

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.

Giessen, den 01. September 2023

Yvan Richnel Tchiengue

Inhalt

Inhalt	6
Abbildungen	7
Tabellen	9
Listings	10
Abkürzungen und Formelzeichen	11
1 Einleitung	13
1.1 Motivation und Problemstellung	13
1.2 Zielsetzung	14
1.3 Aufbau der Arbeit	15
2 Stand der Wissenschaft und Technik	16
3 Grundlagen	17
3.1 Entwicklungswerkzeuge	17
3.1.1 Git	17
3.1.2 GitHub	17
3.1.3 WebStorm	17
3.1.4 NodeJS	18
3.1.5 NPM	18
3.2 Verwendete Technologien zur Webentwicklung	18
3.2.1 Angular	18
3.2.2 Html	19
3.2.3 TypeScript	19
3.2.4 JavaScript	20
3.2.5 CSS	20
3.3 Konzepte der Webentwicklung	20
3.3.1 HTTP	20
3.3.2 JSON	21
3.3.3 API	21

3.4	Webanwendung	22
3.4.1	Wie funktioniert eine Webanwendung?	22
3.5	Mobilanwendung	24
3.5.1	Funktionsweise einer mobilen Anwendung	24
3.6	Unterschied zwischen Web- und Mobilanwendung	25
4	Methodik	27
4.1	Anforderungsanalyse	27
4.1.1	Analyse funktionaler Anforderungen	27
4.1.2	Analyse nicht-funktionaler Anforderungen	28
4.1.2.1	Sicherheit	28
4.1.2.2	Leistung	29
4.1.2.3	Benutzerfreundlichkeit	29
4.2	Schritte zur Entwicklung einer Webanwendung	30
4.2.1	Analyse der Bedürfnisse und Ziele	30
4.2.2	Design der Anwendung	30
4.2.3	Back-End-Entwicklung	31
4.2.4	Front-End-Entwicklung	31
4.2.5	Durchführung der Tests	32
4.2.6	Deployment der Webanwendung	32
4.3	Kommunikation zwischen Front-End und Back-End	33
4.4	Testautomatisierung	34
4.4.1	Unit Tests	35
4.4.1.1	Vorteile von Unit Testing	35
4.4.1.2	Karma	35
4.4.1.3	Jasmine	36
4.4.2	End-to-End Tests	36
4.4.2.1	Cypress	37
5	Bearbeitungsphase	38
5.1	Verwendete Bibliotheken	38
5.1.1	CORS	38
5.1.2	JWT	39
5.1.3	Express	39
5.1.4	Bcrypt	39
5.1.5	Multer	40
5.2	Implementierung der wichtigsten Funktionen	40
5.2.1	Konfiguration des Express-Servers und Erstellen der Datenbank	40
5.2.2	Erstellung eines Benutzerkontos auf der Plattform	42

5.2.3	Implementation der Anmeldung bei einem bestehenden Benutzerkonto	45
5.2.4	Dashboard	48
5.2.5	Jobsuche	49
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
5.2.9	Umsetzung von Unit-Tests und e2e-Tests	56
6	Ergebnissbewertung	61
7	Zusammenfassung und Ausblick	64
Literatur	65	

Abbildungen

3.1	Angular Übersicht (aus [19])	19
3.2	Html (aus [10])	19
3.3	Schematische Darstellung des Kommunikationsprozesses gemäß HTTP-Protokoll (aus [11])	21
3.4	HOW API WORKS (aus [1])	22
3.5	The Flow of the Web Application (aus [13])	24
3.6	Mobile application Übersicht (aus [24])	24
4.1	Web Application security against Cyber Attack (aus [33])	29
4.2	User Experience, User Interface (aus [36])	31
4.3	Back-End vs. Front-End Development (aus [31])	32
4.4	Communication between Frontend and Backend (aus [29])	34
4.5	Test Automation (aus [16])	34
4.6	Unit Testing Workflow (aus [5])	35
4.7	End-to-End Testing Lifecycle (aus [17])	36
4.8	Cypress Workflow (aus [7])	37
5.1	How CORSS works (aus [4])	38
5.2	Json Web Token Übersicht (aus [26])	39
5.3	Hashing Passwords with Bcrypt (aus [6])	40
5.4	Formular zur Kontoerstellung	44
5.5	Anmeldung bei einem bestehenden Benutzerkonto	46
5.6	Boardingboard für den Arbeitnehmer	48
5.7	Boardingboard für den Arbeitgeber	48
5.8	Anzeige von verfügbaren Stellenangeboten	49
5.9	Anzeige von verfügbaren Stellenangeboten	50
5.10	Implementierung der Sicherung wichtiger Dokumente in der Webanwendung	51
5.11	Ausfüllen des Formulars und Veröffentlichung einer Stellensuche	53
5.12	Einreichen der Bewerbung	55
5.13	Verwaltung von Bewerbungen	55
5.14	Implementierung von Unit-Tests für die verschiedenen Komponenten der Anwendung	57
5.15	E2e Test für den Aufruf von verfügbaren Stellenangeboten . .	59

5.16 E2e Test zur Erstellung eines Kontos	59
---	----

Tabellen

3.1	Unterschied zwischen Web- und Mobilanwendung.	26
4.1	Nicht-funktionaler Anforderungen.	30
5.1	Testfälle von User Stories	60

Listings

5.1	Serverkonfiguration	40
5.2	Konfiguration der Datenbankerstellung	41
5.3	Front-End Code des Formulars zur Erstellung eines Benutzerkontos	42
5.4	API für die Methode zur Erstellung eines Kontos	44
5.5	implementation of an API route for creating user accounts in our application	44
5.6	Code für die Anmeldung bei einem Benutzerkonto	45
5.7	API Route für die Verbindung mit einem Benutzerkonto.	46
5.8	Methode für die Verbindung mit einem Benutzerkonto.	46
5.9	Anzeige aller in der Datenbank verfügbaren Jobs	50
5.10	Sicherung wichtiger Dokumente in der Webanwendung	51
5.11	Verwaltung des Uploads von Identitätsdokumenten	52
5.12	Implementierung des Controllers und der Api-Route	53
5.13	Implementierung der Funktion zur Veröffentlichung von Stellenangeboten	53
5.14	Methode zum Einreichen der Bewerbung	55
5.15	Abruf von mit Stellenangeboten verknüpften Bewerbungen	56
5.16	Herunterladen des Personalausweises eines Bewerbers	56
5.17	Unit-Tests für die Komponente ConnectionComponent	57
5.18	Abruf von mit Stellenangeboten verknüpften Bewerbungen	59

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 Lands 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 die Personen, 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 jedes 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 **Bearbeitungsphase** 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.[\[14\]](#)

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.[\[22\]](#)

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[27]. Es wurde entwickelt, um alles, von kleinen bis zu sehr großen Projekten, schnell und effizient zu verwalten.

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.

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.[41]

Eines der einzigartigen Merkmale 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.[47]

3.1.4 NodeJS

Historisch gesehen wurde JavaScript hauptsächlich als clientseitige Programmiersprache betrachtet, die dazu verwendet wurde, Webseiten mit Interaktivität zu versehen. 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.[45]

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.[46]

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.[21]



Abb. 3.1: Angular Übersicht (aus [19])

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.[25]

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.[23]

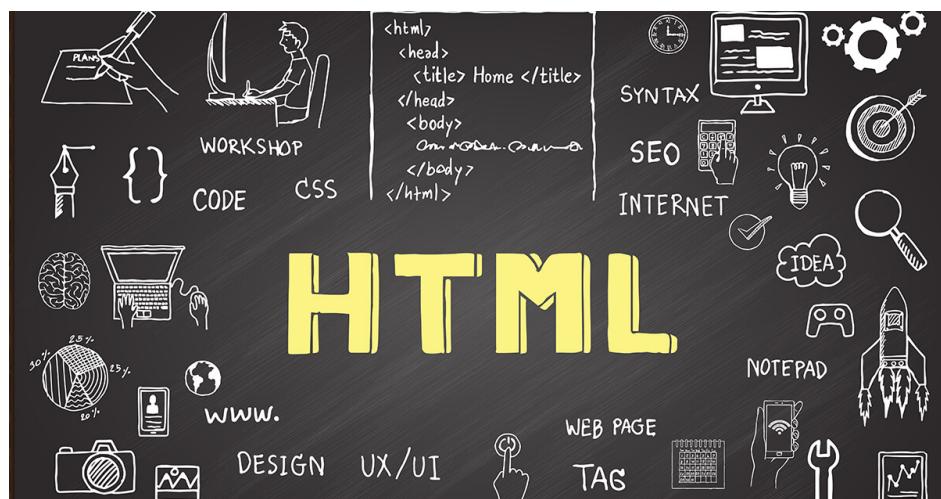


Abb. 3.2: Html (aus [10])

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 Ja-

vaScript 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.[35]

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.[12]

Trotz vieler Kritikpunkte hat JavaScript den Test der Zeit bestanden und ist nach wie vor eine der beliebtesten und meistgenutzten Programmiersprachen der Welt[3]. 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.[48]

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 Webbrower) 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.[42]

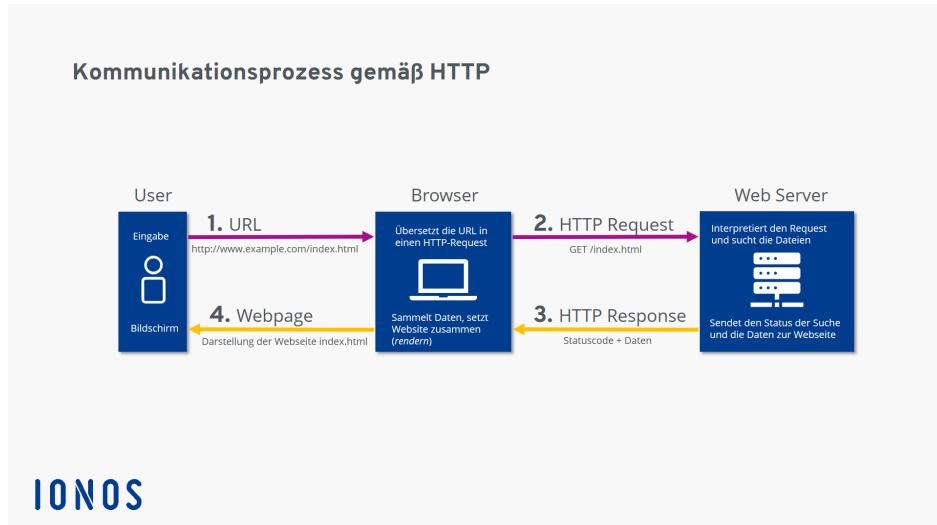


Abb. 3.3: Schematische Darstellung des Kommunikationsprozesses gemäß HTTP-Protokoll (aus [11])

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.[43]

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.[38]

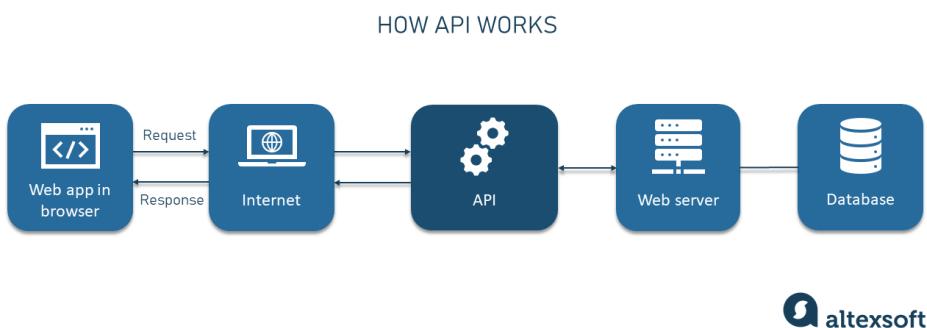


Abb. 3.4: HOW API WORKS (aus [1])

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.[13]

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.

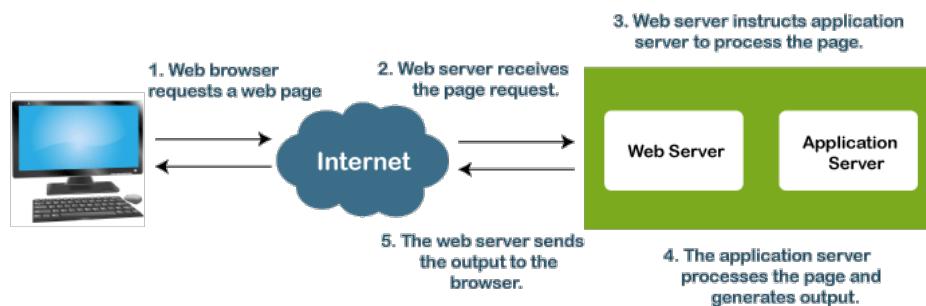


Abb. 3.5: The Flow of the Web Application (aus [13])

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.[30]

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

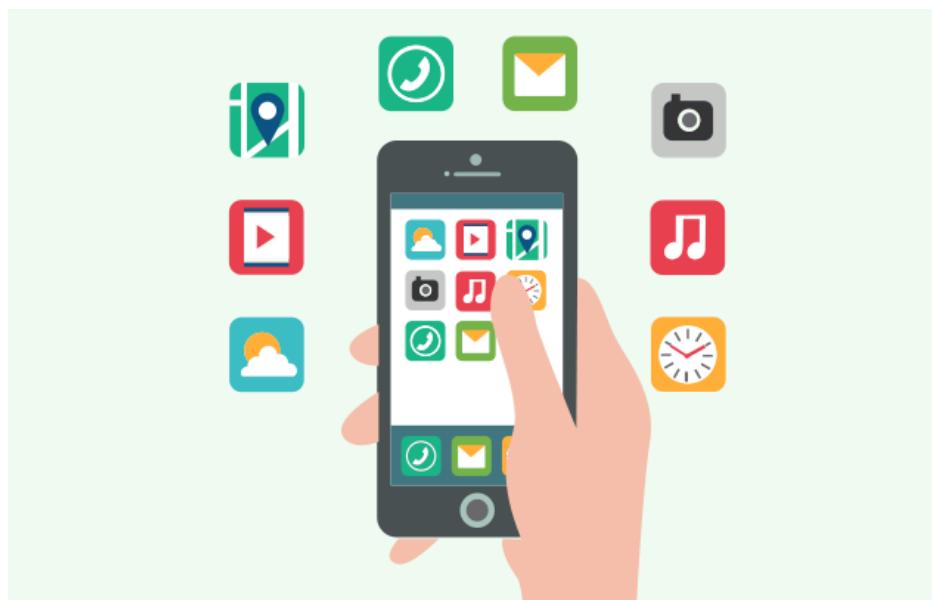


Abb. 3.6: Mobile application Übersicht (aus [24])

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 Ge-

rät 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 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[34]	Sie werden entwickelt, um auf mobilen Geräten wie Smartphones und Tablets installiert und ausgeführt zu werden.[34]
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 [33])

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.

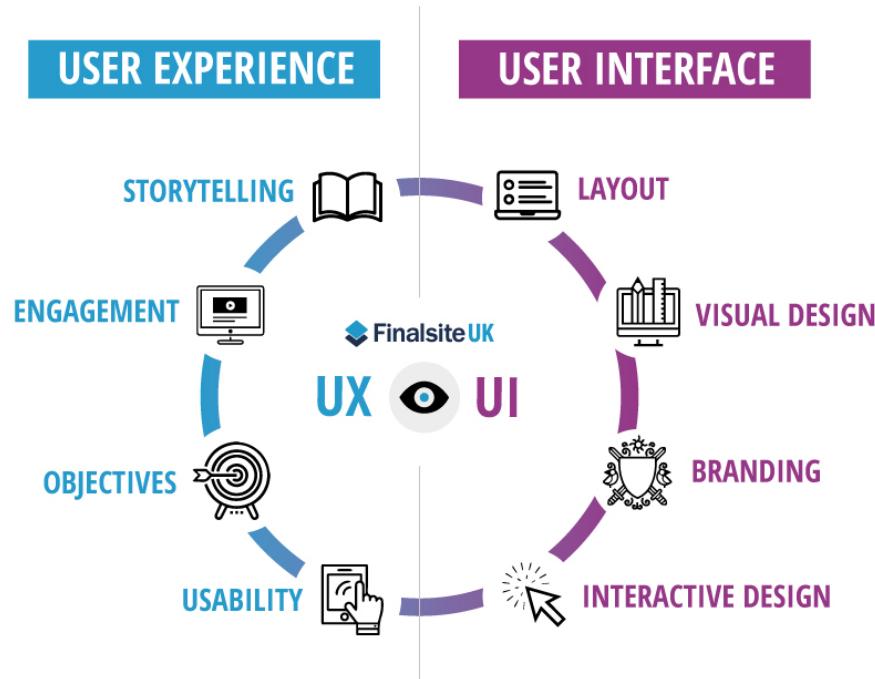


Abb. 4.2: User Experience, User Interface(aus [36])

4.2.3 Back-End-Entwicklung

Die Back-End-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 Back-End-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.[28]

Die Verwaltung der Datenbank ist ein weiterer wichtiger Aspekt der Back-End-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.

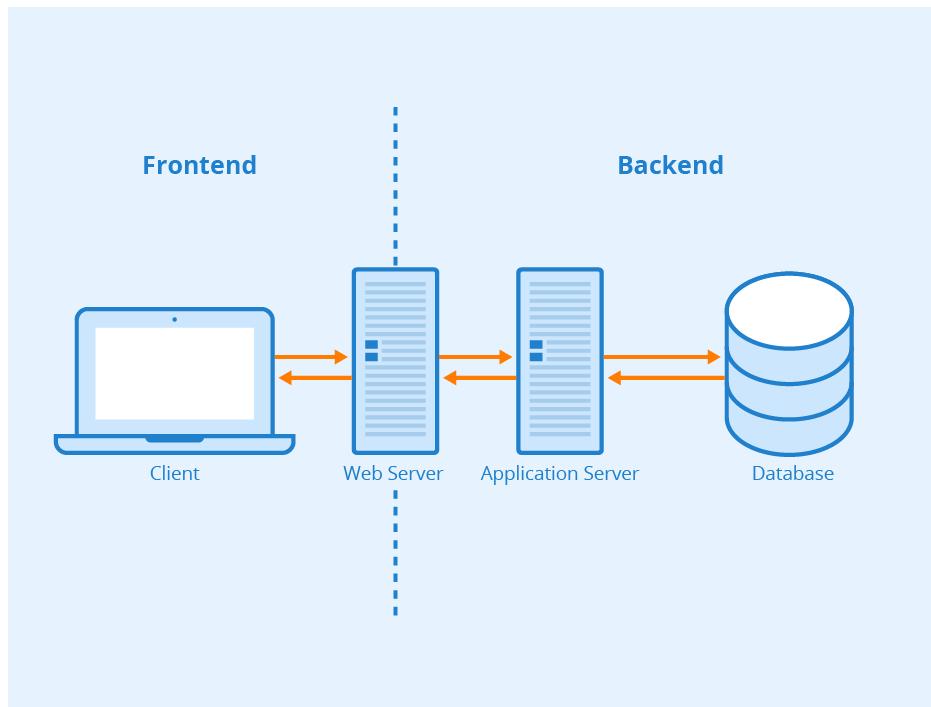


Abb. 4.3: Back-End vs. Front-End Development(aus [31])

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.[18]
- **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.[15]
- **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[9].

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.[37]

4.3 Kommunikation zwischen Front-End und Back-End

Die Kommunikation zwischen Frontend und Backend erfolgt in der Regel mithilfe von HTTP-Anfragen.[29] 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.

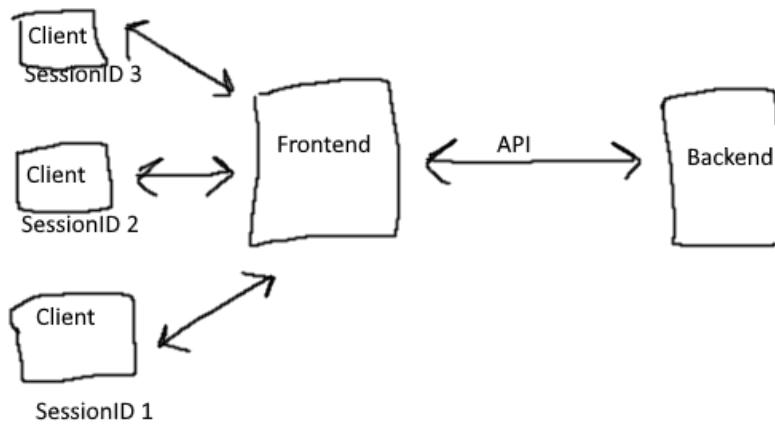


Abb. 4.4: Communication between Frontend and Backend (aus [29])

4.4 Testautomatisierung

Im digitalen Zeitalter, in dem Software in fast allen Bereichen unseres täglichen Lebens eine vorherrschende Rolle spielt, ist die Gewährleistung ihrer Zuverlässigkeit, Leistung und Sicherheit zwingend notwendig geworden. In diesem Zusammenhang stellt sich das Testen von Software als ein entscheidender Schritt im Entwicklungszyklus dar. Während manuelles Testen lange Zeit die Norm war, haben die schnelle Expansion der Technologien, die zunehmende Komplexität der Anwendungen und die Notwendigkeit, schneller auf den Markt zu kommen, einen effizienteren Ansatz hervorgebracht: die Testautomatisierung.

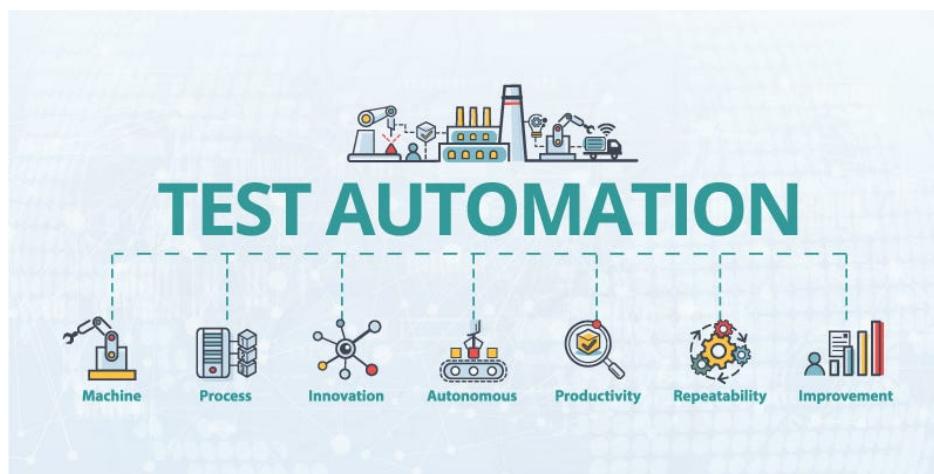


Abb. 4.5: Test Automation (aus [16])

Bei der Testautomatisierung werden Tests automatisch durchgeführt, die Testdaten verwaltet und die Ergebnisse zur Verbesserung der Softwarequalität genutzt.[32] Im weiten Feld der automatisierten Tests gibt es zwei Arten, die sich aufgrund ihrer Bedeutung und Relevanz im Softwareentwicklungszyklus besonders hervorheben: **Unit-Tests** und **End-to-End-Tests (oder e2e-Tests)**.

4.4.1 Unit Tests

Der Unit-Test bezeichnet eine Methode der Softwarevalidierung, die sich auf die individuelle Prüfung von Einheiten oder Komponenten eines Systems konzentriert. Sein Hauptziel ist es, zu bestätigen, dass jedes Segment der Software gemäß seinen geplanten Spezifikationen arbeitet und die festgelegten Kriterien erfüllt. Üblicherweise werden diese Tests von den Entwicklern durchgeführt und finden in den frühen Phasen der Entwicklung statt, lange bevor der Code integriert und als Teil des Gesamtsystems validiert wird.[5]

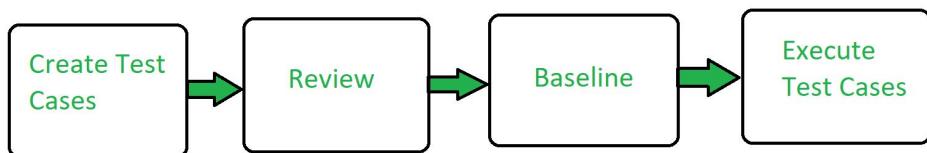


Abb. 4.6: Unit Testing Workflow (aus [5])

4.4.1.1 Vorteile von Unit Testing

Unit-Tests bieten mehrere Vorteile, die die Qualität der Software steigern und die Effizienz des Entwicklungsprozesses verbessern. Hier sind einige der bemerkenswertesten Vorteile von Unit-Tests[5] :

- **Frühzeitige Fehlererkennung:** indem jede Komponente einzeln getestet wird, können Fehler bereits in einem frühen Stadium der Entwicklung erkannt und behoben werden, was oftmals kostengünstiger ist als die Behebung von Problemen zu einem späteren Zeitpunkt im Zyklus.
- **Verbesserung der Codequalität:** Entwickler, die Unit-Tests schreiben, achten oft mehr auf die Modularität und Sauberkeit ihres Codes, da sie wissen, dass er ausführlich getestet wird.
- **Code-Dokumentation:** Unit-Tests dienen als lebendige Dokumentation. Durch Beobachtung der Tests kann man verstehen, wie ein bestimmter Teil des Codes funktionieren soll.

4.4.1.2 Karma

Karma basiert auf Node.js und ist ein Werkzeug, mit dem JavaScript-Skripte in verschiedenen konkreten Browsern bewertet werden können. Dieses Instrument, das oft als "Test Runner" bezeichnet wird, optimiert und vereinfacht die testgetriebene Entwicklung. Dabei ist zu beachten, dass es vom Angular-Team entwickelt wurde.[8]

Als Teststarter führt Karma drei Hauptaufgaben aus[8]:

- Es richtet einen Webserver ein und überträgt Ihre JavaScript-Quellskripte und die dazugehörigen Testdateien auf diesen Server.
- Er organisiert und lädt alle Dateien, sowohl die Quell- als auch die Testdateien, in einer festgelegten Reihenfolge.
- Schließlich aktiviert er die Browser, um die Tests durchzuführen.

4.4.1.3 Jasmine

jasmine ist ein Open-Source-Framework für verhaltensgesteuerte Entwicklung beliebt für den Unit-Test von JavaScript-Code.[49] Hier sind einige Schlüsselpunkte über Jasmine[49] :

- **Verständliche Syntax:** Jasmine verwendet eine klare, leicht verständliche Syntax, die von den Frameworks für verhaltensbasierte Tests (Behavior-Driven Development, BDD) inspiriert ist. Dies ermöglicht es, Tests zu schreiben, die nicht nur funktional, sondern auch lesbar sind.
- **Descriptive and it blocks:** Die Tests in Jasmine sind in describe-Blöcken organisiert, um eine Reihe von Tests zu definieren, und **it**, um einen einzelnen Test zu definieren. Dies hilft, Tests logisch zu gruppieren und sie deskriptiv zu beschreiben.
- **Spies:** Jasmine bietet die Möglichkeit, Funktionen auszuspionieren, sodass man sehen kann, wie oft sie aufgerufen wurden, mit welchen Argumenten und anderen Aspekten. Dies ist besonders nützlich für das Testen von Interaktionen und Verhaltensweisen.
- **Mocks und Stubs:** Jasmine kann Objekte und Funktionen simulieren (oder "mocken"), was nützlich ist, wenn Sie beim Testen keine echten Funktionsaufrufe ausführen oder auf externe Dienste zugreifen wollen.

4.4.2 End-to-End Tests

End-to-End-Tests sind ein Ansatz zur Softwarevalidierung, der die Leistung und Gesamtfunktionalität einer Anwendung bewertet, indem er reale Anwendungsfälle nachahmt und aktuelle Daten verwendet. Sie sollen Fehlfunktionen aufspüren, die entstehen, wenn alle Komponenten miteinander verbunden sind, und sicherstellen, dass die Anwendung die erwarteten Ergebnisse reibungslos liefert.[17]



Abb. 4.7: End-to-End testing Lifecycle (aus [17])

End-to-End-Tests stellen sicher, dass alle Komponenten der Anwendung in realen Nutzungsszenarien wie vorgesehen interagieren. Sie decken Probleme auf, die bei der Interaktion zwischen verschiedenen Komponenten auftreten können und die bei Unit-Tests übersehen werden könnten.

4.4.2.1 Cypress

Cypress ist ein modernes automatisiertes Testwerkzeug, das sich auf die Durchführung von End-to-End-Tests für Webanwendungen konzentriert. Es wurde entwickelt, um das Schreiben, Ausführen und Debuggen von Tests in einem Browser zu erleichtern.[\[7\]](#)

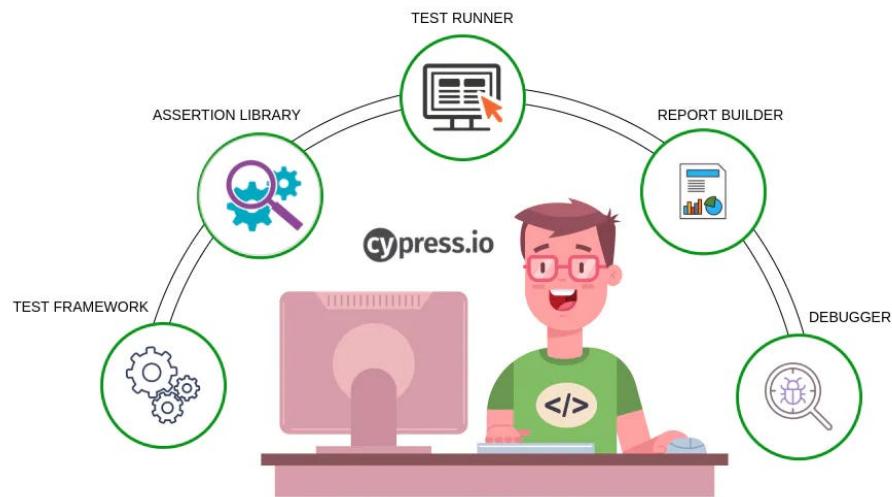


Abb. 4.8: Cypress workflow Lifecycle (aus [\[7\]](#))

5 Bearbeitungsphase

5.1 Verwendete Bibliotheken

5.1.1 CORS

CORS ist ein auf HTTP-Header basierender Mechanismus, mit dem ein Server angeben kann, von welchen anderen Ursprüngen (Domäne, Protokoll oder Port) als seinem eigenen ein Browser das Laden von Ressourcen zulassen sollte. CORS verwendet auch eine Methode, bei der Browser eine "Vorabprüfung" des Servers, der die Ressource eines anderen Ursprungs beherbergt, anfordern, um sicherzustellen, dass der Server die tatsächliche Anforderung zulässt.[\[4\]](#)

Abb. 5.1 zeigt die "Politik der gleichen Herkunft"(Same-Origin Policy). Dies ist eine Sicherheitsmaßnahme, die von Browsern eingeführt wurde, um zu verhindern, dass ein bösartiges Skript auf einer Webseite unbefugt auf Daten eines anderen Ursprungs zugreift. Gemäß dieser Richtlinie kann eine Webseite, die von einem Ursprung (z. B. <https://domain1.com>) stammt, keine AJAX-Anfrage an einen anderen Ursprung (z. B. <https://domain2.com>) ohne die Erlaubnis des zweiten Ursprungs stellen

Wenn der Browser eine Cross-Origin-Anfrage stellt, fügt er den Origin-Header hinzu, um anzugeben, woher die Anfrage kommt. Der Server empfängt die Anfrage und entscheidet, ob er diesen Ursprung zulassen will. Ist dies der Fall, gibt er in seiner Antwort einen Access-Control-Allow-Origin-Header zurück.

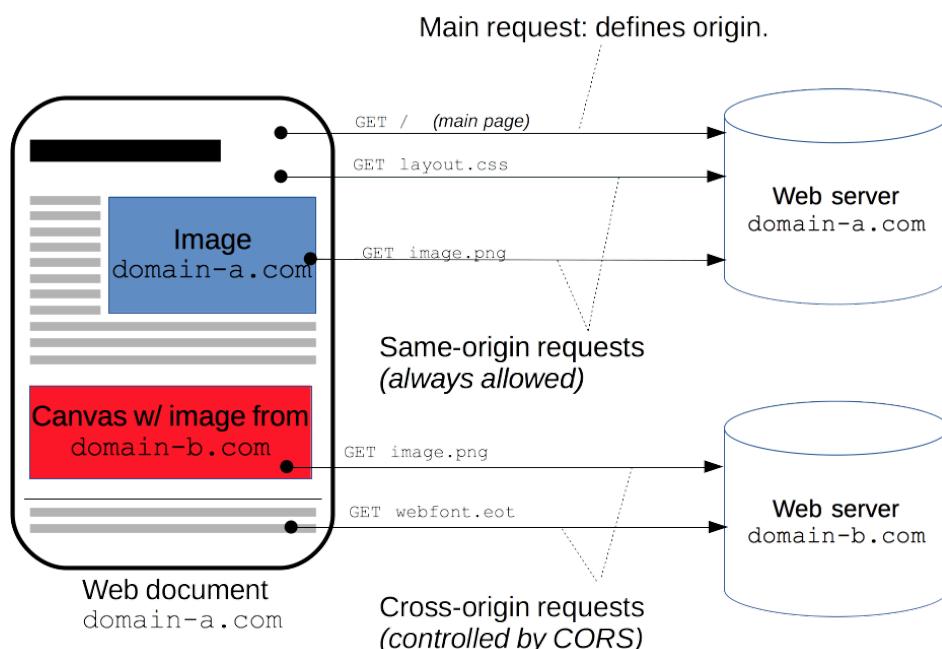


Abb. 5.1: How CORSS works (aus [4])

5.1.2 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. [44] 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

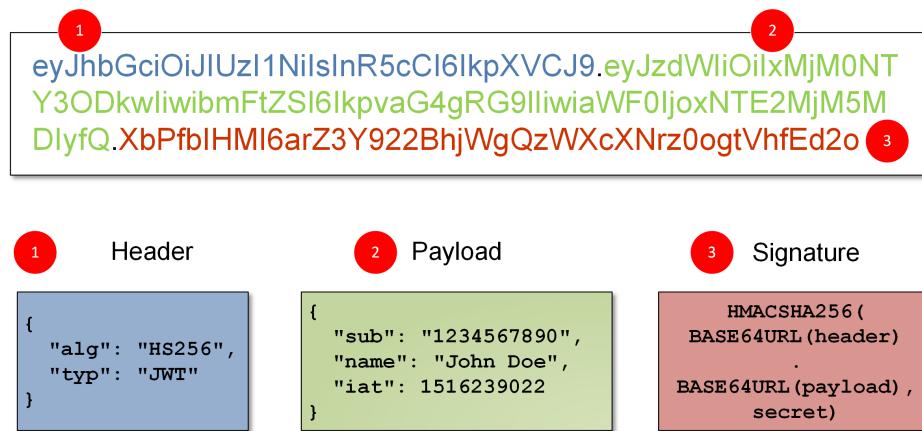


Abb. 5.2: Json Web Token Übersicht (aus [26])

5.1.3 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.[40]

5.1.4 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.[39]

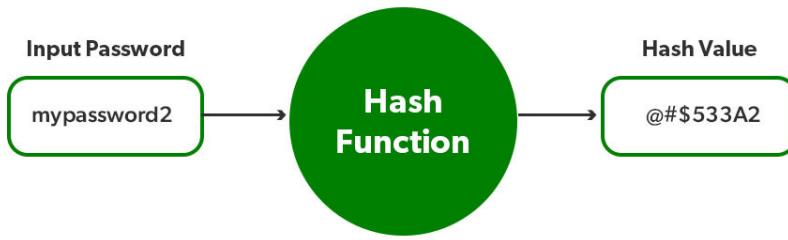


Abb. 5.3: Hashing Passwords with Bcrypt (aus [6])

5.1.5 Multer

Multer ist eine Middleware für Node.js, die speziell für die Verarbeitung von **Multipart/Form-data** entwickelt wurde, ein 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.[20] 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.

5.2 Implementierung der wichtigsten Funktionen

5.2.1 Konfiguration des Express-Servers und Erstellen der Datenbank

Listing 5.1: Serverkonfiguration

```
1 const express = require('express');
2 const app = express();
3 const port = 3000;
4 const cors = require('cors');
5 app.use(cors());
6 const bodyParser = require('body-parser');
7 app.use(bodyParser.json());
```

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

Der Code in [Listing 5.2](#) stellt eine Verbindung zu einer lokalen MySQL-Datenbank her. Die Verbindung wird dann zur späteren Verwendung exportiert. Es ist zu beachten, dass der Code zum Erstellen der Jobsearch-Datenbank nur einmal ausgeführt wird.

[Listing 5.2:](#) Konfiguration der Datenbankerstellung

```

1 const mysql = require('mysql');
2
3 const con = mysql.createConnection({
4     host: "localhost",
5     user: "root",
6     password: "*****",
7     database: 'jobSearch'
8 });
9
10 con.connect((err) => {
11     if (err) throw err;
12     console.log("Connection to database server established: Connected!");
13 });
14
15 const createDatabase = () => {
16     con.query('CREATE DATABASE IF NOT EXISTS JobSearch;', (err,
17         result) => {
18         if (err) throw err;
19         console.log("Database created");
20
21         con.changeUser({ database : 'JobSearch' }, function(err) {
22             if (err) throw err;
23
24             con.query(`CREATE TABLE IF NOT EXISTS Employer (
25                 id INT AUTO_INCREMENT PRIMARY KEY,
26                 namee VARCHAR(255),
27                 adress VARCHAR(255),
28                 email VARCHAR(255),
29                 telephone VARCHAR(255)
30             `), (err, result) => {
31                 if (err) throw err;
32                 console.log("Employer table created");
33
34                 con.query(`CREATE TABLE IF NOT EXISTS JobOffer (
35                     id INT AUTO_INCREMENT PRIMARY KEY,
36                     employerId INT,
37

```

```

38      title VARCHAR(255) ,
39      description TEXT,
40      salary FLOAT,
41      publicationDate TIMESTAMP DEFAULT CURRENT_TIMESTAMP
42      ,
43      location VARCHAR(255),
44      typeContract VARCHAR(255),
45      businessSector VARCHAR(255),
46      FOREIGN KEY (employerId) REFERENCES Employer(id)
47    )` , (err , result) => {
48      if (err) throw err;
49      console.log("JobOffer\u2022table\u2022created");
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\u2022table\u2022created");
62    });
63  });
64);
65);
66);
67};
68
69createDatabase();
70module.exports = con;

```

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

Listing 5.3: Front-End Code des Formulars zur Erstellung eines Benutzerkontos

```

1 <h2>Account Creation</h2>
2 <form [formGroup]="registerForm" (ngSubmit)="submit()">
3   <span>Name</span>
4   <input type="text" formControlName="namee" placeholder="Name">

```

```

5  <span class="error" style="color: red"
6    *ngIf="registerForm?.get('namee')?.errors?.['required'
7      '] && registerForm?.get('namee')?.touched">
8    Name is required
9  </span>
10 <span>E-mail</span>
11 <input type="email" formControlName="email" placeholder="Email">
12 <span class="error" style="color: red"
13   *ngIf="registerForm?.get('email')?.errors?.['required'
14     '] && registerForm?.get('email')?.touched">
15   Email is required
16 </span>
17 <span>Password</span>
18 <input type="password" formControlName="password"
19   placeholder="Password">
20 <span class="error" style="color: red"
21   *ngIf="registerForm?.get('password')?.errors?.['
22     required'] && registerForm?.get('password')?.touched">
23 Password is required
24 </span>
25 <span>Type of Account</span>
26 <select class="sel" formControlName="type_of_account">
27   <option value="jobseeker">JobSeeker</option>
28   <option value="employe">Employe</option>
29 </select>
30 <span class="error" style="color: red"
31   *ngIf="registerForm?.get('type_of_account')?.errors
32     ?. ['required'] && registerForm?.get('
33       type_of_account')?.touched">
34 Type of account is required
35 </span>
36 <button type="submit" [disabled]={!registerForm.valid}>
37   Create an Account</button>
38 </form>

```

The screenshot shows a dark-themed 'ACCOUNT CREATION' form. It includes fields for 'Name', 'E-mail', 'Password', and 'Type of Account'. Each field has a red error message below it: 'Name is required', 'Email is required', and 'Type of Account is required'. A large blue 'create an account' button is at the bottom.

Abb. 5.4: 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. 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 5.4: 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 5.5: implementation of an API route for creating user accounts in our application

```
1 exports.createAccount = 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
6     let tableName;
7     if (user.type_of_account === 'jobseeker') {
8         tableName = 'JobSeeker';
9     } else if (user.type_of_account === 'employe') {
10        tableName = 'Employer';
11    } else {
12        res.status(400).json({ error: 'Invalid account type' });
13    }
14
15 }
```

```

16  const query = `INSERT INTO ${tableName} SET ?`;
17  con.query(query, user, (err, result) => {
18      if (err) {
19          console.error(err);
20          res.status(500).json({ error: 'Error when
21              creating an account' });
22      }
23      res.json({ success: true });
24  });
25 };

```

Dieses Stück Code in Listing 5.5 veranschaulicht die Implementierung einer Methode für die Erstellung von Benutzerkonten in der Anwendung. 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 5.6: Code für die Anmeldung bei einem Benutzerkonto

```

1 <h2>Welcome to our website</h2>
2 <form [formGroup]="registerForm" (ngSubmit)="submit()">
3     <span>E-mail</span>
4     <input formControlName="email" name="email" placeholder="Email">
5     <span class="error" style="color: red" *ngIf="registerForm
6         ?.get('email')?.errors?.['required'] && registerForm?
7             get('email')?.touched">Email is required</span>
8
9     <span>Password</span>
10    <input formControlName="password" name="password"
11        placeholder="Mot de passe" type="password">
12    <span class="error" style="color: red" *ngIf="registerForm
13        ?.get('password')?.errors?.['required'] && registerForm
14        ?.get('password')?.touched">Password is required</span>
15    <span class="error" style="color: red" *ngIf="registerForm
16        ?.get('password')?.errors?.['minlength'] && registerForm
17        ?.get('password')?.touched">
18        Password must be at least 8 characters
19    </span>
20    <button type="submit" [disabled]="!registerForm.valid || !
21        isSubmitted">log in</button>

```

```
14 </form>
15 <a routerLink="/account-creation">Create an account</a>
```

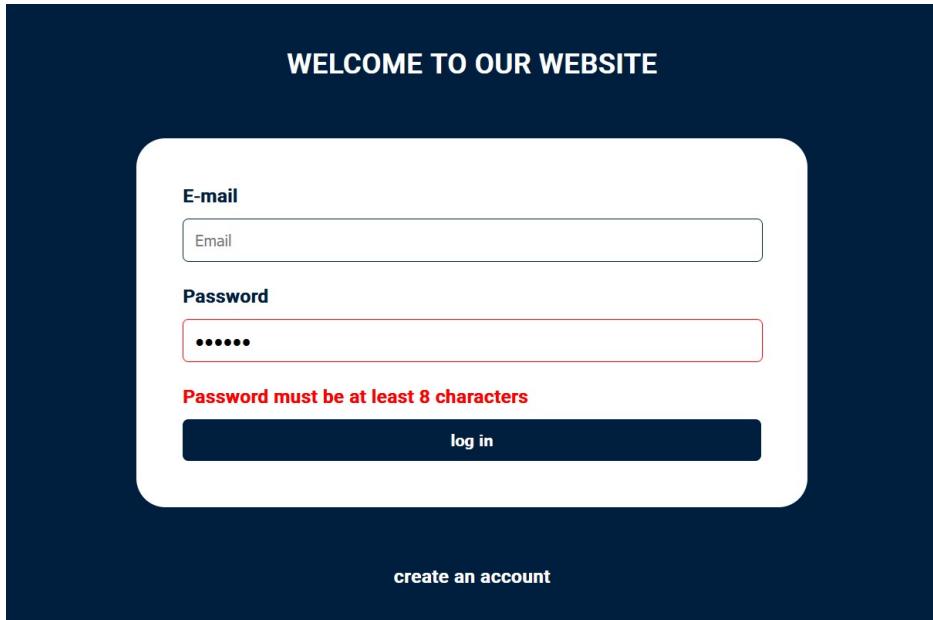


Abb. 5.5: Anmeldung bei einem bestehenden Benutzerkonto

Bei der Gestaltung des Anmeldeformulars in Abb. 5.5 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.

Listing 5.7: API Route für die Verbindung mit einem Benutzerkonto.

```
1 const account_connection = require('./controllers/account-
  connection');
2 app.post('/api/authentification', account_connection.
  accountConnection);
```

Listing 5.8: Methode für die Verbindung mit einem Benutzerkonto.

```
1 exports.accountConnection = (req, res) =>{
2   const { email, password } = req.body;
3
4   con.query('SELECT * FROM JobSeeker WHERE email = ? ', [email
  ], async (error, jobSeekerResults) => {
5     if (error) {
6       return res.status(500).json({ error: 'Database error'
      });
    }
  }
}
```

```

7      }
8
9      if (jobSeekerResults.length > 0) {
10         const user = jobSeekerResults[0];
11         const isMatch = await bcrypt.compare(password, user.
12             password);
13
14         if (!isMatch) {
15             return res.status(400).json({ error: 'Incorrect_email
16                 _address_or_password' });
17         }
18
19         const token = jwt.sign({ id: user.id, type: 'JobSeeker'
20             }, 'tony', { expiresIn: '24h' });
21         return res.json({ token, userType: user.type_of_account
22             , userID: user.id, userName: user.namee });
23     }
24
25
26     con.query('SELECT * FROM Employer WHERE email = ?',
27     [email], async (error, employerResults) => {
28         if (error) {
29             return res.status(500).json({ error: 'Database_error' });
30         }
31
32         if (employerResults.length === 0) {
33             return res.status(400).json({ error: 'Incorrect_email
34                 _address_or_password' });
35         }
36
37         const user = employerResults[0];
38         const isMatch = await bcrypt.compare(password, user.
39             password);
40
41         if (!isMatch) {app.post('/api/uploadTitleOfStayFiles',
42             (req, res)=>{
43                 console.log('test_1_ok');
44             })
45             return res.status(400).json({ error: 'Incorrect_email
46                 _address_or_password' });
47         }
48
49         const token = jwt.sign({ id: user.id, type: 'Employer'
50             }, 'tony', { expiresIn: '24h' });
51         res.json({ token, userType: user.type_of_account,
52             userID: user.id, userName: user.namee });
53     });
54 }

```

Die Authentifizierungsroute in [Listing 5.7](#) ist dafür gedacht, Anmeldeversuche von Benutzern zu verarbeiten, unabhängig davon, ob es sich um Arbeitssuchende *Job-Seeker* oder Arbeitgeber *Employe* handelt. Die Verbindungsroute in [Listing 5.8](#) wird ausgeführt, wenn die Route in [Listing 5.7](#) aufgerufen wird.

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.

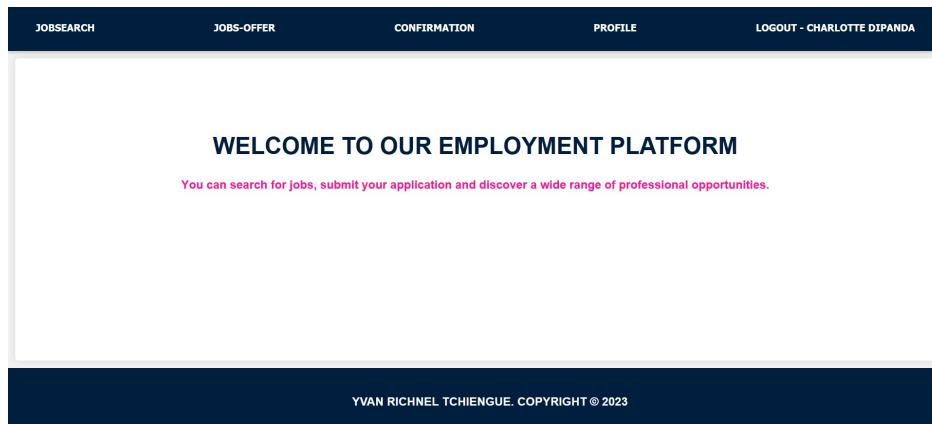


Abb. 5.6: Boardingboard für den Arbeitnehmer

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.

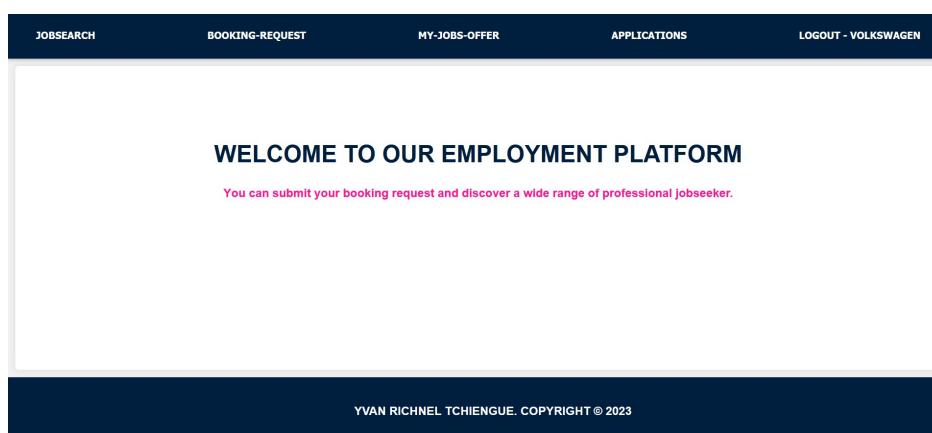


Abb. 5.7: Boardingboard für den Arbeitgeber

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.

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.

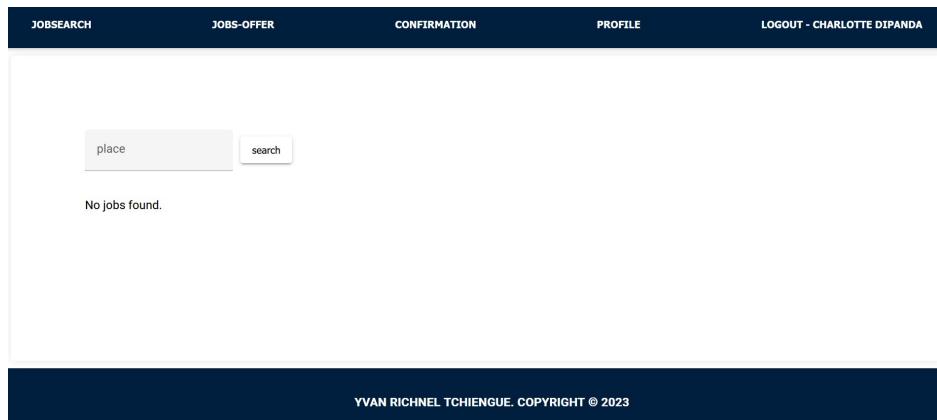


Abb. 5.8: Anzeige von verfügbaren Stellenangeboten

- **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*

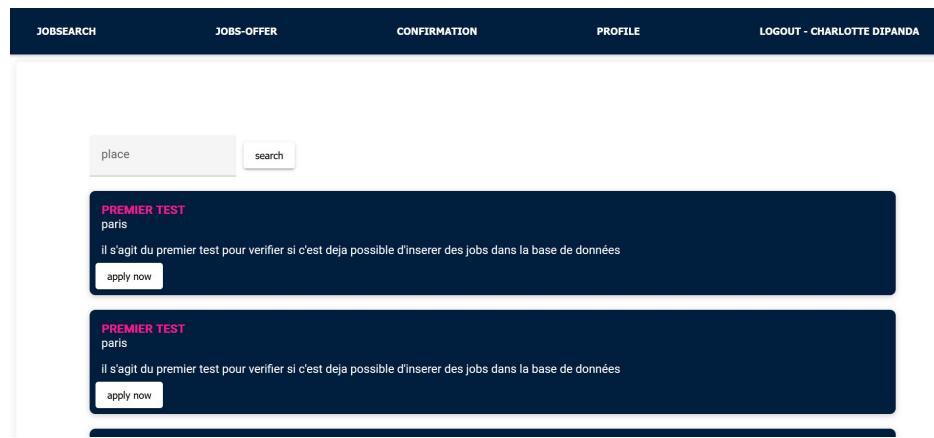


Abb. 5.9: Anzeige von verfügbaren Stellenangeboten

Listing 5.9: Anzeige aller in der Datenbank verfügbaren Jobs

```

1 exports.displayJobsoffer = (req, res) => {
2   con.query('USE JobSearch', (err) => {
3     if (err) throw err;
4     let sqlSelect = `SELECT * FROM joboffer`;
5     con.query(sqlSelect, (err, results) => {
6       if (err) throw err;
7       res.json(results);
8     });
9   });
10 }

```

Die Funktion in [listing 5.9](#), wenn sie aufgerufen wird, gibt die Gesamtheit der in der Datenbank verfügbaren Stellenangebote zurück. Diese Funktion ist von entscheidender Bedeutung, da sie den ersten Schritt der Interaktion zwischen dem Nutzer und den Daten der Plattform darstellt.

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 dark-themed web application header with tabs for 'JOBSEARCH', 'JOBS-OFFER', 'CONFIRMATION', 'PROFILE', and 'LOGOUT - CESAR BOUBEKI'. Below the header is a form section enclosed in a light gray border. Inside the form, there are four groups, each containing a file input field and a 'send' button. The first group is labeled 'IDENTITY CARD', the second 'CV', the third 'TITLE OF STAY', and the fourth 'WORK PERMIT'. Each file input field displays the message 'No file selected.' and has a 'Browse...' button to its left.

Abb. 5.10: Implementierung der Sicherung wichtiger Dokumente in der Webanwendung

Listing 5.10: Sicherung wichtiger Dokumente in der Webanwendung

```

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($event)" accept=".pdf">
7     </div>
8     <button type="submit" [disabled]="isApplied">send</button>
9   </form>
10  <form (submit)="onFilesMotivationLetterUpload()">
11    <div class="form-group">
12      <span>CV</span>
13      <input type="file" (change)="onMotivationLetterSelected($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     $event)" accept=".pdf">
29     </div>
30     <button type="submit" [disabled]="isApplied3">send</
31         button>
32   </form>
33 </div>
32 <footer class="footer">
33   <p>Yvan Richnel Tchiengue. Copyright © 2023</p>
34 </footer>
```

Der Code in [Listing 5.10](#) 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.

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.

[Listing 5.11: Verwaltung des Uploads von Identitätsdokumenten](#)

```

1 const createFileUploadConfig = (filenamePrefix) => {
2   return multer.diskStorage({
3     destination: (req, file, cb) => {
4       cb(null, 'C:/Users/yvant/Desktop/Cours/Bachelorarbeit/
5           Bachelorarbeit-JobSearch/src/assets');
6     },
7     filename: (req, file, cb) => {
8       const bearerHeader = req.headers['authorization'];
9       const bearerToken = JSON.parse(bearerHeader.replace(
10           'Bearer', ""));
11      const userId = bearerToken.userID;
12      cb(null, `${filenamePrefix}${userId}.pdf`);
13    }
14  });
15
16  const uploadFile = (filenamePrefix) => {
17    return multer({ storage: createFileUploadConfig(
18      filenamePrefix) });
19
20  const uploadTitleOfStay = uploadFile('titleofstay');
21  const uploadIdentityCard = uploadFile('identitycard');
22  const uploadWorkPermit = uploadFile('workpermit');
23  const uploadCv = uploadFile('cv');
24
25  app.post('/api/uploadTitleOfStayFiles', uploadTitleOfStay.
26    single('titleOfStay'), async (req, res) => {
});
```

```

27 app.post('/api/uploadIdentityCardFiles', uploadIdentityCard.
28   single('identityCard'), async (req, res) => {
29 });
30 app.post('/api/uploadWorkPermitFiles', uploadWorkPermit.
31   single('workPermit'), async (req, res) => {
32 );
33 app.post('/api/uploadMotivationLetterFiles', uploadCv.single(
34   'cv'), async (req, res) => {
35 );

```

Der Code in [Listing 5.11](#) verwendet multer. Die für multer angegebene Konfiguration bestimmt, wo und wie die hochgeladenen Dateien gespeichert werden.

5.2.7 Ausfüllen des Formulars und Veröffentlichung einer Stellensuche

Abb. 5.11: Ausfüllen des Formulars und Veröffentlichung einer Stellensuche

[Listing 5.12:](#) Implementierung des Controllers und der Api-Route

```

1 const booking_request = require('./controllers/booking-
2   request');
3 app.post('/api/booking-request', booking_request.
4   bookingRequests);

```

[Listing 5.13:](#) Implementierung der Funktion zur Veröffentlichung von Stellenangeboten

```

1 exports.bookingRequests = (req, res) =>{
2   const bearerHeader = req.headers['authorization'];
3   const bearerToken = bearerHeader.split(' ')[1];
4   const bearerObject = JSON.parse(bearerToken);
5   const userId = bearerObject.userID;
6   const title = req.body.title ;
7   const description = req.body.description ;

```

```

8  const location = req.body.location;
9
10 con.query('USE JobSearch', (err) => {
11   if (err) throw err;
12
13   let sqlInsert = `INSERT INTO joboffer (employerId, title,
14   description, location) VALUES (?, ?, ?, ?)`;
15   con.query(sqlInsert, [userId, title, description,
16   location], (err, result) => {
17     if (err) throw err;
18   });
19 });

```

In Listing 5.12 und Listing 5.13 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. 5.12 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. 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. 5.12: Einreichen der Bewerbung

Listing 5.14: 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 5.14 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-candidature** legt nahe, dass das Backend diese Anfrage verarbeiten wird, indem es die Bewerbung in der Datenbank speichert.

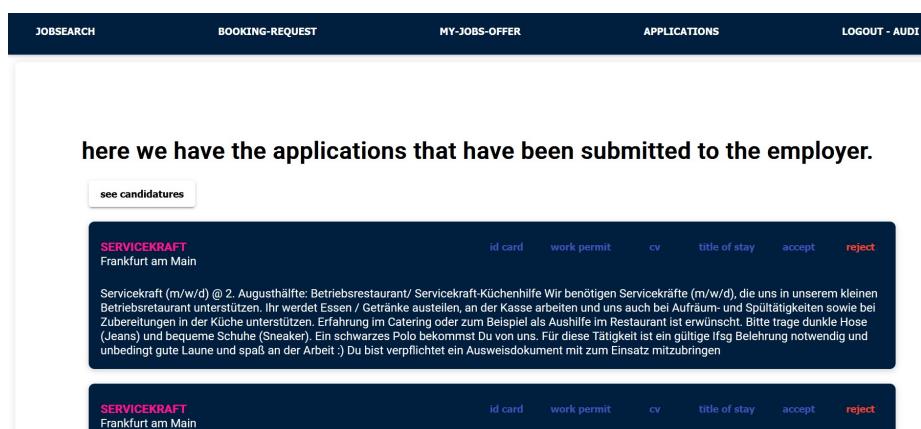


Abb. 5.13: Verwaltung von Bewerbungen

Im Einstellungsprozess ist es von entscheidender Bedeutung, einen einfachen Zugang zu den relevanten Dokumenten der Bewerber zu haben. Die in Abb. 5.13 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 5.15: Abruf von mit Stellenangeboten verknüpften Bewerbungen

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

Die Funktion `openCandidatures()` Listing 5.15 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 5.16: 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 5.16 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.

5.2.9 Umsetzung von Unit-Tests und e2e-Tests

In der Anwendung wurde auch die Praxis der Unit-Tests zur Gewährleistung der Robustheit und Zuverlässigkeit des Codes übernommen.

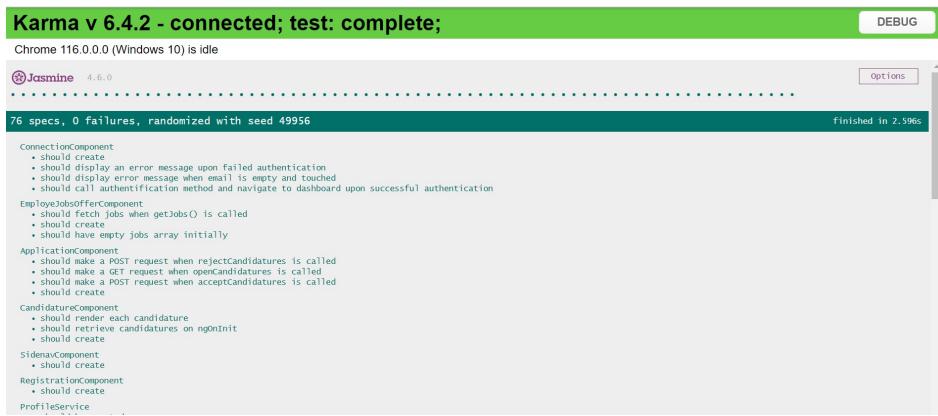


Abb. 5.14: Implementierung von Unit-Tests für die verschiedenen Komponenten der Anwendung

Wie die Abb. 5.14 zeigt, ist jeder Test darauf ausgelegt, eine bestimmte Komponente der Anwendung zu überprüfen. Nachdem wir die Unit-Tests ausgeführt hatten, beobachteten wir, dass alle Tests erfolgreich waren. Dies bestätigt, dass die getesteten Komponenten der Anwendung wie erwartet funktionieren. Dieser Erfolg ist das Ergebnis eines klaren Verständnisses der Anforderungen und einer effektiven Durchführung der Unit-Tests.

[Listing 5.17](#) ist eine Sammlung von Unit-Tests für die Komponente ConnectionComponent der Anwendung. Er verwendet das Jasmine-Framework, um die Tests zu definieren und auszuführen. Die Tests überprüfen die Erstellung der Komponente, die Validierung des E-Mail-Feldes, die Navigation zum Dashboard nach erfolgreicher Authentifizierung und die Anzeige eines Fehlers, wenn die Authentifizierung fehlschlägt.

Listing 5.17: Unit-Tests für die Komponente ConnectionComponent

```
1 describe('ConnectionComponent', () => {
2   let component: ConnectionComponent;
3   let fixture: ComponentFixture<ConnectionComponent>;
4   let mockAuthService: jasmine.SpyObj<JobOfferService>;
5   let mockSessionService: jasmine.SpyObj<SessionService>;
6
7   beforeEach(() => {
8     mockAuthService = jasmine.createSpyObj('JobOfferService',
9       ['authentication']);
10    mockSessionService = jasmine.createSpyObj('SessionService',
11      ['setSession']);
12
13    TestBed.configureTestingModule({
14      declarations: [ConnectionComponent, DashboardComponent],
15      imports: [ReactiveFormsModule, HttpClientTestingModule,
16        RouterTestingModule, RouterTestingModule.
17          withRoutes([
18            { path: 'dashboard', component: DashboardComponent }
19          ]),
20      providers: [
21      ]});
22  });
23
24  it('should ...', () => {
25    fixture.detectChanges();
26    expect(component).toBeTruthy();
27  });
28});
```

```
17      { provide: JobOfferService , useValue: mockAuthService
18          },
19      { provide: SessionService , useValue:
20          mockSessionService }
21  );
22
23  fixture = TestBed.createComponent(ConnectionStringComponent);
24  component = fixture.componentInstance;
25  fixture.detectChanges();
26 });
27
28 it('should create', () => {
29   expect(component).toBeTruthy();
30 });
31
32 it('should display error message when email is empty and
33   touched', () => {
34   const emailControl = component.registerForm.get('email');
35   emailControl?.markAsTouched();
36   fixture.detectChanges();
37   const emailErrorElement: HTMLElement = fixture.
38    .nativeElement.querySelector('.error');
39   expect(emailErrorElement.textContent).toContain('Email is
40   required');
41 });
42
43
44 it('should call authentication method and navigate to
45   dashboard upon successful authentication', () => {
46   mockAuthService.authentication.and.returnValue(of({
47     token: 'testToken', userType: 'tony1', userID: '123',
48     userName: 'tony' }));
49
50   component.submit();
51
52   expect(mockAuthService.authentication).toHaveBeenCalled();
53   expect(mockSessionService.setSession).
54     toHaveBeenCalledWith('testToken', 'tony1', '123', 'tony');
55 });
56
57 it('should display an error message upon failed
58   authentication', () => {
59   mockAuthService.authentication.and.returnValue(
60     throwError({ error: { error: 'Auth Error' } }));
61   spyOn(window, 'alert');
62
63   component.submit();
```

```

53
54     expect(window.alert).toHaveBeenCalledWith('Authentication
      ↳error: ↳wrong ↳email ↳address ↳or ↳password!');
55   });
56 });

```

Abb. 5.15 und **Abb. 5.16** zeigen, dass die e2e-Tests erfolgreich waren und somit die Qualität und Robustheit der Anwendung in verschiedenen Nutzungsszenarien validiert wurde. Dadurch wird sichergestellt, dass nicht nur jede Komponente einzeln funktioniert, sondern dass das gesamte System die Erwartungen erfüllt, wenn es unter realen Bedingungen eingesetzt wird.

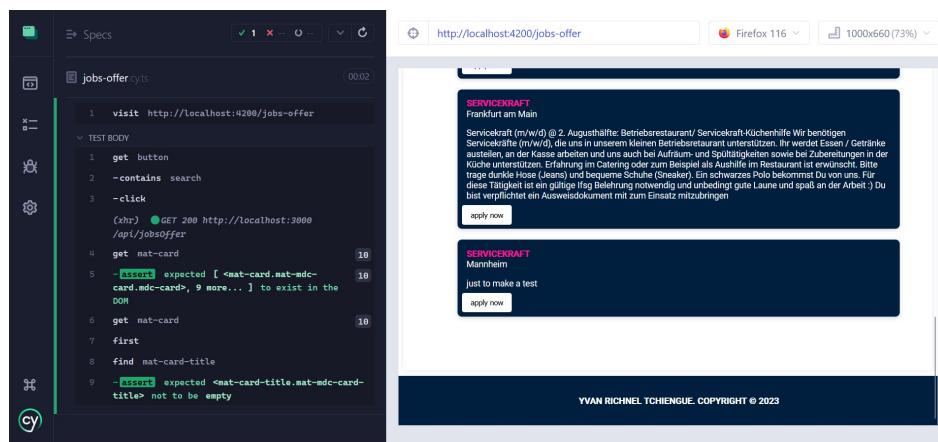


Abb. 5.15: E2e Test für den Aufruf von verfügbaren Stellenangeboten

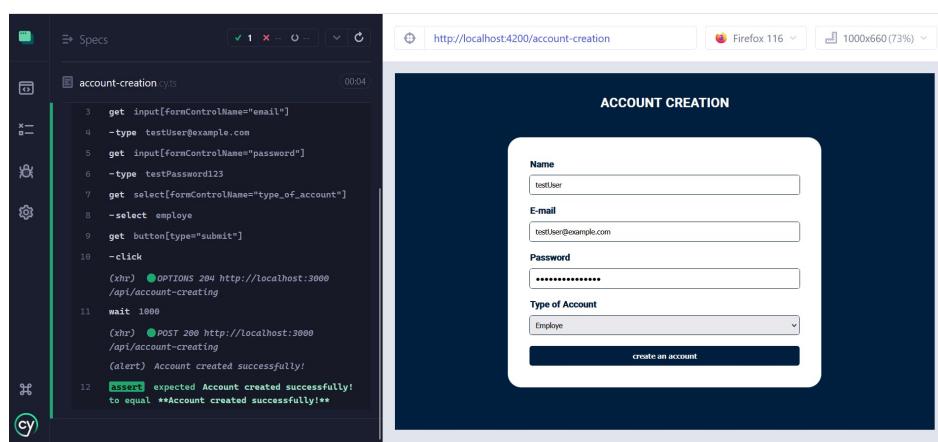


Abb. 5.16: E2e Test zur Erstellung eines Kontos

Listing 5.18 ist ein Satz automatisierter e2e Tests für die Anmeldefunktionalität eines Benutzerkontos, der das Cypress-Framework verwendet.

Listing 5.18: Abruf von mit Stellenangeboten verknüpften Bewerbungen

```

1 describe('User account login', () => {
2   beforeEach(() => {
3     cy.visit('http://localhost:4200/connection');
4   });
5

```

```

6  it('must enable the user to connect successfully', () => {
7    cy.get('input[formControlName="email"]').type('yvan.
8      tchiengue1504@yahoo.fr');
9    cy.get('input[formControlName="password"]').type('
10     qqqqqqqq');
11   cy.get('button[type="submit"]').click();
12   cy.url().should('include', '/dashboard');
13 });
14
15 it('should fail when logging in with incorrect credentials',
16   () => {
17   cy.get('input[formControlName="email"]').type('
18     wrongEmail@example.com');
19   cy.get('input[formControlName="password"]').type('
20     wrongPassword123');
21   cy.get('button[type="submit"]').click();
22   cy.on('window:alert', (str) => {
23     expect(str).to.equal('Erreur lors de l\'
24       authentification: Authentication error: wrong email
25         address or password!');
26   });
27 });
28 });

```

User Stories	Status von User Stories
Erstellung eines Benutzerkontos	✓
Anmeldung bei einem bestehenden Benutzerkonto	✓
Jobsuche	✓
Sich auf ein Stellenangebot bewerben	✓
Ergebnisse der Bewerbungen	✓
Speicherung von Dateien	✓
Abmelden des Benutzers	✓
Stellenangebote durch den Arbeitgeber veröffentlichen	✓
Abfrage von Stellenangeboten	✓
Empfang von Bewerbungen durch den Arbeitgeber	✓

Tab. 5.1: Testfälle von User Stories

6 Ergebnissbewertung

Im Rahmen der Entwicklung der Webanwendung zur Stellensuche konzentrierte sich unsere Bewertung auf die wichtigsten User Stories, die wir in Abschnitt 4.1.1 zusammengefasst haben:

- **Erstellen eines Benutzerkontos:** Wir haben Anmeldetests durchgeführt, bei denen verschiedene Nutzer das Anmeldeformular ausgefüllt haben. Der Prozess erwies sich als intuitiv und die Nutzer konnten sich erfolgreich anmelden.
- **Anmelden bei einem bestehenden Benutzerkonto:** Die Authentifizierungs- tests zeigten, dass sich die Nutzer mit ihren Anmeldedaten problemlos in ihre Konten einloggen konnten.
- **Suche nach Stellenangeboten:** die angezeigten Stellenangebote stimmen gut mit den Suchkriterien der Nutzer überein, wodurch die Relevanz der Ergebnisse gewährleistet wird.
- **Auf eine Stellenanzeige bewerben:** die Nutzer können leicht navigieren, um Details zu den Stellenangeboten zu sehen und sich zu bewerben.
- **Ergebnisse der Bewerbungen:** Die Mitarbeiter erhielten erfolgreich die von den *Jobseekern* eingereichten Bewerbungen, was ihren Einstellungsprozess erleichterte. Stellensuchende profitierten von der Nachverfolgung ihrer Bewerbungen und konnten sich über den Status jeder Bewerbung informieren, unabhängig davon, ob sie angenommen oder abgelehnt wurde.
- **Speicherung von Dateien:** die Möglichkeit für Nutzer, verschiedene sensible Dokumente wie Aufenthaltstitel, Lebenslauf, Arbeitserlaubnis und Personalausweis zu speichern, wurde erfolgreich implementiert. Arbeitgeber können bei der Bewertung von Bewerbungen auf die Dokumente der Bewerber zugreifen, was den Auswahlprozess erleichtert.
- **Abmelden des Nutzers:** es wurden Tests durchgeführt, um sicherzustellen, dass sich die Benutzer leicht abmelden können und dass ihre Sitzungen ordnungsgemäß beendet werden, wodurch die Sicherheit der Benutzerdaten gewährleistet wird.
- **Veröffentlichung von Stellenangeboten durch den Arbeitgeber:** der Prozess der Veröffentlichung von Stellenangeboten ist einfach und direkt. Sobald ein Stellenangebot veröffentlicht wurde, ist es sofort auf der Plattform sichtbar, wodurch sichergestellt wird, dass die verfügbaren Stellenangebote in Echtzeit aktualisiert werden.
- **Eingang von Bewerbungen beim Arbeitgeber:** die spezielle Schnittstelle für den Eingang von Bewerbungen bietet einen klaren Überblick über alle eingegangenen Bewerbungen, was die Arbeit des Arbeitgebers erleichtert. Arbeitgeber können jede Bewerbung öffnen, um auf Details und angehängte Dokumente zuzugreifen, was die Effizienz des Bewerbungsprozesses erhöht.

Nach unserer Bewertung können wir sagen, dass unsere Webanwendung für die Stellensuche die Erwartungen und Bedürfnisse erfüllt, die in der Entwurfsphase definiert wurden. Die Schlüsselfunktionen wurden erfolgreich getestet und validiert, sodass eine effektive Lösung für Arbeitssuchende und Arbeitgeber zur Verfügung steht. Obwohl das System einsatzbereit und leistungsfähig ist, sind kontinuierliche Verbesserungen geplant, um sich an die sich ändernden Bedürfnisse des Arbeitsmarktes anzupassen.

7 Zusammenfassung und Ausblick

Diese Thesis befasste sich mit dem Entwurf und der Implementierung einer Webanwendung für die Stellensuche unter Einbeziehung der Testautomatisierung. In der heutigen Zeit, in der Digitalisierung und Wettbewerbsfähigkeit ihren Höhepunkt erreichen, ist es von entscheidender Bedeutung, dass Webanwendungen optimal funktionieren und den Nutzern eine nahtlose Erfahrung garantieren.

Die im Rahmen dieser Forschungsarbeit entwickelte Anwendung soll den Jobsuchprozess für Bewerber erleichtern und gleichzeitig eine effiziente Verwaltung für Arbeitgeber gewährleisten. Sie umfasst mehrere Funktionen, wie die Veröffentlichung von Stellenangeboten und das Versenden von Bewerbungen.

Ein entscheidender Teil dieser Thesis war der Testautomatisierung gewidmet. Automatisierte Tests wurden eingeführt, um sicherzustellen, dass jede Funktion der Anwendung wie vorgesehen funktioniert, wodurch das Risiko von Fehlern oder Bugs verringert wird. Dies gewährleistet auch eine einfachere Wartung und reibungslose Updates.

Der Erfolg dieser Forschung stellt einen wichtigen Meilenstein dar und öffnet die Tür zu weitreichenden Zukunftsperspektiven. Ein entscheidender Aspekt ist die kontinuierliche Verbesserung. Denn auf der Grundlage des Feedbacks von Nutzern und Arbeitgebern wird es möglich, Anpassungen und Verbesserungen an der Anwendung vorzunehmen, damit sie optimal auf die sich ändernden Bedürfnisse des Marktes reagieren kann.

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] 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).
- [4] developer.mozilla.org. *Cross-Origin Resource Sharing (CORS)*. Aug. 2023. URL: <https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS> (siehe S. 38).
- [5] GeeksforGeeks. *Unit Testing | Software Testing*. Aug. 2023. URL: <https://www.geeksforgeeks.org/unit-testing-software-testing/> (siehe S. 35).
- [6] geeksforgeeks. *Hashing Passwords in Python with BCrypt*. Aug. 2023. URL: <https://www.geeksforgeeks.org/hashing-passwords-in-python-with-bcrypt/> (siehe S. 40).
- [7] Kancer (Nilay) Gökirmak. *How to use Cypress for end-to-end(e2e) testing*. Jan. 2022. URL: <https://medium.com/de-bijenkorf-techblog/how-to-use-cypress-for-end-to-end-e2e-testing-beb1aa82696a> (siehe S. 37).
- [8] Software Testing Help. *Karma Tutorial: Front-End Unit Testing Using Karma Test Runner*. Juni 2023. URL: <https://www.softwaretestinghelp.com/karma-test-runner-tutorial/> (siehe S. 35).
- [9] *How to choose a Web Hosting Provider*. Apr. 2019. URL: <https://www.exposure.com/blog/how-to-choose-a-web-hosting-provider/> (siehe S. 33).
- [10] *Html*. URL: <https://www.ionos.de/digitalguide/websites/web-entwicklung/html-tags/> (siehe S. 19).
- [11] 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).
- [12] *JavaScript History*. URL: https://www.w3schools.com/js/js_history.asp (siehe S. 20).
- [13] Javatpoint. *What is a Web Application?* Aug. 2023. URL: <https://www.javatpoint.com/web-application> (siehe S. 22, 24).
- [14] 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. 16).

- [15] 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. 32).
- [16] Sunanda Karunajeewa. *Test automation 101*. Jan. 2021. URL: <https://bootcamp.uxdesign.cc/test-automation-264d219ac83> (siehe S. 34).
- [17] Katalon. *What is End-to-End Testing?* Aug. 2023. URL: <https://katalon.com/resources-center/blog/end-to-end-e2e-testing> (siehe S. 36).
- [18] Oliver Moradov. *Unit testing: definition, Examples, and Critical Best Practices*. Mai 2022. URL: <https://brightsec.com/blog/unit-testing/> (siehe S. 32).
- [19] Gaurav Mukherjee. *Angular 16 is huge*. Apr. 2023. URL: <https://itnext.io/angular-16-is-huge-67288a3ff58b> (siehe S. 19).
- [20] NPM. *Multer*. Aug. 2023. URL: <https://www.npmjs.com/package/multer> (siehe S. 40).
- [21] 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).
- [22] OWASP. *OWASP Top 10 Web Application security Risks*. URL: <https://owasp.org/www-project-top-ten/> (siehe S. 16).
- [23] Priya Pedamkar. *Versions of Html, 6.html5*. Juli 2023. URL: <https://www.educba.com/versions-of-html/> (siehe S. 19).
- [24] proceedinteractive. *Exploring the Benefits of Mobile App Development*. Jan. 2019. URL: <https://proceedinteractive.com/exploring-the-benefits-of-mobile-app-development/> (siehe S. 24).
- [25] Rinki. *Hyper Text Markup Language*. Mai 2023. URL: <https://www.c-sharpcorner.com/article/the-importance-of-html-in-web-development/> (siehe S. 19).
- [26] Michał Sajdak. *Introduction to JWT*. Okt. 2019. URL: <https://research.securitum.com/jwt-json-web-token-security/> (siehe S. 39).
- [27] Robert Sheldon. *The history of Git*. Feb. 2023. URL: <https://www.techtarget.com/searchitoperations/definition/Git> (siehe S. 17).
- [28] Joe Silk. *THE BASICS OF BACKEND DEVELOPMENT, APIS*. Juli 2022. URL: <https://www.startechup.com/blog/back-end-development/> (siehe S. 31).
- [29] 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. 33, 34).
- [30] Techopedia. *Mobile Application*. Aug. 2020. URL: <https://www.techopedia.com/definition/2953/mobile-application-mobile-app> (siehe S. 24).
- [31] Vinayak Tekade. *Getting Started with Backend Development*. März 2021. URL: <https://medium.com/coders-capsule/getting-started-with-backend-development-8ce55585e860> (siehe S. 32).
- [32] Testim. *What Is Test Automation? A Simple, Clear Introduction*. Aug. 2019. URL: <https://www.testim.io/blog/what-is-test-automation/> (siehe S. 34).

- [33] 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. 29).
- [34] 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. 26).
- [35] TypeScript. URL: <https://en.wikipedia.org/wiki>TypeScript> (siehe S. 20).
- [36] User Experience, User Interface. URL: <https://www.finalsites.com/blog/p-board/b/post/what-is-user-experience> (siehe S. 31).
- [37] What is an SSL Certificate? URL: <https://www.digicert.com/what-is-an-ssl-certificate> (siehe S. 33).
- [38] Wikipedia. API. Aug. 2023. URL: <https://en.wikipedia.org/wiki/API> (siehe S. 22).
- [39] Wikipedia. Bcrypt. Aug. 2023. URL: <https://en.wikipedia.org/wiki/Bcrypt> (siehe S. 39).
- [40] Wikipedia. Express.js. Aug. 2023. URL: <https://en.wikipedia.org/wiki/Express.js> (siehe S. 39).
- [41] Wikipedia. GitHub. Aug. 2023. URL: <https://en.wikipedia.org/wiki/GitHub> (siehe S. 17).
- [42] Wikipedia. HTTP. Aug. 2023. URL: <https://en.wikipedia.org/wiki/HTTP> (siehe S. 21).
- [43] Wikipedia. JSON. Aug. 2023. URL: <https://en.wikipedia.org/wiki/JSON> (siehe S. 21).
- [44] Wikipedia. JWT. Aug. 2023. URL: https://en.wikipedia.org/wiki/JSON_Web_Token (siehe S. 39).
- [45] Wikipedia. Nodejs. Aug. 2023. URL: <https://en.wikipedia.org/wiki/Node.js> (siehe S. 18).
- [46] Wikipedia. npm. Aug. 2023. URL: <https://en.wikipedia.org/wiki/Npm> (siehe S. 18).
- [47] Wikipedia. WebStorm. Aug. 2023. URL: <https://de.wikipedia.org/wiki/WebStorm> (siehe S. 17).
- [48] Wikipedia. CSS. URL: <https://en.wikipedia.org/wiki/CSS> (siehe S. 20).
- [49] Zymr. *Angular Unit Testing With Karma And Jasmine.* Feb. 2023. URL: <https://www.zymr.com/blog/angular-unit-testing-with-karma-and-jasmine> (siehe S. 36).