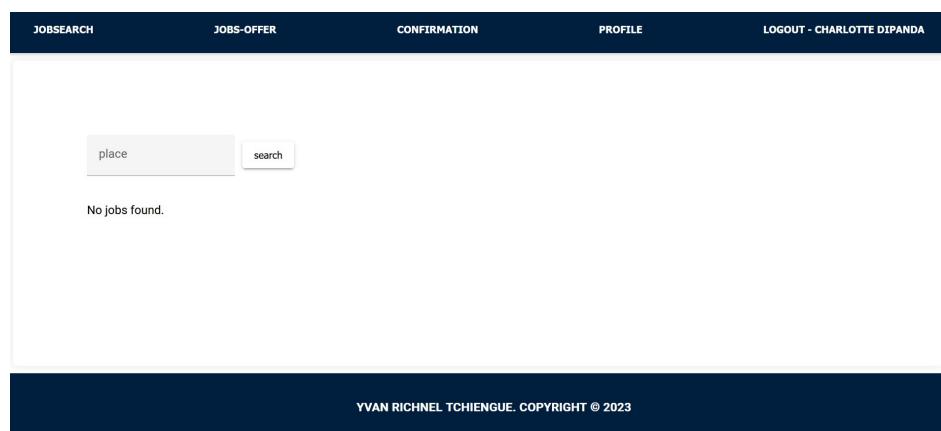


Abb. 5.9: Anzeige von verfügbaren Stellenangeboten

- **Mechanismus der Suche:** das Herzstück dieser Funktion ist der Suchprozess, der vor allem benutzerzentriert ist. Sobald die Schaltfläche *search* gedrückt wird, wird sofort eine Verbindung zur Datenbank hergestellt, die alle verfügbaren Stellenangebote anzeigt.

- **Hervorheben von Angeboten:** ebenso entscheidend ist die Hervorhebung der Angebote. Zu diesem Zweck wurde die Wahl auf Mat-Cards getroffen. Diese Wahl ist nicht unbedeutend. Erstens, was die Lesbarkeit angeht, bieten die Mat-Cards eine unübertroffene Klarheit, indem sie jedes Angebot isolieren. Zweitens sorgt die Verwendung von Angular-Material in Bezug auf Ästhetik und Kohärenz für eine visuelle Harmonie, die ein optimales Nutzererlebnis garantiert. Darüber hinaus ist ihr interaktives Potenzial ein unbestreitbarer Vorteil und bietet direkte Handlungsmöglichkeiten wie *apply now*

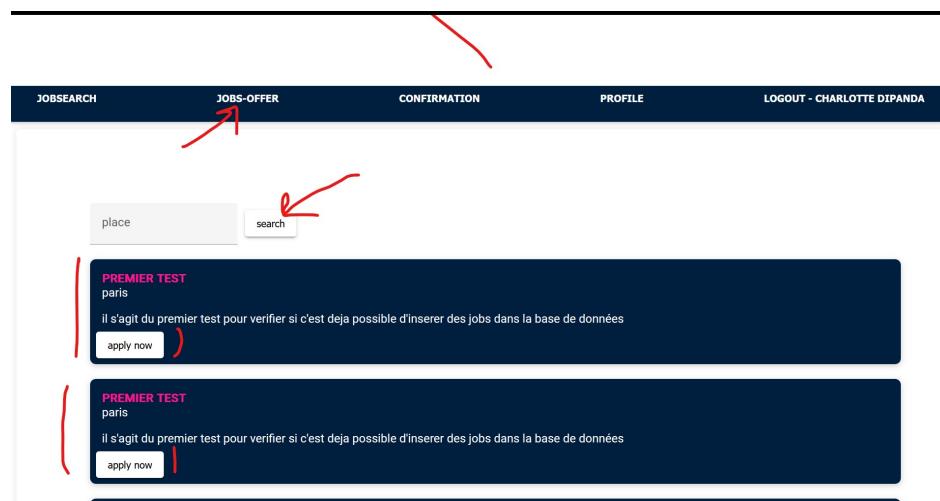


Abb. 5.10: Anzeige von verfügbaren Stellenangeboten

Listing 5.7: Anzeige aller in der Datenbank verfügbaren Jobs

```

1 app.get('/api/jobsOffer', (req, res) => {
2
3   con.query('USE `JobSearch`', (err) => {
4     if (err) throw err;
5
6     let sqlSelect = `SELECT * FROM joboffer`;
7     con.query(sqlSelect, (err, results) => {
8       if (err) throw err;
9       res.json(results);
10    });
11  });
12});
13}

```

Die Funktion in [listing 5.7](#) ist eine API-Route, die, wenn sie aufgerufen wird, die Gesamtheit der in der Datenbank verfügbaren Stellenangebote zurückgibt. Diese Funktion ist von entscheidender Bedeutung, da sie den ersten Schritt der Interaktion zwischen dem Nutzer und den Daten der Plattform darstellt. Der erste Schritt dieser Funktion besteht darin, die richtige Datenbank auszuwählen. Dies ist entscheidend, um sicherzustellen, dass alle nachfolgenden Abfragen auf der richtigen Datenbank durchgeführt werden. Nachdem die Datenbank ausgewählt wurde, führt die Funktion die SQL-Abfrage **SELECT * FROM joboffer** aus. Mit dieser Abfrage werden

alle Stellenangebote, die sich in der Tabelle *joboffer* der Datenbank befinden, ausgewählt und abgerufen. Unbedingt zu beachten ist das Vorhandensein von Fehlerbehandlungen mit den Klauseln **if (err) throw err**. Diese Mechanismen stellen sicher, dass im Falle eines Problems bei der Ausführung von Datenbankabfragen der Fehler identifiziert und gemeldet wird, was die Wartung und Fehlerbehebung erheblich erleichtert. Schließlich werden die Stellenangebote, nachdem sie abgerufen wurden, im JSON-Format mit dem Befehl **res.json(results)** an den Kunden zurückgegeben. Das JSON-Format ist universell anerkannt und ermöglicht eine einfache Integration in viele Front-End-Systeme, wodurch eine hohe Kompatibilität und Flexibilität gewährleistet wird.

5.2.6 Speicherung von Dateien

Eine der wichtigsten Funktionen dieser Webanwendung, wie in Abb. 5.11 dargestellt, ist die Möglichkeit für die Nutzer, entscheidende Dokumente wie ihren *Personalausweis*, *Lebenslauf*, *Aufenthaltstitel* und *Arbeitsgenehmigung* hochzuladen.

The screenshot shows a user interface for document uploads. At the top, there are five navigation links: 'JOBSEARCH', 'JOBS-OFFER', 'CONFIRMATION', 'PROFILE', and 'LOGOUT - CESAR BOUBEKI'. Below these, there is a large form area containing four sections, each with a 'Browse...' button and a 'send' button:

- IDENTITY CARD:** 'Browse...' button, 'No file selected.' message, 'send' button.
- CV:** 'Browse...' button, 'No file selected.' message, 'send' button.
- TITLE OF STAY:** 'Browse...' button, 'No file selected.' message, 'send' button.
- WORK PERMIT:** 'Browse...' button, 'No file selected.' message, 'send' button.

Abb. 5.11: Implementierung der Sicherung wichtiger Dokumente in der Webanwendung

Listing 5.8: Anzeige aller in der Datenbank verfügbaren Jobs

```

1 <app-header></app-header>
2 <div class="form-container">
3   <form (submit)="onFilesIdentityCardUpload()">
4     <div class="form-group">
5       <span>Identity Card</span>
6       <input type="file" (change)="onIdentityCardSelected(
7         $event)" accept=".pdf">
8     </div>
9     <button type="submit" [disabled]="isApplied">send</button>
10    </form>
11    <form (submit)="onFilesMotivationLetterUpload()">
12      <div class="form-group">
13        <span>CV</span>

```

```

13   <input type="file" (change)="onMotivationLetterSelected($event)" accept=".pdf">
14 </div>
15 <button type="submit" [disabled]="isApplied1">send</
    button>
16 </form>
17 <form (submit)="onFilesTitleOfStayUpload()">
18   <div class="form-group">
19     <span>Title of Stay</span>
20     <input type="file" (change)="onTitleOfStaySelected(
        $event)" accept=".pdf">
21   </div>
22   <button type="submit" [disabled]="isApplied2">send</
    button>
23 </form>
24 <form (submit)="onFilesWorkPermitUpload()">
25   <div class="form-group">
26     <span>Work Permit</span>
27     <input type="file" (change)="onWorkPermitSelected(
        $event)" accept=".pdf">
28   </div>
29   <button type="submit" [disabled]="isApplied3">send</
    button>
30 </form>
31 </div>
32 <footer class="footer">
33   <p>Yvan Richnel Tchiengue. Copyright © 2023</p>
34 </footer>

```

Der Code in Listing 5.8 beginnt mit dem Aufruf einer <app-header>-Komponente, die den Kopf der Anwendung darstellt und für ästhetische und funktionale Konsistenz über alle Seiten der Anwendung hinweg sorgt.

Der Hauptteil des Codes in Listing 5.8 ist in einer CSS-Klasse *form-container* gekapselt, die dazu dient, den darin enthaltenen Formularen Stil und Struktur zu verleihen.

Jedes Formular ist darauf ausgelegt, einen bestimmten Dokumententyp hochzuladen. Durch die Verwendung des Attributs *accept=.pdf* bei Eingabefeldern für Dateitypen wird sichergestellt, dass nur Dateien im PDF-Format ausgewählt werden können, wodurch eine gewisse Einheitlichkeit beim Upload gewährleistet wird.

Wenn eine Datei ausgewählt wird, wird ein Ereignis (change) ausgelöst, das eine spezielle Funktion für diesen Dokumententyp aufruft, z. B. *onIdentityCardSelected(event)*. Dies zeigt eine gut durchdachte Architektur, in der jede Benutzeraktion separat behandelt wird, was die Wartung und Skalierbarkeit des Codes erleichtert.

Die Schaltflächen zum Einreichen werden durch die Variable *isApplied* bedingt, was bedeutet, dass der Benutzer eine Datei nur einmal einreichen kann und danach die Schaltfläche deaktiviert wird.

Listing 5.9: Verwaltung des Uploads von Identitätsdokumenten

```

1 const storageICard = multer.diskStorage({
2     destination: (req, file, cb) => {
3         cb(null, 'C:/Users/yvant/Desktop/Cours/Bachelorarbeit'
4             '/Bachelorarbeit-JobSearch/src/assets');
5     },
6     filename: (req, file, cb) => {
7         const bearerHeaderMLetter = req.headers['
8             authorization'];
9         const bearerTokenMLetter = JSON.parse(
10             bearerHeaderMLetter.replace("Bearer", ""));
11         const userIdMLetter = bearerTokenMLetter.userID;
12         cb(null, `identitycard${userIdMLetter}.pdf`);
13     }
14 });
15
16 const uploadICard = multer({storage: storageICard});
17
18 app.post('/api/uploadIdentityCardFiles', uploadICard.single('
19     identityCard'), async (req, res) => {
20     if (!req.file) {
21         return res.status(400).send('No file uploaded');
22     }
23     const bearerHeaderICard = req.headers['authorization'];
24     const bearerTokenICard = JSON.parse(bearerHeaderICard.
25         replace("Bearer", ""));
26     const userIdICard = bearerTokenICard.userID;
27     const userTypeICard = bearerTokenICard.userType;
28     const jwtTokenICard = bearerTokenICard.token;
29
30     const filePathICard = `C:/Users/yvant/Desktop/Cours/
31         Bachelorarbeit/Bachelorarbeit-JobSearch/src/assets/
32         identitycard${userIdICard}`;
33
34     con.query('USE JobSearch', (err, result) => {
35         if (err) throw err;
36
37         let sqlUpdate = `UPDATE ${userTypeICard} SET
38             identitycard = ? WHERE id = ?`;
39         con.query(sqlUpdate, [filePathICard, userIdICard], (err,
40             result) => {
41             if (err) throw err;
42         });
43     });
44 });

```

Der Code in Listing 5.9 verwendet multer, eine Middleware für die Verarbeitung von *Multipart/Form-Data*, die hauptsächlich für das Hochladen von Dateien verwendet wird. Die für multer angegebene Konfiguration bestimmt, wo und wie die hoch-

geladenen Dateien gespeichert werden.

- **Zielort:** der genaue Pfad, in dem die Datei gespeichert wird, wird festgelegt. Dadurch wird sichergestellt, dass alle Identitätsdateien zentral in einem bestimmten Verzeichnis gespeichert werden, um einen einfachen Zugriff und eine einfache Verwaltung zu ermöglichen.
- **Dateiname:** die Benennungsfunktion ist so konzipiert, dass die Dateinamen unter Verwendung der Benutzer-ID eindeutig sind. Dadurch werden mögliche Namenskonflikte vermieden und die Datei kann leicht einem bestimmten Benutzer zugeordnet werden.
- **Entschlüsselung und Verarbeitung des JWT-Tokens:** die Anwendung verwendet JWT-Token, um Informationen über den Benutzer zu authentifizieren und zu übermitteln. Dieses Token wird zunächst entschlüsselt, um Informationen wie die Benutzer-ID, den Benutzertyp und das Token selbst zu extrahieren. Dieser Prozess stellt sicher, dass die Datei mit dem richtigen Benutzer verknüpft wird und dass nur authentifizierte Benutzer Dateien hochladen können.

5.2.7 Ausfüllen des Formulars und Veröffentlichung einer Stellensuche

Abb. 5.12: Ausfüllen des Formulars und Veröffentlichung einer Stellensuche

Listing 5.10: Umsetzung des Systems zur Veröffentlichung von Stellenangeboten

```

1 app.post('/api/booking-request', (req, res) => {
2
3   const bearerHeader = req.headers['authorization'];
4   const bearerToken = bearerHeader.split(' ')[1];
5   console.log(bearerToken);
6   const bearerObject = JSON.parse(bearerToken);
7   const userId = bearerObject.userID;
8   const userType = bearerObject.userType;
9   const jwtToken = bearerObject.token;
10  const title = req.body.title;
11  const description = req.body.description;

```

```

12  const location = req.body.location;
13
14  con.query('USE `JobSearch`', (err) => {
15      if (err) throw err;
16
17      let sqlInsert = `INSERT INTO joboffer (employerId,
18          title, description, location) VALUES (?, ?, ?, ?)`
19          `;
20      con.query(sqlInsert, [userId, title, description,
21          location], (err, result) => {
22          if (err) throw err;
23      });
24  });

```

In Listing 5.10 empfängt der Server eine POST-Anfrage an den Endpunkt `/api/booking-request`. Der Code extrahiert dann die wichtigsten Details der Stellenanzeige aus den Daten der Anfrage, darunter *Titel*, *Beschreibung* und *Standort*. Diese Details sind für eine vollständige Stellenanzeige entscheidend und helfen potenziellen Bewerbern, die Anforderungen und Erwartungen an die Stelle zu verstehen. Eine SQL INSERT-Abfrage wird vorbereitet, um das neue Stellenangebot in die Datenbank aufzunehmen. Die Abfrage wird mit den Details des Stellenangebots und den Informationen des Arbeitgebers ausgeführt, wodurch sichergestellt wird, dass jedes Stellenangebot korrekt mit einem bestimmten Arbeitgeber verknüpft ist.

5.2.8 Einreichen der Bewerbung auf ein Stellenangebot und deren Überprüfung durch den Arbeitgeber

Mit einem einzigen Klick auf die Schaltfläche *apply now*, wie in Abb. 5.13 zu sehen, wird die Bewerbung des Arbeitsuchenden an den entsprechenden Arbeitgeber gesendet. Diese Einfachheit verringert die Reibung für die Nutzer und erhöht somit die Anzahl der potenziellen Bewerbungen für eine Stelle. Hinter dieser scheinbaren Einfachheit verbirgt sich ein robuster Prozess, der die Sicherheit der Bewerberdaten gewährleistet. Wenn eine Bewerbung eingereicht wird, stellt die Anwendung sicher, dass alle sensiblen Informationen mit äußerster Sorgfalt behandelt werden und eine unbefugte Offenlegung verhindert wird. Nach dem Absenden einer Bewerbung ist es von entscheidender Bedeutung, dem Nutzer eine sofortige Rückmeldung zu geben, die bestätigt, dass seine Aktion berücksichtigt wurde. In unserem System wird die Schaltfläche *apply now* deaktiviert, nachdem die Bewerbung abgeschickt wurde. Dies erfüllt zwei wesentliche Funktionen:

- **Bestätigung der Bewerbung:** das Deaktivieren der Schaltfläche gibt einen visuellen Hinweis darauf, dass die Bewerbung korrekt abgeschickt wurde, und beruhigt den Bewerber, dass seine Bewerbung bearbeitet wird.
- **Vermeidung von Duplikaten:** durch das Deaktivieren der Schaltfläche nach dem Absenden verhindert die Anwendung, dass Nutzer versehentlich dieselbe Bewerbung mehrmals abschicken, was ihrem Ruf bei Arbeitgebern schaden und die Datenbank unnötig belasten könnte.

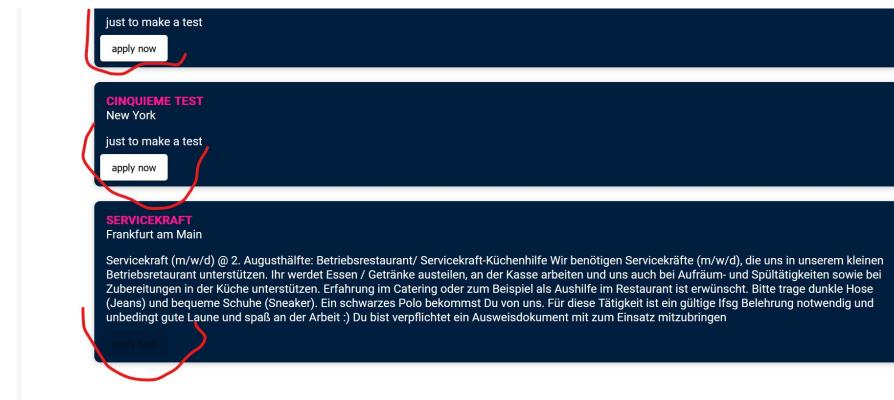


Abb. 5.13: Einreichen der Bewerbung

Listing 5.11: Methode zum Einreichen der Bewerbung

```

1 sendCandidature(offre: any): void {
2   offre.isApplied = true;
3   const headers = new HttpHeaders({
4     'Authorization': `Bearer ${this.authService.
5       getLocalStorage()}`})
6   this.http.post(` ${this.apiUrl}/upload-candidature`, offre
7     , {headers}).subscribe(
8     response => {
9       console.log(response);
10    },
11    error => {
12      console.error(error)
13    })
}

```

Die Funktion **sendCandidature** in Listing 5.10 nimmt als Parameter ein Objekt *offre* an, das die Stellenanzeige darstellt, auf die sich der Bewerber bewerben möchte. Die Bewerbung wird dann mithilfe einer HTTP POST-Anfrage an die Backend-API gesendet. Die Ziel-URL **/upload-bewerbung** legt nahe, dass das Backend diese Anfrage verarbeiten wird, indem es die Bewerbung in der Datenbank speichert.

Beim Absenden der Anfrage können zwei Situationen auftreten. Erstens wird im Falle einer erfolgreichen Bewerbung, wenn diese angemessen registriert wurde, die Antwort des Servers über den zugehörigen Callback verarbeitet. Wenn es bei der Übermittlung der Bewerbung zu Komplikationen kommt, wird die Anomalie sofort erkannt. Um die Ursache des Fehlers leichter nachvollziehen zu können, wird dieser in der Konsole untergebracht und dient so als Debugging-Tool.

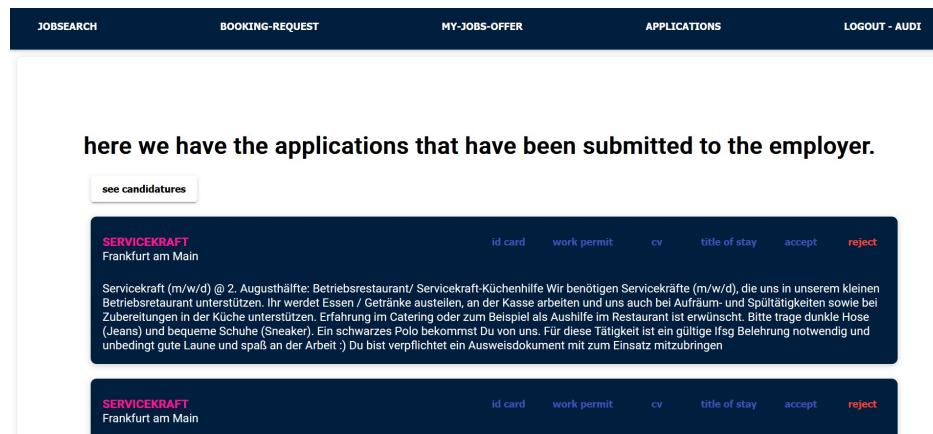


Abb. 5.14: Verwaltung von Bewerbungen

In der Anwendung , wird jede Bewerbung, wie in Abb. 5.14 dargestellt, von zwei Hauptschaltflächen begleitet: *accept* und *reject*. Diese Schaltflächen bieten eine direkte und intuitive Interaktion, die es dem Arbeitgeber ermöglicht, schnelle Entscheidungen über jede Bewerbung zu treffen. Jedes Mal, wenn ein Arbeitgeber eine Bewerbung annimmt oder ablehnt, wird diese Aktion aufgezeichnet und derjenige, der die Bewerbung abgeschickt hat, wird darüber informiert. Dies gibt nicht nur den Bewerbern ein Feedback, sondern sorgt auch für Transparenz im Einstellungsprozess.

Im Einstellungsprozess ist es von entscheidender Bedeutung, einen einfachen Zugang zu den relevanten Dokumenten der Bewerber zu haben. Die in Abb. 5.14 dargestellte Anwendung bietet eine solche Funktion, indem sie direkte Links wie *ID Card*, *Work Permit*, *CV* und *Title of Stay* anzeigt. Wenn der Arbeitgeber auf diese Links klickt, kann er die Dokumente einsehen, die die Bewerber zuvor hochgeladen haben. So wird sichergestellt, dass die benötigten Informationen immer zur Hand sind.

Listing 5.12: Abruf von mit Stellenangeboten verknüpften Bewerbungen

```

1 openCandidatures () {
2
3     const headers = new HttpHeaders ({
4         'Authorization': `Bearer ${this.authService .
5             getLocalStorage ()}` ,
6     }) ;
7
8     this . http . get<any [] >(` ${this . apiUrl} / myJobsCandidatures ` ,
9         {headers}) . subscribe (( data: any []) => {
10         this . jobs = data ;
11     })

```

Die Funktion `openCandidatures()` Listing 5.12 zielt darauf ab, eine Liste der Bewerbungen abzurufen, die mit Stellenangeboten verbunden sind, die von einem bestimmten Arbeitgeber gepostet wurden. Dadurch erhält der Arbeitgeber einen zentralen Überblick über die potenziellen Bewerber für seine Stellenangebote.

Listing 5.13: Herunterladen des Personalausweises eines Bewerbers

```
1 idCardDownload(job: any) {  
2  
3     const jobseek = job.jobSeekerId;  
4     const url = `assets/identitycard${jobseek}.pdf`;  
5  
6     this.http.get(url, { responseType: 'blob' }).subscribe(  
7         blob => {  
8             saveAs(blob, `identitycard_${jobseek}.pdf`);  
9         };  
10    };
```

Die Funktion *idCardDownload()* in Listing 5.13 wurde entwickelt, um einem Arbeitgeber zu ermöglichen, den Personalausweis eines Bewerbers herunterzuladen. Dies ist besonders nützlich, wenn die Authentizität und Legitimität eines Bewerbers überprüft wird, was bei der Einstellung von entscheidender Bedeutung ist.

6 Ergebnissbewertung

7 Zusammenfassung und Ausblick

Literatur

- [1] altexsoft. *HOW AN API WORKS*. Nov. 2022. URL: <https://www.altexsoft.com/blog/engineering/what-is-api-definition-types-specifications-documentation/> (siehe S. 22).
- [2] Angular. URL: [https://en.wikipedia.org/wiki/Angular_\(web_framework\)](https://en.wikipedia.org/wiki/Angular_(web_framework)) (siehe S. 18).
- [3] codebeautify. *JSON Full Form Is JavaScript Object Notation*. Jan. 2022. URL: <https://codebeautify.org/blog/json-full-form/> (siehe S. 22).
- [4] CodeCondo. *15 Websites Built With Express.js*. Juli 2015. URL: <https://codecondo.com/15-websites-built-with-express-js/> (siehe S. 37).
- [5] David Curry. *TypeScript Fastest Growing Programming Language, JavaScript Most Popular*. Feb. 2023. URL: <https://www.clouddatainsights.com/typescript-fastest-growing-programming-language-javascript-most-popular/> (siehe S. 20).
- [6] geeksforgeeks. *Hashing Passwords in Python with BCrypt*. Aug. 2023. URL: <https://www.geeksforgeeks.org/hashing-passwords-in-python-with-bcrypt/> (siehe S. 37).
- [7] Laurent Giret. *GitHub is Now Being Used by Over 100 Million Developers*. Jan. 2023. URL: <https://www.thurrott.com/cloud/278695/github-crosses-100-million-developers-milestone> (siehe S. 15).
- [8] Git. URL: <https://www.20i.com/blog/beginners-guide-git-all-you-need-to-know/> (siehe S. 14).
- [9] Alexis Hevia. *The magic behind npm link*. Apr. 2017. URL: <https://medium.com/@alexishevia/the-magic-behind-npm-link-d94dc3a81af> (siehe S. 17).
- [10] *How to choose a Web Hosting Provider*. Apr. 2019. URL: <https://www.exposure.com/blog/how-to-choose-a-web-hosting-provider/> (siehe S. 34).
- [11] *Html*. URL: <https://www.ionos.de/digitalguide/websites/web-entwicklung/html-tags/> (siehe S. 19).
- [12] IONOS. *Schematische Darstellung des Kommunikationsprozesses gemäß HTTP-Protokoll*. Mai 14. 2020. URL: <https://www.ionos.de/digitalguide/hosting/hosting-technik/was-ist-http/> (siehe S. 21).
- [13] *JavaScript*. URL: [=%20https://www.pngwing.com/en/free-png-ngqmz](#) (siehe S. 20).
- [14] *JavaScript History*. URL: https://www.w3schools.com/js/js_history.asp (siehe S. 19).
- [15] Javatpoint. *What is a Web Application?* Aug. 2023. URL: <https://www.javatpoint.com/web-application> (siehe S. 23, 24).

- [16] David Farle Jez Humble. *Continuous Delivery: Reliable Software Releases through Build, Test and Deployment Automation*. 1. Addison-Wesley Professional, Juli 2010. URL: <https://continuousdelivery.com/> (siehe S. 13).
- [17] Javier Calvarro Nelson Jos van der Til Martin Costello. *Integration tests in ASP.NET Core*. März 2023. URL: <https://learn.microsoft.com/en-us/aspnet/core/test/integration-tests?view=aspnetcore-7.0> (siehe S. 33).
- [18] logowik. *JetBrains Webstorm Logo*. URL: <https://logowik.com/jetbrains-webstorm-logo-vector-svg-pdf-ai-eps-cdr-free-download-11836.html> (siehe S. 16).
- [19] Oliver Moradov. *Unit testing: definition, Examples, and Critical Best Practices*. Mai 2022. URL: <https://brightsec.com/blog/unit-testing/> (siehe S. 33).
- [20] Gaurav Mukherjee. *Angular 16 is huge*. Apr. 2023. URL: <https://itnext.io/angular-16-is-huge-67288a3ff58b> (siehe S. 18).
- [21] Amir Mustafa. *How to Upload Files in Node.js using Multer*. Nov. 2021. URL: <https://javascript.plainenglish.io/uploading-files-in-node-js-using-multer-754526aa6817> (siehe S. 38).
- [22] NPM. *Multer*. Aug. 2023. URL: <https://www.npmjs.com/package/multer> (siehe S. 38).
- [23] Adekola Olawale. *What is Angular? Understanding Angular's Modular structure*. Aug. 2023. URL: <https://www.freecodecamp.org/news/front-end-javascript-development-react-angular-vue-compared/> (siehe S. 18).
- [24] OWASP. *OWASP Top 10 Web Application security Risks*. URL: <https://owasp.org/www-project-top-ten/> (siehe S. 13).
- [25] Priya Pedamkar. *Versions of Html, 6.html5*. Juli 2023. URL: <https://www.educba.com/versions-of-html/> (siehe S. 18).
- [26] proceedinteractive. *Exploring the Benefits of Mobile App Development*. Jan. 2019. URL: <https://proceedinteractive.com/exploring-the-benefits-of-mobile-app-development/> (siehe S. 25).
- [27] Rinki. *Hyper Text Markup Language*. Mai 2023. URL: <https://www.c-sharpcorner.com/article/the-importance-of-html-in-web-development/> (siehe S. 18).
- [28] Michał Sajdak. *Introduction to JWT*. Okt. 2019. URL: <https://research.securitum.com/jwt-json-web-token-security/> (siehe S. 36).
- [29] Robert Sheldon. *The history of Git*. Feb. 2023. URL: <https://www.techtarget.com/searchitoperations/definition/Git> (siehe S. 14).
- [30] Joe Silk. *THE BASICS OF BACKEND DEVELOPMENT, APIS*. Juli 2022. URL: <https://www.startechup.com/blog/back-end-development/> (siehe S. 32).
- [31] Jarosław Szutkowski. *Real-Time Communication Between Frontend And Backend*. Feb. 2023. URL: <https://dev.to/jszutkowski/real-time-two-way-communication-between-frontend-and-backend-using-sockets-5ghc> (siehe S. 34, 35).
- [32] Techopedia. *Mobile Application*. Aug. 2020. URL: <https://www.techopedia.com/definition/2953/mobile-application-mobile-app> (siehe S. 24).

- [33] Vinayak Tekade. *Getting Started with Backend Development*. März 2021. URL: <https://medium.com/coders-capsule/getting-started-with-backend-development-8ce55585e860> (siehe S. 33).
- [34] thinklogic. *What Is Web Application Security and How Does It Work?* Okt. 2021. URL: <https://www.thinklogic.com/post/what-is-web-application-security-and-how-does-it-work> (siehe S. 30).
- [35] Daragh Ó Tuama. *What is the Difference Between Web App Mobile App?* Aug. 2023. URL: <https://codeinstitute.net/de/blog/web-app-vs-mobile-app/> (siehe S. 27).
- [36] *TypeScript*. URL: <https://en.wikipedia.org/wiki/TypeScript> (siehe S. 19).
- [37] *TypeScript*. URL: <https://javascript.plainenglish.io/how-to-start-a-blank-typescript-project-1d260f7e2aa8> (siehe S. 19).
- [38] *User Experience, User Interface*. URL: <https://www.finalsites.com/blog/p-board/b/post/what-is-user-experience> (siehe S. 32).
- [39] *What is an SSL Certificate?* URL: <https://www.digicert.com/what-is-an-ssl-certificate> (siehe S. 34).
- [40] Wikipedia. *API*. Aug. 2023. URL: <https://en.wikipedia.org/wiki/API> (siehe S. 22).
- [41] Wikipedia. *Bcrypt*. Aug. 2023. URL: <https://en.wikipedia.org/wiki/Bcrypt> (siehe S. 37).
- [42] Wikipedia. *Express.js*. Aug. 2023. URL: <https://en.wikipedia.org/wiki/Express.js> (siehe S. 37).
- [43] Wikipedia. *GitHub*. Aug. 2023. URL: <https://en.wikipedia.org/wiki/GitHub> (siehe S. 15).
- [44] Wikipedia. *HTTP*. Aug. 2023. URL: <https://en.wikipedia.org/wiki/HTTP> (siehe S. 21).
- [45] Wikipedia. *JSON*. Aug. 2023. URL: <https://en.wikipedia.org/wiki/JSON> (siehe S. 22).
- [46] Wikipedia. *JWT*. Aug. 2023. URL: https://en.wikipedia.org/wiki/JSON_Web_Token (siehe S. 36).
- [47] Wikipedia. *Nodejs*. Aug. 2023. URL: <https://en.wikipedia.org/wiki/Node.js> (siehe S. 16).
- [48] Wikipedia. *npm*. Aug. 2023. URL: <https://en.wikipedia.org/wiki/Npm> (siehe S. 17).
- [49] Wikipedia. *WebStorm*. Aug. 2023. URL: <https://de.wikipedia.org/wiki/WebStorm> (siehe S. 15).
- [50] Wikipedia. *CSS*. URL: <https://en.wikipedia.org/wiki/CSS> (siehe S. 20).
- [51] ZTM. *Complete Node.js Developer in 2023*. Aug. 2023. URL: <https://zerotomastery.io/courses/learn-node-js/> (siehe S. 17).