HTL - Perg
Höhere Abteilung für Informatik

Diplomarbeit

datadive

Projektteam: Fürnhammer Pia, Ziegler Katja
Projektbetreuer: Prof. Ing. Praher Patrick, MSc, BSc

In Zusammenarbeit mit der Firma Dynatrace
Kreuzer Daniel

Bearbeitungszeitraum: 01.11.2023 – 04.04.2024

1 EIDESSTATTLICHE ERKLÄRUNG

Hiermit versichern wir, die vorliegende Arbeit selbständig, ohne fremde Hilfe und ohne Benutzung anderer als den von uns angegebenen Quellen angefertigt zu haben. Alle Stellen, die wörtlich oder sinngemäß aus fremden Quellen direkt oder indirekt übernommen wurden, sind als solche gekennzeichnet.

Perg, 02.04.2024 Unterschrift Fürnhammer Pia

Fürnhammer Pia

Perg, 02.04.2024 Unterschrift Ziegler Katja

Ziegler Katja

Pia Fürnhammer

Digital unterschrieben von Pia Fürnhammer
Datum: 2024.04.03 08:36:55 +02'00'

Katja Ziegler

Digital unterschrieben von Katja Ziegler
Datum: 2024.04.03 08:53:56 +02'00'

2 GENDERERKLÄRUNG

Im Sinne der besseren Lesbarkeit werden in dieser Diplomarbeit personenbezogene Bezeichnungen, die sich zugleich auf Frauen und Männer beziehen, generell nur in der im Deutschen üblichen maskulinen Form angeführt. Dies soll jedoch keinesfalls eine Geschlechterdiskriminierung oder eine Verletzung des Gleichheitsgrundsatzes zum Ausdruck bringen.

Perg, 02.04.2024 Unterschrift Fürnhammer Pia

Fürnhammer Pia

Perg, 02.04.2024 Unterschrift Katja

Ziegler Katja

Pia Fürnhammer
Digital unterschrieben von Pia Fürnhammer
Datum: 2024.04.03 08:37:32 +02'00'

Katja Ziegler
Digital unterschrieben von Katja Ziegler
Datum: 2024.04.03 08:54:50 +02'00'

3 DANKSAGUNG

Wir möchten uns herzlich bei allen bedanken, die uns aufgrund ihrer Unterstützung unsere Diplomarbeit ermöglicht haben.

Ein besonders großer Dank gilt der Firma Dynatrace, die uns nicht nur die Diplomarbeit in der Zusammenarbeit mit ihnen, sondern auch ein sechswöchiges Praktikum ermöglicht haben. Besonders hervorheben möchten wir dabei unsere Betreuer Daniel Kreuzer und Jürgen Riegler. Die beiden haben uns nicht nur in technischen Aspekten beraten, sondern auch versucht, uns die Praktikumszeit so schön wie möglich zu gestalten.

Weiteres gilt ein besonderer Dank unserem Betreuer, Herrn Professor Ing. Patrick Praher, MSc, BSc, der uns mit Rat und Tat bei all unseren Fragen, ebenso in Bezug auf das Verfassen der Diplomarbeit, zur Seite stand. Er hat uns auch in unserem Praktikum in der Firma Dynatrace besucht, um unsere Fortschritte zu begutachten.

Weiteres gilt unser Dank auch all unseren Freunden und Bekannten, die uns bei der Korrektur und somit Finalisierung unserer Diplomarbeit tatkräftig unterstützt haben.

4 KURZFASSUNG

Der Name **datadive** hat sich aus der Vision vom Eintauchen in eine Menge aus Daten und anhand deren Informationen übersichtliche Statistiken und Analysen bereitzustellen, entwickelt. Das Ziel der Applikation ist nicht nur eine übersichtliche Auflistung aller Bewerber zu erstellen und somit der Personalabteilung die Arbeit, die während des Bewerbungsprozesses anfällt, zu erleichtern, sondern auch aus den Fakten über diese Bewerber, Statistiken und Analysen zu erstellen. Das bedeutet, dass anhand dieser Zahlen und Graphen beispielsweise herausgefunden werden kann, aus welcher Branche die meisten Interessierten kommen oder ob sie einen Abschluss einer Universität oder einer höheren Schule etc. besitzen. Auch Vergleiche der Bewerber zum Vorjahr können aufgestellt werden.

Die **Quelle** für die Daten der Bewerber stellt hierbei der Lebenslauf dar. Die Lebensläufe sind PDF-Dateien, die von einem Programm eingelesen werden. Durch ein bestimmtes Schema filtert es die ersten Bewerberinformationen, wie zum Beispiel die Fähigkeiten oder die bisherige Ausbildung und so weiter heraus. Diese sind dann in einer Datenbank abgespeichert.

Auf einer **grafischen Benutzerweboberfläche** sind die Informationen aus der Datenbank schlussendlich angezeigt. Diese kann der Benutzer nun filtern, sortieren etc. Auf einer eigenen Seite werden Statistiken anhand dieser Daten erstellt.

Als **Ergebnis** erspart sich der HR-Manager das zeitaufwändige Ansehen und Lesen der vielen verschiedenen Lebensläufe und kann durch die Filterfunktion schnell die passenden Bewerber identifizieren. Darüber hinaus erhält er einen einfachen Überblick darüber, welchen Werdegang oder welche Fähigkeiten – die für die Firma eine zukünftige Anstellung voraussetzen – die Bewerber im Mittel gemeinsam haben. Außerdem kann die Firma einfach herausfinden, wie sich die Bewerberzahl in den vergangenen Jahren entwickelt hat.

5 ABSTRACT

The name **datadive** has evolved from the vision of diving deep into a pile of data to provide clear statistics and analyses. The aim of the application is not only to present a clear list of all applicants, thus, to make the work of the recruitment manager easier, but also to enable analyses of these applicants' data using statistics. This means that through these statistics, for example, it can be determined from which industry the applicants come, or whether they have a degree from a university or a higher school, etc. Comparisons of applicants to the previous years can also be made.

The **source** for the applicant data is the CV. The CVs are PDF files that are read in by a program. A certain scheme is used to filter out the initial data, such as soft and hard skills or previous employment or education and so on. This data is then stored in a database.

On **a graphical user web interface**, the information from the database is finally displayed. The user can now filter, sort, etc. this information. Statistics based on this data are created on a separate page.

As a **result**, the recruitment manager saves time by not having to manually review and read through the many different resumes and can quickly identify suitable candidates through the filtering function. Furthermore, there is a simple overview of what career paths or genders, and much more, the applicants have in common on average. In addition, the company can easily determine how the number of applicants has increased or decreased compared to previous years.

6 INHALTSVERZEICHNIS

7 EINLEITUNG

7.1 MOTIVATION

Die Firma Dynatrace bekommt jedes Jahr zahlreiche Bewerbungen für offene Jobs oder Praktika vom In- aber auch vom Ausland. Diese Bewerbungen müssen von dem Recruiting Team händisch geprüft und verglichen werden, um die Besten für die Firma herauszufiltern. Dieser Prozess ist sehr zeitaufwendig, deshalb hat uns die Firma Dynatrace die Möglichkeit gegeben, dieses Problem im Zuge unserer Diplomarbeit zu lösen. Unsere Webapplikation soll es dem Recruiting Team mithilfe von Datenanalysen und Statistiken erleichtern**,** die perfekten Kandidaten zu finden.

7.2 ZIELSETZUNG

Unser Ziel ist eine Webapplikation, welche die Daten aller Lebensläufe analysiert und mithilfe von Statistiken übersichtlich in einem Dashboard darstellt. Aufgrund der Vereinheitlichung aller Daten aus den Lebensläufen soll parallel bei allen Bewerbern nach bestimmten Kriterien gefiltert oder gesucht werden. Die Lebensläufe sollen ebenfalls grafisch in unserem Dashboard ersichtlich sein. Die Anforderung ist, dass der Benutzer bei Bedarf die Daten aus der Datenbank mit den Daten aus anderen Lebensläufen vergleichen und eventuelle Änderungen vornehmen kann.

Die Daten, die aus den bereits erwähnten Lebensläufen entnommen werden, sollen in einer Datenbank mit relationalem Schema erfasst werden. Mithilfe einer API sollen die Informationen der Lebensläufe analysiert und diese Ergebnisse an die Webapplikation gesendet werden.

7.3 PROJEKTINHALT – ÜBERBLICK

7.3.1 Daten der Lebensläufe in Datenbank speichern

Die Lebensläufe werden in Form von PDF-Dateien in einem Ordner im Verzeichnis des Produktes abgelegt und zunächst im Projekt eingelesen.

Anhand der eingelesenen Dateien werden dann wichtige Informationen herausgefiltert, welche für den Bewerbungsprozess relevant sind. Die jeweiligen Daten, werden programmgerecht in Objekte gespeichert und schlussendlich in die Datenbank eingefügt.

Zusätzlich gibt es noch ein zweites Szenario, bei dem Daten über die Personen in die Datenbank gespeichert werden. Wenn die Angaben zu einem jeweiligen Bewerber falsch oder noch gar nicht eingetragen sind, weil beispielsweise etwas nicht richtig aus der PDF-Datei gelesen oder gefiltert worden ist, gibt es die Möglichkeit, die Eintragungen und die originale Datei zu vergleichen und etwaige Fehler richtig zu stellen.

7.3.2 Daten auswerten

Die Informationen aus den Lebensläufen, die in der Datenbank festgehalten sind, dienen als Grundlage für die Auswertungen der Statistiken.

Die Auswertungen der Daten sind im Backend auszuführen. Danach werden die Ergebnisse der Berechnungen und Analysen über die API an das Frontend gesendet. Mit Hilfe von Diagrammen und Statistiken werden die analysierten Daten übersichtlich aufbereitet und visualisiert.

Die Vereinheitlichung aller Daten von allen Bewerbern in einer Tabelle schafft die Möglichkeit, mithilfe von Filter- und Suchfunktionen diese effizient zu vergleichen.

7.3.3 Statistiken erstellen

Unser Produkt beinhaltet vier Dashboards, Yearly Applicants, Career Snapshot, Academic Snapshot und Gender Analytics. Die Statistiken werden mithilfe der Frameworks CoreUI und GoogleCharts erstellt. Diese beiden Frameworks sind von uns gewählt worden, da das Design am besten zu unserer Applikation gepasst hat.

Die Statistiken stellen die Daten aus dem Lebenslauf für den Benutzer kompakt und visualisiert dar. Das ermöglicht dem Unternehmen aus den Daten schnelle Schlüsse zu ziehen: zum einen von welchen Schulen oder Studieneinrichtung die meisten Bewerber kommen und zum anderen, wie sich die Bewerberzahlen zu den Zahlen im Vorjahr unterscheiden oder wie sich die Geschlechter der Bewerber aufteilen.

7.4 PROJEKTUMFELD

7.4.1 Projektteam

Das Projektteam besteht aus zwei Schülerinnen der HTL Perg: Pia Fürnhammer und Katja Ziegler. Pia Fürnhammer ist aufgrund ihrer Begeisterung für das Designen von Webseiten, für die Gestaltung und Implementierung des Frontends verantwortlich gewesen. Katja Ziegler war aufgrund ihres großen Interesses für Datenbanken für das Einlesen der Lebensläufe sowie die Analyse der Daten im Backend zuständig gewesen.

Abbildung 1 Pia Fürnhammer

Abbildung 2 Katja Ziegler

7.4.2 Betreuung

Abbildung 3 HTL Perg Logo

Die gesamte Diplomarbeit hat Professor Patrick Praher betreut. Er ist auch zugleich unser Professor in Programmieren mit der Sprache Java. Aber auch bei Fragen rund um das Schreiben der Diplomarbeit stand er uns zur Seite.

Abbildung 4 Dynatrace Logo

Das Produkt datadive ist weiters von Herrn Daniel Kreuzer und Jürgen Riegler, zwei Mitarbeiter unseres Auftraggebers Dynatrace, unterstützt worden. Wir haben unser Projekt innerhalb von sechs Wochen im Rahmen eines Ferialpraktikums beginnen und teilweise fertigstellen können.

In der Zeit, in der wir das Praktikum absolviert haben, ist es uns von unseren Betreuern ermöglicht worden, die Ressourcen der Firma bestmöglich zu unserem Vorteil zu nutzen, was uns bei der Umsetzung des Produktes sehr geholfen hat. Durch die Anwesenheit unserer Betreuer im selben Haus ist es uns auch durch sofortige Meetings möglich gewesen, Probleme schnell zu beheben.

Ebenfalls haben wir unseren Diplomarbeitsbetreuer aus der Schule in das Firmengebäude einladen dürfen und ihm somit den Fortschritt unseres Produktes präsentieren können.

7.4.3 Auftraggeber

Abbildung 5 Firmen Logo Dynatrace

Die Firma Dynatrace, die uns den Auftrag erteilte, hat sich zum Ziel gesetzt, dass in jeder Branche die digitalen Erfahrungen sowie Interaktionen überall fehlerfrei und sicher erfolgen können.

Daher bietet sie eine einheitliche und offene Plattform, die Observability, AIOps und Anwendungssicherheit kombiniert. Diese soll genaue Antworten und eine intelligente Automatisierung auf der Datenbasis liefern. [1]

8 THEORETISCHE UND FACHPRAKTISCHE GRUNDLAGEN UND METHODEN

8.1 VERWENDETE TECHNOLOGIEN

8.1.1 Java

Abbildung 6 Java Logo

Java ist eine oft verwendete Programmiersprache und für ihre Portabilität und Vielseitigkeit bekannt. Java läuft unabhängig vom Betriebssystem und kann somit auf verschiedensten Plattformen, wie zum Beispiel Windows, macOS, Linux, etc., verwendet werden. Die Programmiersprache ist darüber hinaus sehr flexibel und benutzerfreundlich. [2]

8.1.2 Java Spring Boot

Abbildung 7 Java Spring Boot Logo

Um einige Komplikationen**,** die Java mit sich bringt, lösen zu können, wird nun das Open-Source Framework **Java Spring Boot** verwendet**.** Die Kernaufgabe von Spring Boot ist es, einen optimierten modularen Ansatz für das Erstellen von Anwendungen in Java bereitzustellen.

Java Spring Boot selbst leitet sich wiederum von dem Anwendungsframework Java Spring ab. Es ist ein Tool, das die Entwicklung von Webapplikationen und Microservices innerhalb des Java-Frameworks optimiert.

[3]

8.1.3 PostgreSQL & pgAdmin

Abbildung 8 PostgreSQL Logo

Als Speichersystem wird für datadive die Datenbank **PostgreSQL** verwendet. Die Wahl der Datenbank ist in diesem Fall für uns eindeutig gewesen, weil der Auftraggeber diese als Wunsch geäußert hat.

PostgreSQL ist eine relationale Open-Source Datenbank. Ähnlich wie bei der Programmiersprache Java ist auch die Datenbank, die in dem Produkt verwendet wird, sehr beliebt. Sie ist sehr flexibel und einfach zu integrieren. PostgreSQL ist mit allen gängigen Betriebssystemen kompatibel. Außerdem ist sie sehr robust, erweiterbar und sicher. [4]

Zum Testen und Überprüfen jener Daten, die durch das Backend eingelesen worden sind, diente die Weboberfläche **pgAdmin**.

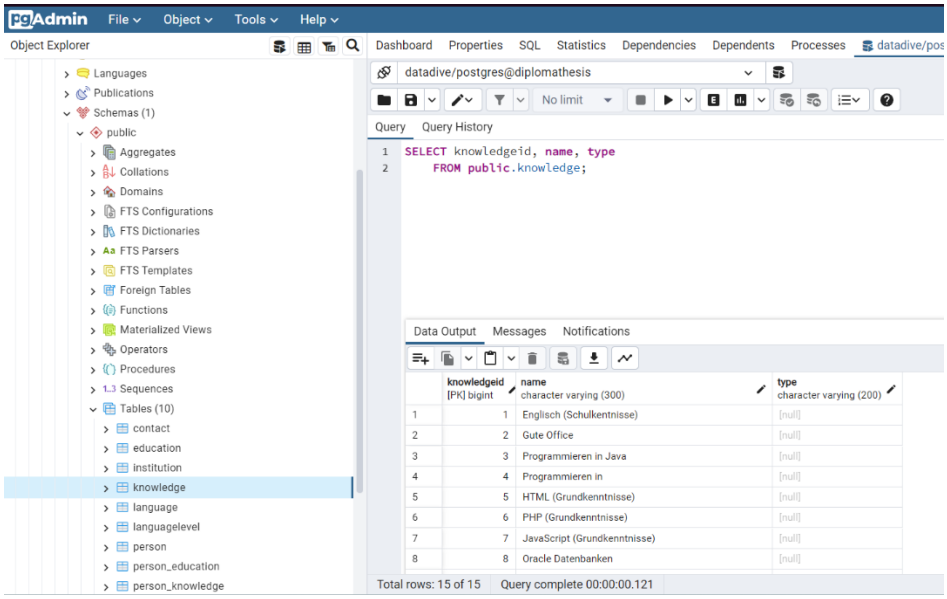


Abbildung 9 pgAdmin Weboberfläche

8.1.4 JPA & Hibernate

Abbildung 10 JPA & Hibernate Logo

JPA steht für **Java Persistence API** und definiert, wie Daten in einer Java Applikation persistiert werden.
Der Prozess, bei dem die Java Objekte in Tabellen der Datenbank umgewandelt werden, nennt man ORM – Object-Relational Mapping. Der hauptsächliche Fokus von JPA dient der Schicht des ORMs.

Dazu ist **Hibernate** eines der beliebtesten und bekanntesten ORM-Frameworks, die heutzutage verwendet werden. Hibernate gibt es bereits seit ungefähr 20 Jahren und fungiert mittlerweile als Standard der Implementierung von JPA. [5]

8.1.5 React

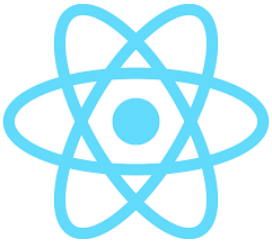


Abbildung 11 React Logo

React ist eine JavaScript Bibliothek, die für die Entwicklung von Webanwendungen verwendet wird. Eine React Application besteht aus wiederverwendbaren Components, die zusammen die Benutzeroberfläche bilden. Über die einzelnen Components kann mittels Routings navigiert werden.

Bei React handelt es sich um eine Single-Page-Application. Daher wird beim Neu laden der Seite nicht jedes Mal eine Anfrage an den Server gesendet, sondern der Inhalt dieser Seite wird direkt aus der Component geladen.

React verwendet eine Syntaxerweiterung zu JavaScript und diese heißt JSX. Das ermöglicht dem Entwickler einen HTML ähnlichen Code direkt in JavaScript zu schreiben. So können effiziente Webapplikationen entwickelt werden, da HTML-Elemente einwandfrei mit der JavaScript-Logik verknüpft werden können. [6]

8.1.6 TypeScript

Abbildung 12 TypeScript Logo

Bei **TypeScript** handelt es sich um eine Open-Source-Programmiersprache von Microsoft. Es ist eine Erweiterung von JavaScript, die auch z.B. die Funktion „Typisierung“ umfasst, was so viel bedeutet, dass jede Variable einen bestimmten Typ oder Klassendefinitionen hat.

Die Syntax wird bei TypeScript von JavaScript übernommen, demnach ist jeder JavaScript Code auch TypeScript Code.

TypeScript ermöglicht eine bessere Fehleranalyse, sowie eine schnellere Entwicklung. Es ermöglicht, größere Projekte mit vielen Entwicklern eine konsistente Codebasis zu schaffen.

Die objektorientierte Programmierung wird von dieser Webprogrammiersprache ebenfalls unterstützt, die es Entwicklern erleichtert, mit dieser Programmiersprache zu arbeiten. Es ist üblich, dass TypeScript in Kombination mit React verwendet wird, deshalb haben auch wir uns bei der Frontendentwicklung dafür entschieden. [7]

8.1.7 Git und GitHub Desktop

Abbildung 13 Git Logo

Git wird von vielen Unternehmen mit Entwicklerteams verwendet. Denn es ist ein Open-Source Tool, welches zur gemeinsamen Verwendung von Software, Code und deren Kontrolle der Versionen dient. In einem Git-Repository ist es möglich, verschiedenste Programme bzw. Applikationen hochzuladen. Andere aus dem Team können diese Version dann wieder herunterladen und weiterprogrammieren. Durch sogenannte „**Push- und Pull-Befehle**“ funktioniert dies.

Sinn von Git ist es, den Entwicklern das gemeinsame Programmieren an einem Projekt zu vereinfachen und zu strukturieren.

Abbildung 14 GitHubDesktop Logo

Obwohl Git auch simpel über das Terminal von Windows verwendet werden kann, gibt es einige Benutzeroberflächen, die den Umgang mit den bestimmten Befehlen erleichtern. Jene, die wir bei unserem Produkt datadive verwendet haben, ist **GitHubDesktop**.

Die schnell abrufbare Veranschaulichung der bisherig gespeicherten Commits hat uns unter dem Einsatz von Git unsere Arbeit erleichtert.

8.2 Verwendete Bibliotheken und Plug-Ins

8.2.1 Bootstrap

Abbildung 15 Bootstrap Logo

Bei **Bootstrap** handelt es sich um ein Frontend-Framework, mit dem Webapplikationen gestaltet werden können. Von Bootstrap werden HTML- und CSS-Vorlagen für den Entwickler zur Verfügung gestellt. Das erleichtert das Designen und Entwickeln.

Diese Vorlagen umfassen Formulare, Tabellen, Buttons, sowie Layout Vorlagen. [8]

Bootstrap stellt vorgefertigte CSS Styles, welche für die einzelnen Elementen verwendet werden können, zur Verfügung. Von der Hintergrundfarbe bis hin zur Größe der Elemente kann alles individuall angepasst werden. [8]

8.2.2 Material-UI

Abbildung 16 Material-UI-Logo

Material-UI stellt den Entwicklern UI-Komponenten zur Verfügung, die für React entwickelt wurden. [9] Es gibt verschiedene Elemente für ein strukturiertes Layout, wie zum Beispiel Container, Stack oder Grid. In unserer Applikation verwenden wir die Material-UI für das Data Grid auf unserer Overview- oder Applikationsseite. [10]

8.2.3 CoreUI

Abbildung 17 CoreUI Logo

CoreUI ist eine Erweiterung zu der bereits erwähnten Bibliothek Bootstrap. Es handelt sich dabei um ein Designframework, die es ermöglicht, ansprechende und hochwertige Anwendungen zu gestalten. [11]

Die bereitgestellten Komponenten und CSS Styles sind einfach zu verstehen, sodass die Elemente leicht in die Applikation eingebaut werden können. [12]

8.2.4 React Google Charts

Abbildung 18 React Google Charts Logo

Da bei unserem Produkt die Visualisierung der Daten eine sehr große Rolle spielt, haben wir uns für **React Google Charts** entschieden. Dabei handelt es sich um eine Bibliothek, welche für React Anwendungen Charts bereitstellt. [13]

8.2.5 ApexCharts

Abbildung 19 APEXCHARTS Logo

Bei **ApexCharts** ist genau genommen eine Bibliothek, die verschiedene Charts zur Verfügung stellt. Das ermöglicht den Entwicklern eine einfache und ansprechende Visualisierung der Daten in deren Applikationen. Zusätzlich können die ApexCharts nach Belieben angepasst werden. [14]

8.2.6 Nodejs

Abbildung 20 Node.js Logo

„**Node.js** ist eine plattformübergreifende Open-Source-Laufzeitumgebung für die Ausführung von JavaScript-Code. [15]“

Node.js kann auf Windows, Linux und MacOS Systemen verwendet werden und besitzt die V8-JavaScript-Engine, die das Parsen sowie Ausführen des JavaScript-Codes ermöglicht. Außerdem ermöglicht Node das Programmieren von „serverseitigem Code“ in der Programmiersprache JavaScript. [15]

8.2.7 Node Package Manager

Abbildung 21 NPM-Logo

Um den **Node Package Manger** nutzen zu können, müssen Node.js sowie die NPM-Registry auf dem Rechner installiert sein.

NPM findet dann Verwendung, um Bibliotheken oder Frameworks für die Projekte zu installieren. Außerdem kann beim Entwicklungsprozess mit dem Node Package Managerzeit gespart werden. [16]

8.2.8 Swagger & OpenAPI Specifikation

Abbildung 22 OpenApi Logo

Abbildung 23 Swagger Logo

Swagger und **OpenApi Specification** sind zwei Frameworks, die zur Dokumentation von APIs verwendet werden, wobei OpenApi eine weiterentwickelte Version von Swagger ist. [17]

OpenApi ist ein Tool, das automatisch eine Dokumentation des Codes generiert und eine Webapplikation zur Verfügung stellt, auf der die Endpunkte der API getestet werden können, ohne dass ein funktionierendes Frontend oder selbstgeschriebener http-Request vorhanden sein muss.

8.2.9 Axios

Abbildung 24 Axios Logo

Bei **Axios** handelt es sich um einen http-Client, basierend auf dem „$http service [in] Angular.js v1.x“. [18] Mit Axios ist es möglich**,** auf eine REST-API zuzugreifen.

Außerdem hat Axios die Vorteile von JavaScript, Axios async sowie await verwendet**,** um den asynchronen Code noch lesbarer zu gestalten. Damit Axios im Projekt verwendet werden kann**,** muss es**,** wie in Listing 1 Install Befehl für Axios ersichtlich, installiert werden. [19]

Für unsere Applikation haben wir fast nur GET-Requests benötigt. Nur beim Updaten der Daten aus den Lebensläufen benötigten wir einen PUT-Request und beim Laden der neuen Lebensläufe einen POST-Request. [19]

```
npm install axios@0.24.0
```

Listing 1 Install Befehl für Axios

8.2.10 React-PDF-Viewer

Wir haben für das Rendern der Lebensläufe einen Web-Worker verwendet, welcher diese zeitaufwendige Aufgabe verarbeitet, sowie eine Viewer – Komponente.

Bevor die Elemente Worker und Viewer verwendet werden können, müssen sie zuvor importiert werden. Nur so, wie in Listing 2 Import Viewer, Worker und CSS zu sehen, können diese in dem Child Component verwendet werden.

```
import {Viewer,Worker} from "@react-pdf-viewer/core";
import '@react-pdf-viewer/core/lib/styles/index.css';
```

Listing 2 Import Viewer, Worker und CSS

Dieser Web-Worker benötigt, wie in Listing 3 Web-Worker ersichtlich, einen Parameter, den sogenannten „workerUrl“, über den er geladen wird.

```
<Worker workerUrl="https://unpkg.com/pdfjs-dist@3.11.174/build/pdf.worker.min.js" >

</Worker>
```

Listing 3 Web-Worker

Die Viewer - Komponente, wie in der Listing 4 Viewer Komponente anschaulich gezeigt, wird dem Parameter „fileUrl“ der Lebenslauf zugewiesen, welcher angezeigt werden soll**.** [20]

In unserem Fall bekommen wir den Filenamen als URL-Parameter von der Overview an die Edit Seite, so wird im CVs Verzeichnis automatisch nach dem richtigen Lebenslauf gesucht. [20]

```
<Viewer fileUrl={`/CVs/${filename}`} />
```

Listing 4 Viewer Komponente

8.2.11 Apache PDFBox

Abbildung 25 Apache PDFBox Logo

Das Plugin namens „Apache PDFBox“ ist Open-Source und dient dazu, PDF-Dateien mit Hilfe von Code im Backend einlesen zu können.

9 IMPLEMENTIERUNG

9.1 DATENBANK

Die Datenbank hat im Produkt datadive die Aufgabe, relevante Informationen über die Bewerber, wie beispielsweise deren Kenntnisse, bisherige Anstellung, bzw. die Ausbildung und so weiter zu speichern. Die gespeicherten Daten sind ebenso die Grundlage für die Analysen und für Statistiken im Frontend.

Es ist eine Vorgabe des Auftraggebers gewesen, das Datenbank Management System PostgreSQL zu verwenden.

9.1.1 Container

Darüber hinaus war es der Wunsch von unserem Auftraggeber, die Datenbank in einem Docker Container zu halten. Dadurch wird die Flexibilität der Datenerhaltung erhöht. Auch die bereits erklärte Benutzeroberfläche pgAdmin ist in Form eines Docker Containers vorzufinden.

9.1.2 Datenmodell

Das Datenmodell für datadive gleicht einem Star-Schema. Hierbei stellt die Tabelle **person** die Faktentabelle dar. Da die Firma Dynatrace international vertreten ist und deshalb die Hauptsprache Englisch ist, wird das Datenmodell ebenfalls in dieser Sprache gehalten.

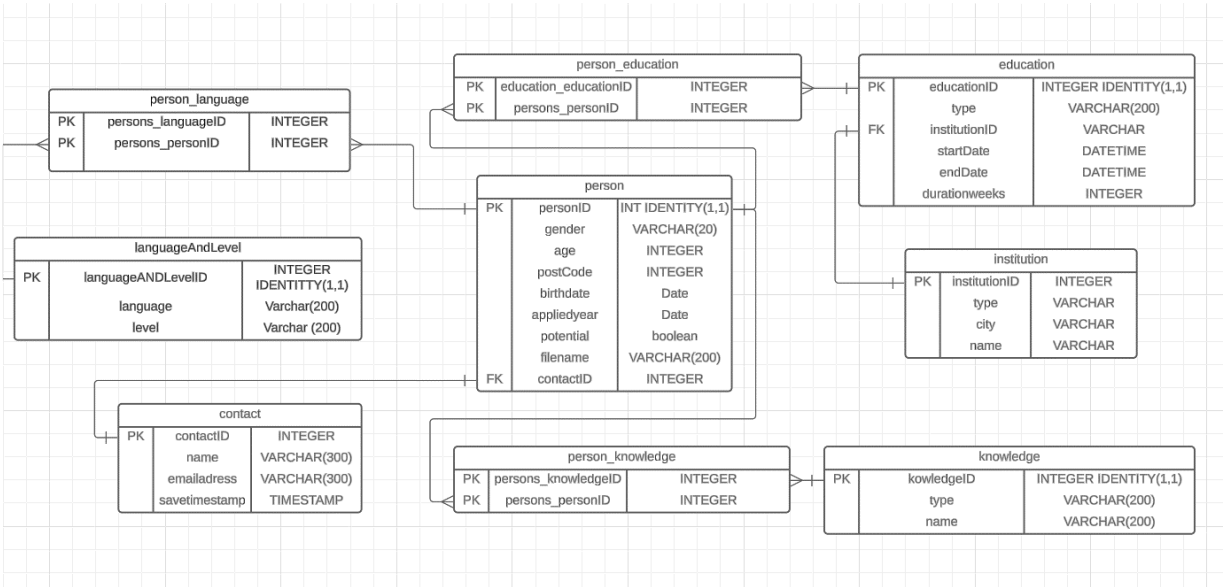


Abbildung 26 Datenmodell datadive

9.1.2.1 Entität person

Zu einer Person gehört neben den typischen Eigenschaften, wie Alter, Geschlecht und so weiter auch das Attribut **appliedYear**, also das Bewerbungsjahr der jeweiligen Person. So können später nicht nur Statistiken über die Bewerberzahlen jahresübergreifend verglichen werden, sondern auch ob der Ehrgeiz des Bewerbers über Jahre hinweg ausschlaggebend war.

Zusätzlich gibt es noch das Attribut **potential**. Das Potential sollte jene Personen markieren, die bereits ein Arbeitsverhältnis mit der Firma gehabt haben. Diese Funktion wird in der Praxis sehr wahrscheinlich bei Praktika und Ferialjobs ihre Verwendung finden. Das Attribut **filename** speichert, wie der Name bereits verrät, den Dateinamen des Lebenslaufes der Person, damit dieser schnell zugeordnet werden kann. Der filename besitzt ein **UNIQUE CONSTRAINT**. Dadurch ist es möglich, wenn zusätzlich neue Lebenslaufdateien automatisiert eingefügt werden, die bisher eingelesenen Dateien trotzdem in dem Ordner zu lassen.

9.1.2.2 Entität contact

Eine Person hat immer genau einen Kontakt angegeben, deshalb gibt es zwischen den Tabellen person und **contact** eine **1:1-Beziehung**. Da der Kontakt aus Datenschutzgründen maximal einen Monat lang gesichert werden darf, gibt es in dieser Entität zusätzlich ein Attribut namens **savetimestamp**. Dieses Attribut speichert den Timestamp, an dem der Tabelle ein neues Tupel hinzugefügt wird, um nach einem Monat den Kontakt automatisiert auf „No Entry“ setzen zu können.

Da die gleichen Datenschutzregelungen auch für den Namen einer Person gelten, ist der Name eines Bewerbers nur in dieser Entität aufzufinden.

Im Backend befindet sich eine Methode mit dem Namen **checkContacts**. Diese wird in der API gemeinsam mit der Methode, die die Daten aus der Datenbank lädt, aufgerufen. So wird jedes Mal, wenn die Informationen aus der Datenbank kommen, zuvor eine Überprüfung der Kontaktdaten durchgeführt. Dabei wird der Timestamp, der zusätzlich zum Datensatz gespeichert worden ist, mit dem des aktuellen Tages überprüft. Wenn die Differenz der Monate (**MONTHS.between**) größer als Eins ist, werden die Emailadresse und der Name auf „No Entry“ gesetzt.
Für das Berechnen der Monate aus den beiden Zeitobjekten wird die Bibliothek Chronounit verwendet. Um die Monate aus dem Timestamp errechnen zu können, gibt es eine Methode mit dem Namen MONTHS(). In dieser werden die Sekunden**,** die ein Jahr hat, durch 12 dividiert.

9.1.2.3 Entität knowledge

In der Relation **knowledge** sind alle Kenntnisse, die eine Person haben kann, gesichert. Durch das Attribut **type** kann festgestellt werden, ob es sich um einen Hard- oder Softskill handelt.

Da ein Bewerber mehrere Kenntnisse haben kann und eine Kenntnis auch mehreren Bewerbern zugeordnet werden kann, liegt hier zwischen den Tabellen eine **M:N-Beziehung** vor. Diese wird durch eine Assoziativtabelle **person_knowledge** gelöst.

9.1.2.4 Entität education

Für die Firma ist es außerdem auch interessant, welche Ausbildung der Bewerber hat. Deshalb gibt es die Tabelle **education**. In dieser Relation werden alle Arten an Berufserfahrung, wie Praktika, bisherige Arbeitsstellen, etc. und auch die Ausbildung, wie Universität, Schulabschluss, etc. festgehalten. Das Attribut **durationInWeeks** ist vor allem dazu da, die Dauer von Praktika oder Ferialjobs angeben zu können.

Ähnlich wie bei der Entität knowledge liegt auch hier eine **M:N-Beziehung** zwischen person und education vor, weil mehrere Personen beispielsweise die gleiche Schule oder den gleichen Arbeitsplatz besucht haben können, aber auch eine Person mehrere Ausbildungen, wie zum Beispiel zuerst eine Schule, dann noch eine Universität und so weiter, absolviert haben kann. Diese Beziehung wird ebenfalls durch eine Assoziativtabelle – **person_education** – gelöst.

9.1.2.5 Entität institution

Damit nun auch die Institution einer Universität, Schule oder einer Arbeitsstelle gespeichert werden kann, gibt es noch die Tabelle **institution**. Diese Information wird später vor allem für Analysezwecke verwendet.

Da es nicht notwendig ist, die Education ganz genau zu persistieren – beispielsweise nur der Name der Universität und die Stadt bzw. der Ort, nicht aber der genaue Studiengang bzw. die genaue Adresse – reicht hierfür eine **1:1-Beziehung**.

9.2 BACKEND

Im Backend der Diplomarbeit ist Java als Programmiersprache verwendet worden. Die Entscheidung bei der Wahl der Programmiersprache ist auf Java gefallen, weil wir nicht nur in der Schule Java im Unterrichtsgegenstand Programmieren und Softwareengineering verwenden, sondern auch der Auftraggeber – die Firma Dynatrace – vorwiegend mit Java arbeitet. Somit sind Fragen sowie Probleme und Hindernisse leicht in Zusammenarbeit mit dem Auftraggeber gelöst worden.

Außerdem ist es für die Firma Dynatrace einfacher, mit unserem Produkt zu arbeiten, falls dieser Wunsch zukünftig auftreten sollte. Das Backend von datadive besteht aus einer Java Spring Boot Applikation. Durch das Framework wird nicht nur die Konfiguration, sondern auch der Aufbau vereinfacht.
Der Vorteil von Java Spring Boot im Vergleich zu einer herkömmlichen Java Applikation ist, dass viele Einstellungen bereits automatisch vorgenommen werden, ohne dass diese explizit vom Entwickler des Programms erwähnt werden müssen. Java Spring Boot macht also die Entwicklung im Gesamten einfacher und schneller, weil viele komplexe Konfigurationsdetails bereits vorkonfiguriert sind. Dadurch kann der Entwickler auch Zeit sparen.

Ebenfalls ist der http-Server (Jetty oder Tomcat) bereits integriert und eingebettet, auch die Verbindung zur Datenbank durch die **application.properties-Datei** ist einfach dargestellt.

[3]

Bei dem Produkt datadive wird der standardmäßige TomCat Server verwendet. Dabei ist es auch so, dass bereits viele Starter-Pakete von Spring Boot einige Tomcat-Abhängigkeiten enthalten.

9.2.1 Projektstruktur

Beim Backend handelt es sich um eine hexagonale Architektur.

9.2.1.1 Hexagonale Architektur

Bevor die Anwendung dieses Konzeptes genauer beschrieben wird, folgt nun zuerst eine kurze Einführung in die hexagonale Architektur.

Durch die hexagonale Struktur, wird ein Projekt in verschiedene Komponenten unterteil, die lose und gekoppelt sind, und darüber hinaus auch untereinander ausgetauscht werden können. Diese nennt man auch **Adapter**.

Der Zweck der Verwendung einer solchen Architektur liegt meist darin, dass die Software dadurch in klar unterschiedliche Schichten getrennt werden kann.

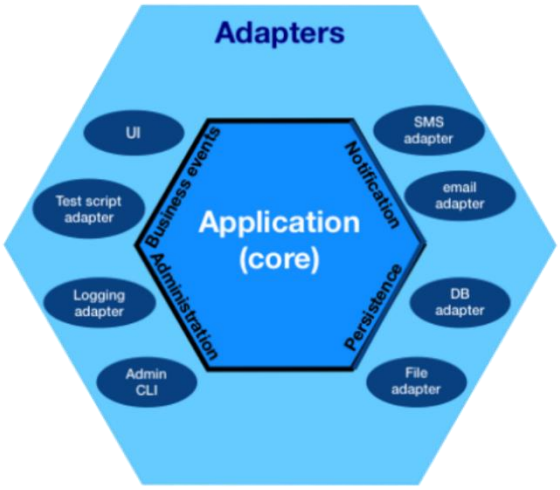


Abbildung 27 hexagonale Architektur

Resultierend ergeben sich wie in Abbildung 27 hexagonale Architektur ersichtlich, der **Anwendungskern (Core),** die **Ports** und die verschiedenen **Adapter.**

Der Kern ist jener Teil, welcher die Logik beinhaltet. Die Ports stellen die Schnittstellen zwischen dem Anwendungskern und der Komponenten dar. Diese treten praktisch beispielsweise als Interfaces auf. Die bereits erwähnten Adapter sind die Implementierungen dieser Interfaces. Der zentrale Kern ist von den verschiedenen Ports umgeben. [21]

Das Spring Boot Backend ist hierarchisch in insgesamt acht Modulen vorzufinden. Dabei sind diese Module logisch voneinander getrennt, wobei sich in jedem von diesen Modulen eigene Klassen mit spezifischen Aufgaben befinden.

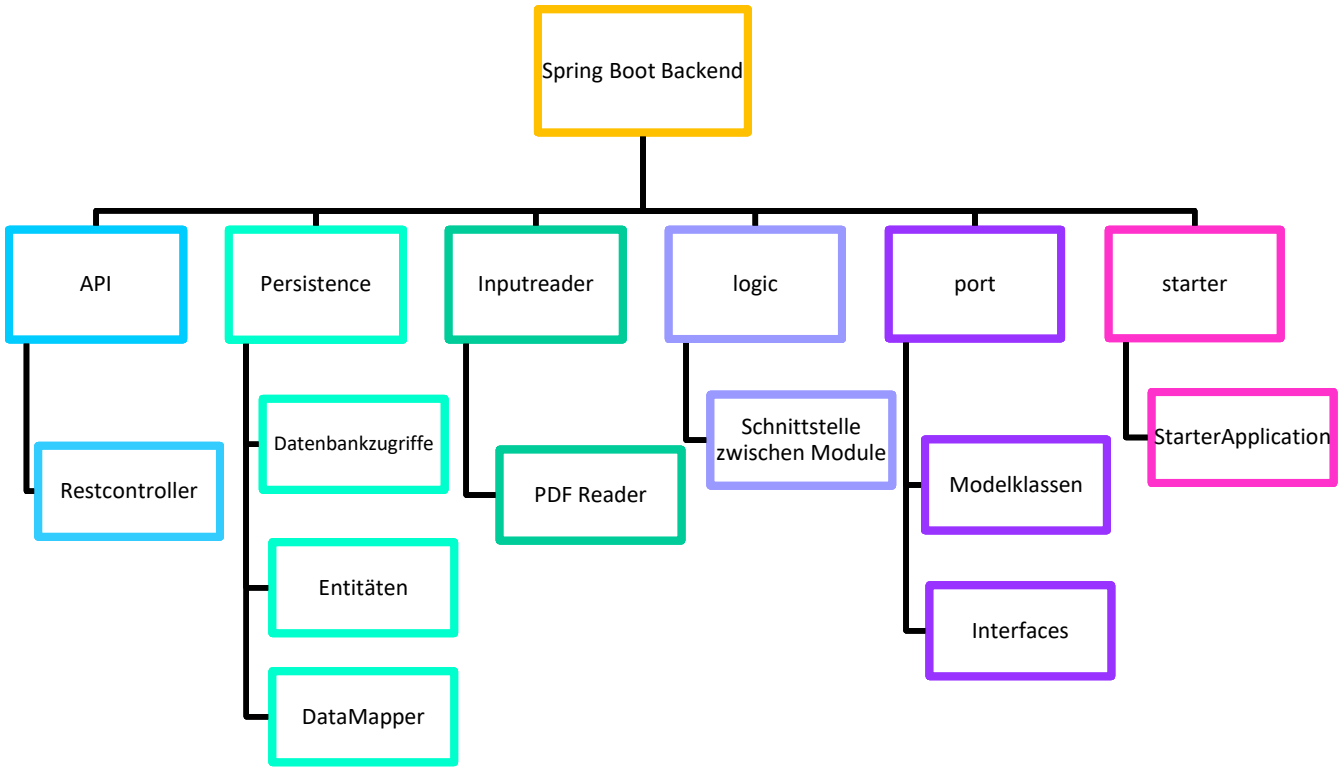


Tabelle 1 Projektstruktur Backend

Im Modul **Port** befinden sich die Interfaces, in denen die Logik der Datenbank, der Logic und der API vordefiniert sind.

Die Implementierung von diesen Interfaces ist allerdings in anderen Modulen zu finden. Beispielsweise befindet sich die Implementierung des Interfaces IDatabase in dem Modul Persistence und die Implementierung der Logik im Modul Logic.

Somit befinden sich alle Klassen, die für den Datenbankzugriff benötigt werden – also auch die Entitäten und der DataMapper in dem Modul **Persistence**. Wie die sprechenden Namen bereits erraten lassen, befindet sich im **Inputreader-**Modul die Implementierung für das Einlesen der PDF-Lebenslaufdateien und im Modul **Starter** die Startapplikation des Backends.

Das **API**-Modul ist ausschließlich für die http-Anfragen zuständig. Deshalb ist in einer Methode eines Endpoints lediglich der Aufruf der Methode die, die Logik für den Datenbankzugriff implementiert hat, vorzufinden.

Während in den bisher erwähnten Modulen die jeweilig offensichtlichen Aufgaben implementiert sind, befindet sich im Modul **Logic** die logische Verknüpfung der einzelnen Module. Das bedeutet, dass die API-Aufrufe, die etwas mit der Datenbank zu tun haben, zuerst über das Logic Modul und dann zur Datenbank gelangen. Erst dort geschieht der eigentliche Datenbankzugriff bzw. die Verwendung der Objekte aus der Datenbank.

Wenn ein http-Request erfolgt und somit ein Endpunkt der API anspringt, kommuniziert diese über das Logic-Modul mit der Datenbank.

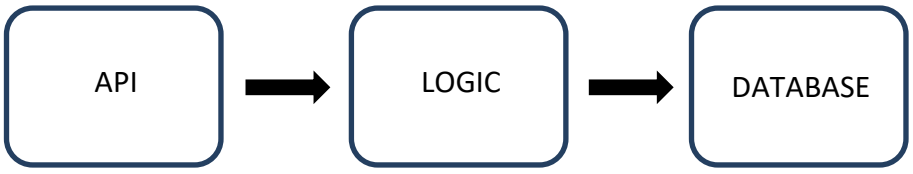


Tabelle 2 API-Datenbank Logik

In folgender Tabelle 3 praktischer Ablauf HTTP-Request ist nochmals veranschaulicht, wie der Ablauf des http-Requests in der Praxis aussieht.

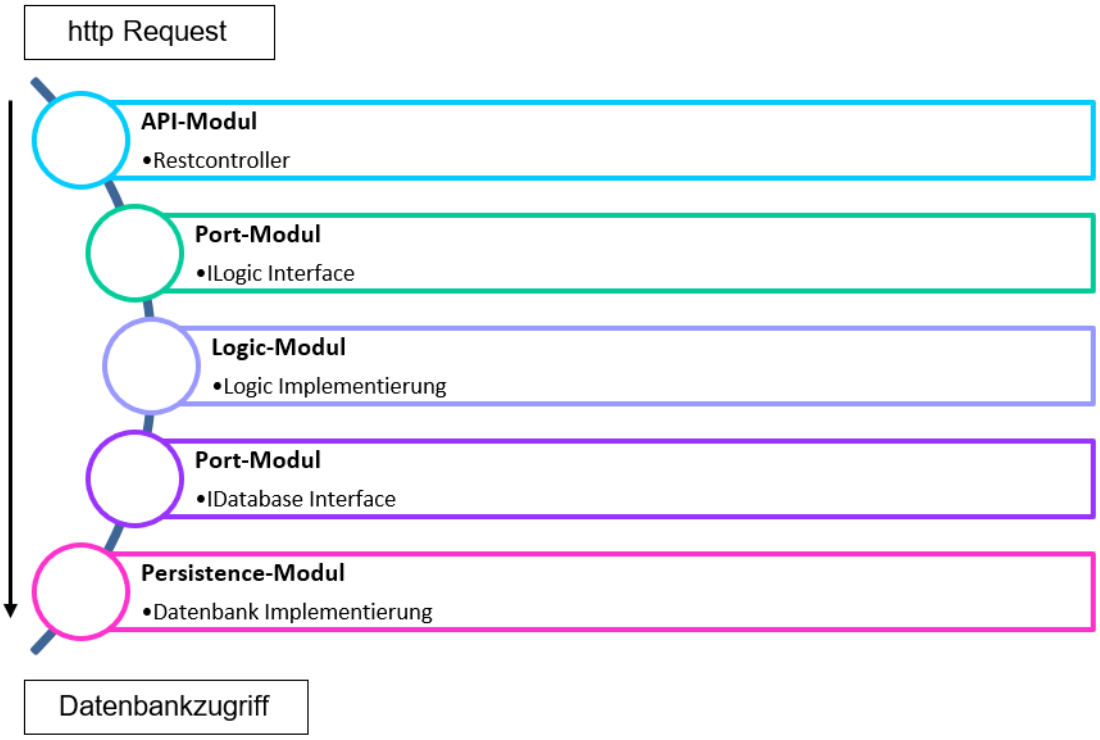


Tabelle 3 praktischer Ablauf HTTP-Request

Das Backend von datadive ist in einer hexagonalen Projektstruktur vorzufinden, weil sie viele Vorteile in Bezug auf die Flexibilität und Erweiterbarkeit bietet.

Beispielsweise kann, wenn der Wunsch, eine andere Datenbank verwenden zu wollen, vorliegt, aufgrund der Interfaces die Implementierung schnell und einfach ausgetauscht werden. Dabei ist ein zusätzlicher positiver Aspekt, dass die bisherige Implementierung der Datenbank nicht verändert werden muss, sondern die Methoden des Interfaces neu implementiert werden können. Falls sich in Zukunft die bisherige Implementierung doch als besser erweisen sollte, ist es möglich, wieder die alte Version der Implementierung zu verwenden.

Dadurch wird der schnelle Austausch, die Veränderung sowie die Flexibilität des Backendes gewährleistet.

Um eine korrekte Kommunikation der Module untereinander zu ermöglichen, ist es notwendig, in der pom.xml (Maven Konfigurationsdatei) eine Abhängigkeit mit dem jeweiligen Modulen anzugeben.

Wichtig dabei zu beachten ist, dass die Module nicht gegenseitige Dependencies haben dürfen, das bedeutet, dass beispielsweise das Persistence-Modul in der pom.xml von Port angeführt sein kann, nicht mehr aber das Port-Modul in der Konfigurationsdatei von dem Modul Logic.

Ist dies der Fall, kann es zu einer sogenannten **Cycle Detection** kommen. Das bedeutet, dass die Abhängigkeit in einen endlosen Kreis geraten würde, da das Eine eine Dependency von dem Anderen hat.

Das Modul **Starter** ist das einzige Modul, das alle Artefakte der anderen Module kennt, es ist auch das einzige Modul, das in keinem anderen Modul verwendet wird. Die StarterApplication ist die jeweilige Klasse im Starter-Modul, wodurch die gesamte Applikation gestartet wird.

Zum Beispiel sieht eine solche Verkettung der Module in dem Modul wie folgt aus.

```
<dependency>
    <groupId>com.example</groupId>
    <artifactId>persistence</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <scope>compile</scope>
</dependency>
<dependency>
    <groupId>org.example</groupId>
    <artifactId>inputreader</artifactId>
    <version>1.0-SNAPSHOT</version>
    <scope>compile</scope>
</dependency>
```

Abbildung 28 Dependencies der Module im Starter pom.xml

9.2.2 Entitäten, Modelklassen und Data Mapping

Anhand des zuvor beschriebenen Datenmodells gibt es nun im Modul persistence – entities - jeweilige Klassen, die die **Entitäten** bilden und das Schema der Datenbank automatisch bei der Ausführung erstellt.
Um sie als Datenbankobjekte erkennen zu können, steht allen von diesen Klassen ein „DB“ zuvor. Der vollständige Name setzt sich aus diesem Kennzeichen und dem Namen der Entität, aus dem Datenbankschema, zusammen – **„DB<Name>“.**

Diese Entitäten sind allerdings nur im Persistence-Modul bekannt. Andere Module, die keine Dependency auf das Persistence Modul besitzen, können somit auch diese Objekte nicht verwenden.

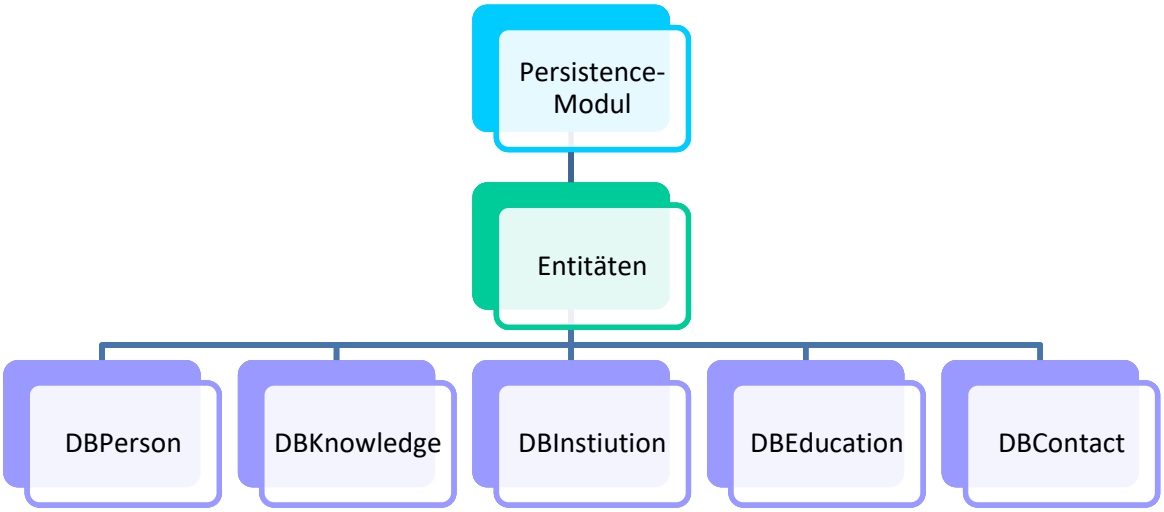


Tabelle 4 Entitäten im Persistence-Modul

Damit diese nun im gesamten Projekt des Backends von datadive verwendet werden können, gibt es im Modul Port die **Modelklassen**. Bei diesen Objekten handelt es sich um jene, die von der API an das Frontend gegeben werden. Um sie von den Entitäten unterscheiden zu können, handelt es sich bei diesen Klassen, bei der Benennung um den gleichen Namen wie ihn die Entitäten aus dem Datenbankschema tragen.

Tabellen, wie DBContact und DBPerson, werden bei den Modelklassen in eine Klasse – Person – zusammengefasst.

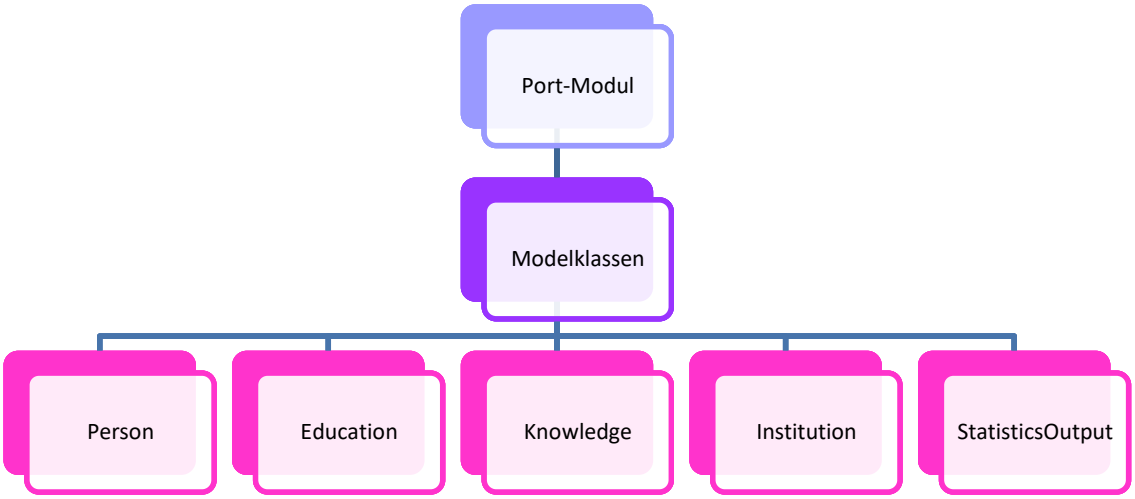


Tabelle 5 Modelklassen Port-Modul

Um nun die Daten aus der Datenbank in die Modelklassen aufbereiten zu können, gibt es im Persistence-Modul noch die Klasse **DataMapper**. Wie der Name bereits vermuten lässt, ist es die Aufgabe dieser, die Daten der Entitäten richtig in die Modelklassen zu konvertieren und umgekehrt.

In dieser Klasse sind jegliche Methoden implementiert, die eine „DBPerson“ in ein „Person“-Element speichern. Hier werden auch diverse Überprüfungen auf null vorgenommen und ansonsten mit – wie zum Beispiel bei der „contact“-Tabelle – „No Entry“ befüllt.

9.2.3 Persistieren der Daten

In datadive wird die Bibliothek JPA für die Implementierung der Datenbankzugriffe verwendet. Somit sind nicht nur die Klassen als Entitäten mit ihren Attributen annotiert, sondern auch die Informationen werden durch sogenannte JPA-Repositories in die Datenbank gespeichert bzw. aus ihr selektiert.

Jede Entität hat ein eigenes Repository, das ein Interface von der Klasse **JpaRepository<Entität, Typ der ID>** ist. In der Praxis sieht ein Repository der Klasse DBPerson beispielsweise wie folgt aus:

```
IPersonRepository extends JpaRepository<DBPerson,Long>
```

Listing 5 JPA-Repository

Diese Repositories besitzen Methoden wie **.save(), .findAll()** und so weiter. Dadurch ist es nicht notwendig, auf die Datenbank mittels SQL-Queries zugreifen zu müssen.

9.2.3.1 JPA Repositories vs. Entity Manager

Zu den Spring JPA Repositories ist zusätzlich auch der Entity Manager ein beliebtes Werkzeug, um auf die Datenbank zugreifen zu können. Deshalb folgt ein kurzer Vergleich:

Abstratkionsebene

Die Spring-JPA Repositories besitzen eine höhere Abstraktionsebene als der Entity Manager. Dies ermöglicht einen einfacheren Umgang mit den Crud-Abfragen der Datenbank. [22]

Abfragen

JPA-Repository beinhaltet bereits einige vordefinierte Methoden, die einfach verändert und angepasst werden können.

Beim Entity Manager gibt es eine Cirteria-API, von der ebenfalls Methoden verwendet werden können oder auch direkte SQL-Queries geschrieben werden müssen. [22]

Transaktionsverwaltung

Bei den JPA-Repositories gibt es bereits standardmäßig eine Transaktionsverwaltung, welche den Umgang mit den Transaktionen erleichtert.

Beim Entity Manager werden die Transaktionen mittels Methoden und Funktionen wie begin() oder commit() manuell verwendet. [22]

Verwaltung von Entitäten

Durch den Entity Manager wird eine genauere Kontrolle über den Lebenszyklus von den Entitäten, die bereits persistiert sind und auch jener, die abgetrennt sind, gewährleistet. [22]

Leistung

In gewissen Situationen sind JPA-Repositories leistungsfähiger als der Entity Manager, weil die Zwischenspeicherung und Optimierung von allgemeinen Operationen mit der Datenbank gegeben ist. [22]

Aufgrund der einfachen Art und Weise, die Datenbankzugriffe durchzuführen, der Sauberkeit des Codes und den weiteren bisher genannten Aspekten, ist die Wahl für datadive auf die **JPA-Repositories** gefallen.

Der erste Zugriff auf die Datenbank beginnt, wenn die Webapplikation das erste Mal aufgerufen wird. Zu diesem Zeitpunkt werden die Informationen der Bewerber aus der Datenbank selektiert.

Ein weiterer ist es aber, nicht etwa Daten aus den Tabellen herauszuholen, sondern Informationen aus den eingelesenen Lebensläufen zu speichern.

Diese Lebensläufe, die eingelesen werden, befinden sich im PDF-Format und sind in einem Ordner, der im Verzeichnis des Backends abgelegt ist. Hier werden nun die Dateien anhand ihres Namens zuerst mit jenen, die bereits in der Datenbank abgespeichert sind, verglichen, damit Duplikate beim Einlesen der Dateien vermieden werden können. All jene, die noch nicht in der Datenbank hinterlegt sind, werden also nun eingelesen.

Dies funktioniert durch die **Apache PDFBox-Bibliothek**. Die Inhalte der PDF-Dateien werden zeilenweise eingelesen, hierbei wird das Bild allerdings nicht beachtet, da es für spätere Zwecke nicht verwendet wird.

```
try {
    inputStream = new FileInputStream(file);
    document = PDDocument.load(file);
    textStripper = new PDFTextStripper();

    String docText = textStripper.getText(document);
    System.out.println(docText);
```

Listing 6 PDF-Dateien einlesen

Durch die verwendete Java-Bibliothek beschränkt sich das Einlesen der PDF-Dateien – bis auf die Initialisierungen der verwendeten Objekte – wie im *Listing 6 PDF*-Dateien einlesen ersichtlich, auf die wenigen dort angeführten Zeilen.

Zuerst wird das File wie beim herkömmlichen Einlesen der Texte in einen **Fileinputstream** eingelesen. Wodurch sie dann in ein **PDDocument-Objekt** geladen werden. Der **PDFTextStripper** wird dazu verwendet, den Text aus dem PDDocument zu extrahieren. Dieses Objekt macht es möglich, den Text aus einem PDF-Dokument herausfiltern zu können, ohne auf die Formatierung und ähnliches zu achten. Am Ende wird mithilfe der **getText()-Methode** des TextStrippers der Inhalt der Datei in ein String-Element gespeichert, mit dem im Anschluss das Filtern der wichtigen Informationen aus den Lebensläufen folgt. [23]

Wenn nun alle Inhalte der Dateien eingelesen worden sind, beginnt die Durchsuchung der Informationen nach wichtigen Kriterien für den Auftraggeber und anhand eines sogenannten **Patterns** werden diese danach gefiltert.
Das Pattern kann beliebig definiert werden.

```
String schoolPattern = "(HTL|FH|Universität|JKU).*";
```

Listing 7 Schoolpattern

Ein Beispiel für die praktische Verwendung dieser Patterns in datadive ist in Listing 7 Schoolpattern zu sehen. Hierbei handelt es sich um ein sogenanntes schoolPattern, welches nach den Schulen und bzw. oder nach den Universitäten bzw. Fachhochschulen filtert, die die Bewerber bisher besucht haben. Es ist möglich, dieses Pattern auch auf andere Schul- oder Universitätstypen auszuweiten, in datadive wird es derzeit so verwendet, da für die Firma Dynatrace derzeit nur diese wichtig sind.

Wie zu erkennen ist, wird nach HTL-, Fachhochschul-, Universitätsabgängern gesucht, der weitere Name ist hierbei egal. Dies führt dazu, dass Schulen wie „HTL-Perg“ etc. gefunden werden können. Da sich der Sitz der Firma in Hagenberg befindet und in Linz die Johannes-Kepler-Universität, ist auch JKU in dem Pattern mitinbegriffen. Sobald sich aber die Anforderungen der Firma ändern sollten, können diese Vorgaben adaptiert bzw. geändert werden.

Insgesamt gibt es in datadive drei Patterns, für die Education, Knowledge und Institution.

Das Filtern des daraus resultierenden Textes funktioniert wie in dem

Listing 8 Pattern.compile angeführt. Während die Pattern-Klasse definiert, wonach gesucht werden soll, übernimmt die Matcher-Klasse die Suche nach dem angegebenen Pattern. Das schoolPattern aus dem Listing 7 Schoolpattern wird mit der Methode Pattern.compile in ein Objekt der Klasse Pattern umgewandelt.

Diese Compile Methode besitzt zwei Parameter, wobei der erste notwendig ist. Dieser ist das Pattern, wonach gefiltert werden soll. Der zweite Parameter ist ein Flag, welches nicht unbedingt nötig ist. Allerdings können hier Dinge angegeben werden, wie beispielsweise MULTILINE, wie es in datadive für das Education-Pattern verwendet wird. Weiters kann bei der Flag noch definiert werden, ob bei der Suche auf die Groß- und Kleinschreibung geachtet werden soll und so weiter.

[24]

```
Pattern findSchoolPattern = Pattern.compile(schoolPattern);
```

Listing 8 Pattern.compile

```
matcher = findSchoolPattern.matcher(docText);
while (matcher.find()) {
```

Listing 9 Matcher.find

Die Matcher.find Methode sucht im Anschluss nach dem Pattern, das zuvor definiert worden ist. Als Rückgabeparameter kommt hierbei true oder false zurück. Daraus setzt sich das Abbruchkriterium für die While-Schleife zusammen. Solange etwas gefunden worden ist, werden dann mittels Matcher.group() die jeweiligen Attribute, die gefiltert worden sind, in die einzelnen Objekte gespeichert.

Sobald Inhalte aus den Lebensläufen in dieses Muster passen, werden sie den entsprechenden Datenhaltungsobjekten zugewiesen. Diese werden wiederum über den DataMapper zuerst in die DB-Entitätsobjekte konvertiert und danach in der Datenbank gespeichert und festgehalten.

9.2.4 API und Analyseauswertung

Die API funktioniert als Schnittstelle, über welche die Daten aus der Datenbank an das Frontend gesendet werden. Hierbei handelt es sich um gewöhnliche Objekte von der Klasse Person, die angezeigt werden sollen, aber auch um bereits berechnete Analysen für Statistiken.

So gibt es bereits eine Methode, die den Prozentanteil an den verschiedenen Arten der Ausbildung – beispielsweise Schulen, Universitäten und so weiter – berechnet und nur diese Ergebnisse dann an das Frontend sendet.

Für solche Zwecke gibt es speziell im Port-Modul eine Klasse namens **StatisticsOutput**.

Diese Klasse beinhaltet lediglich eine numerische Variable vom Typ int, die die Anzahl oder zum Beispiel auch Prozentzahl anzeigt und eine Stringvariable, welche den Wert des Berechneten angibt. Beispielsweise existiert in datadive eine Statistik über die Geschlechtsverteilung unter den Bewerbern. So beinhaltet diese Klasse beispielsweise **count = 8 und value = männlich/male**. Durch das StatisticsOutput Objekt können diese Werte an das Frontend gegeben werden, wo dies nur mehr angezeigt werden muss. Somit ist es nur mehr die prozentuelle Verteilung, die im Frontend berechnet wird.

Die Klasse der API mit dem Namen **Restcontroller** selbst befindet sich im API-Modul. Hier sind alle Endpunkte definiert.

9.2.5 Dokumentation und Clientgeneration

Das Backend von datadive ist mithilfe von Swagger und OpenApi Specification dokumentiert worden. Durch die Verwendung dieser Frameworks ist es möglich, ganz einfach einen passenden Client für das Backend zu generieren. Dies funktioniert durch die Verwendung des Swagger Editors.

In der Praxis verläuft es so, dass die Dokumentation – die sich im JSON-Format befindet – in die linke Spalte der folgenden Abbildung kopiert wird und im Anschluss über den Button „Generate Client“ die Art von Client generiert wird, welcher benötigt wird. Für datadive wird der **typescript-axios-client verwendet.**

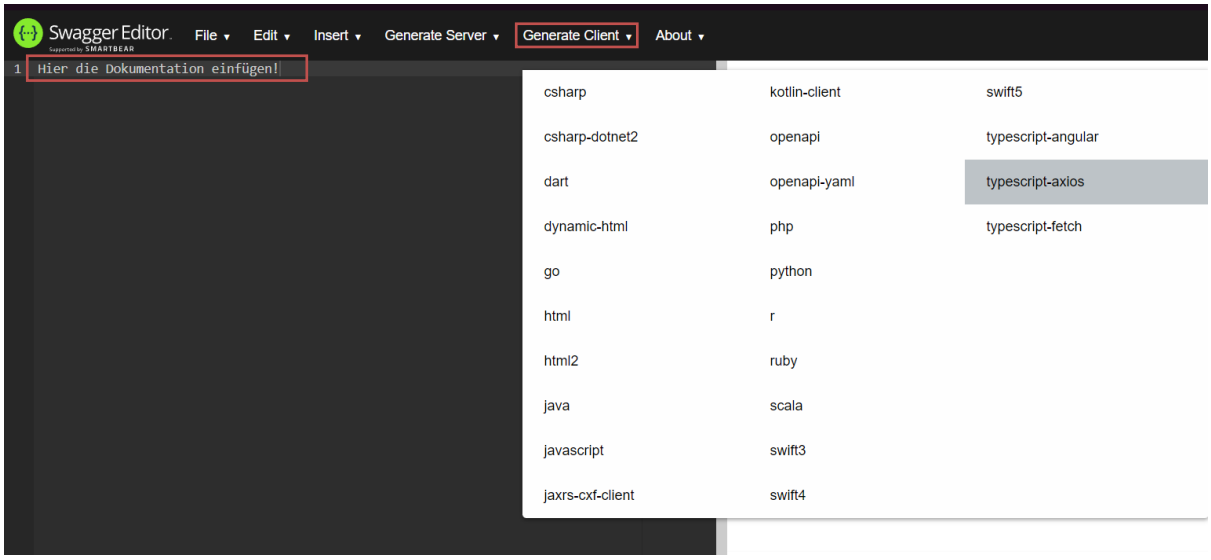


Abbildung 30 Swagger Editor

9.3 FRONTEND

9.3.1 Projektstruktur

Die Projektstruktur ist in wiederverwendbare React-Components, Page- Components, Child- Components und Contexts für die Datenverwaltung sowie CSS-Dateien unterteilt.

Bei unserer Diplomarbeit dreht sich alles um die Seiten Overview, Applications, Files, sowie die verschiedenen Statistiken, die wiederum mit den jeweiligen Child-Components verbunden sind.

In den Child- Components befindet sich die Logik und das Design der jeweiligen Seiten. Um die Daten mittels http – Request vom Backend zu erhalten, gibt es für jede Child- Component einen eigenen Context. Dort werden dann die Endpoints für die benötigte Seite aufgerufen.

Der Ablauf, wie die Seiten mit den jeweiligen Contexts und den dazugehörigen Komponent verbunden sind, wird in der Tabelle 4 Projektaufbau veranschaulicht.

Beim Starten der Applikation wird zuerst die **App.tsx** aufgerufen, dort befindet sich das Routing. Die Routen sind mit den jeweiligen Seiten, die in der Webapplikation enthalten sind, hinterlegt. In den Seiten werden die Contexts aufgerufen, in welchen die API-Zugriffe erfolgen und erst dann werden die Child- Components der Seiten aufgerufen, in denen die Implementierung der Seiten erfolgt.

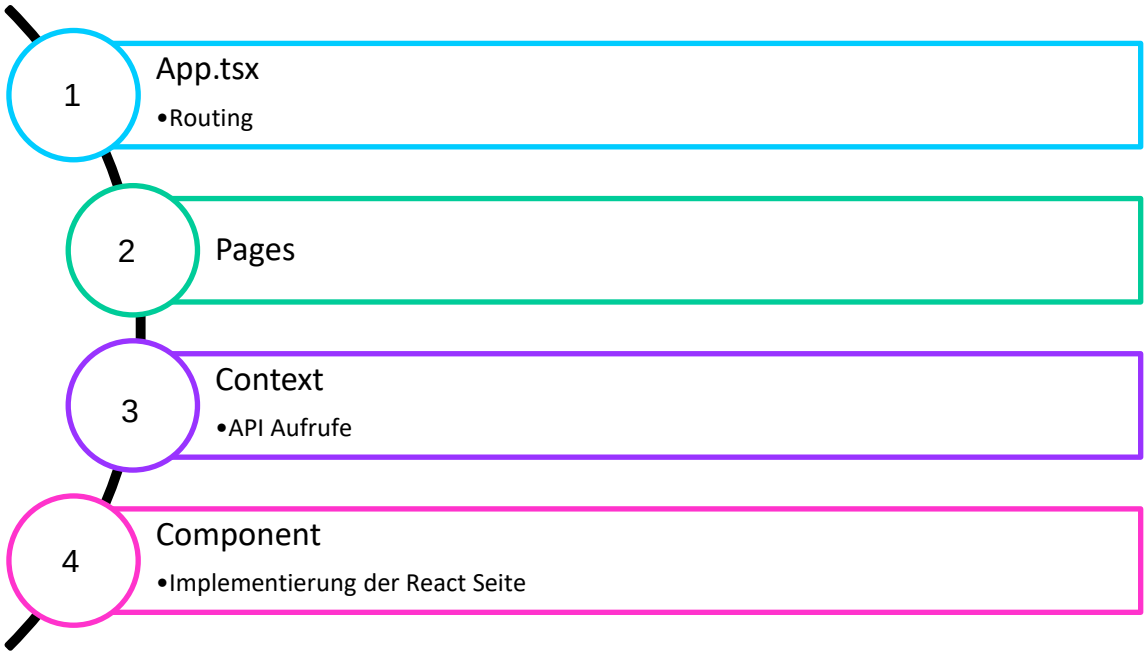


Tabelle 4 Projektaufbau

9.3.2 Routing

Durch den Einsatz des React Routers wird das gesamte Dashboard zu einer Single-Page-Application. Um zwischen den Ansichten navigieren zu können, wird der React Router benötigt.

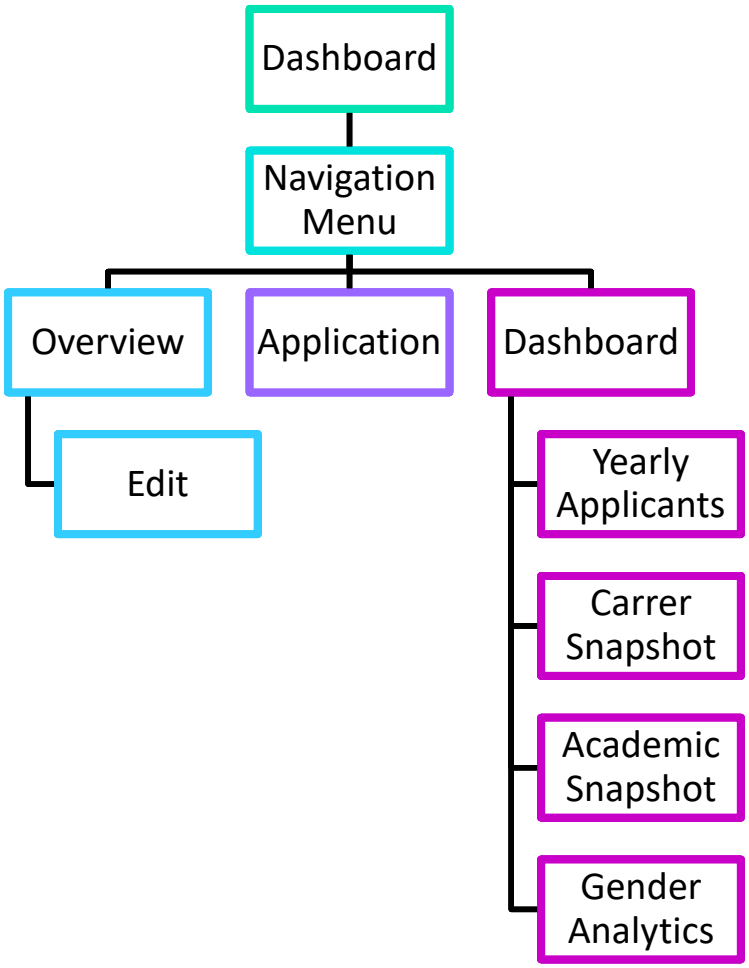


Tabelle 5 Routing

Um zwischen den Ansichten navigieren zu können, werden in unserer Navigationsleiste Links mit einer URL definiert. Diese werden dann in der Adresszeile des Browsers gesetzt.

```
70                  <Nav className="me-auto">
71                      <Nav.Link as={Link} to={""} >  Overview </Nav.Link>
72                      <Nav.Link as={Link} to={"/applications"} >Applications</Nav.Link>
73                      <NavDropdown title="Dashboard" id="basic-nav-dropdown">
74                          <NavDropdown.Item as={Link} to={"/statisticsSchool"}>Yearly Applicants</NavDropdown.Item>
75                          <NavDropdown.Item as={Link} to={"/statisticsEducation"}>Career Snapshot</NavDropdown.Item>
76                          <NavDropdown.Item as={Link} to={"/statisticsKnowledge"}>Knowledge Snapshot</NavDropdown.Item>
77                          <NavDropdown.Item as={Link} to={"/statisticsGirlsBoys"}>Gender Analytics</NavDropdown.Item>
78                      </NavDropdown>
79                  </Nav>
```

Listing 10 Routing Links

Um mit der URL auf die richtige Seite navigieren zu können, sind in der App.tsx Routen definiert, welche mit einem Element hinterlegt sind. Damit das React Routing verwendet werden kann, muss die Anwendung von <BrowserRouter> Tag umschlossen sein.

In einem <Route> Tag wird auf jene Ansicht verwiesen, die angezeigt werden soll, wenn die übereinstimmende URL durch das „Link“ – Element aufgerufen wird. Die <Route> Tags sind Elemente von <Routes>.

```
18          <BrowserRouter>
19              <NavBar/>
20              <Routes>
21                  <Route path="" element={<OverView/>}/>
22                  <Route path="/files/:filename" element={<Files/>}/>
23                  <Route path="/applications" element={<Applications/>}/>
24                  <Route path="/statisticsSchool" element={<StatisticsApplicantsPerYear/>}/>
25                  <Route path="/statisticsEducation" element={<StatisticsEducation/>}/>
26                  <Route path="/statisticsKnowledge" element={<StatisticsKnowledge/>}/>
27                  <Route path="/statisticsGirlsBoys" element={<StatisticsGender/>}/>
28              </Routes>
29              <NavBottom/>
30          </BrowserRouter>
```

Listing 11 Route

Wie in Listing 11 Route ersichtlich, sind die Pfade vordefiniert. Wenn man also mittels Link auf diese URL navigiert, wird so die entsprechende Seite aufgerufen, die in Element hinterlegt ist.

9.3.3 Child-Components

In den Components wird die Page aufgebaut. Das bedeutet, dass alle benötigten Daten vom Context geladen und alle HTML und Design Elemente zusammengeführt werden. Für das Design verwenden wir viele verschiedene Frameworks. So können wir die für uns am besten passende Komponenten aus jedem Design Framework für unsere Webapplikation verwenden.

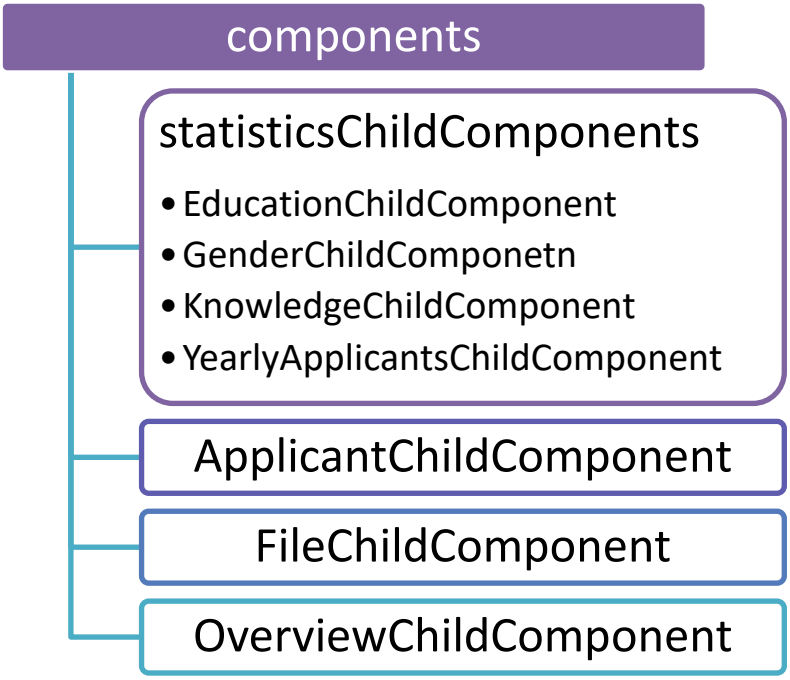


Tabelle 6 Components

Eine Child-Component ist so aufgebaut, dass es eine Funktion gibt, welche programmierten TypeScript Code beinhaltet. Im Rückgabewert ist der HTML-Code definiert, der die Listen und Objekte des darüber implementierten TypeScript Codes verwendet.

Unterhalb der Funktion wird diese mittels „exprot default <funktionname> exportiert, sodass sie von anderen Klassen importiert und verwendet werden kann.

9.3.4 API-Abfragen

In den Context Klassen befinden sich unsere API-Zugriffe. Um unsere Applikation übersichtlicher zu entwickeln, haben wir für jede Component, wie in

Tabelle 7 Context ersichtlich, einen eigenen Context, der für alle API-Abfragen, die diese Component benötigt, zuständig ist.

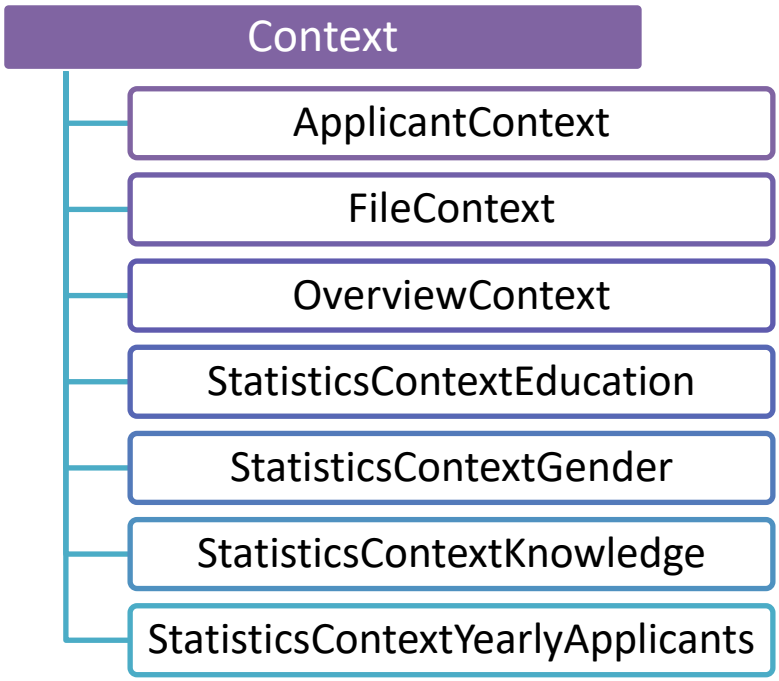


Tabelle 7 Context

9.3.4.1 Implementierung API-Zugriffe

```
23  export function OverViewContextProvider({ children }: { children: React.ReactNode }) {
24      const [data : string , setData : React.Dispatch<React.SetStateA... ] = useState<string>( initialState: '');
25      const [loading : boolean , setLoading : React.Dispatch<React.SetStateA... ] = useState<boolean>( initialState: true);
26
27      useEffect( effect: () : void  => {
28          axios.get<string>( url: 'http://localhost:8081/overview') Promise<AxiosResponse<...>>
29              .then(response : AxiosResponse<string, any>  => {
30                  setData(response.data);
31                  setLoading( value: false);
32
33              }) Promise<void>
34              .catch(error => {
35                  console.log(error);
36              });
37
38      }, deps: []);
39
40      return (
41          <OverViewContext.Provider value={{ data, loading }}>
42              {children}
43          </OverViewContext.Provider>
44      );
45  }
46
```

Listing 12 API-Zugriff

Um zukünftige Änderungen oder Erweiterungen vornehmen zu können haben wir uns in Absprache mit der Firma Dynatrace dazu entschieden Axios für die API-Abfragen zu verwenden.

Bei Axios handelt es sich um einen Promise basierten http-Client für node.js. Serverseitig wird das Modul http angewendet, während im Browser XMLHttpRequests ausgeführt werden. [25]

Ein Promise ist ein Objekt, welches von einer asynchronen Funktion zurückgegeben wird und den derzeitigen Stand der Operation erschließt. [26]

Mittels Axios wird die URL des Endpoints aufgerufen, die Funktion gibt ein Promise zurück welches dann durch „.then“ in das API-Response aufgelöst wird. Mit „**setData()**“ werden die **response.data** in mein data Objekt gespeichert. Außerdem wird das Loading auf false gesetzt, da die Daten von der API erfolgreich geladen worden sind.

Das Data und das Loading Objekt werden dann an den Component übergeben, sodass in der Klasse mit den Daten gearbeitet werden kann.

In unserer Applikation verwenden wir die Methoden GET, POST und PUT.

Vergleichsweise zum GET Request werden beim POST oder PUT die Daten im Body mitübergeben und im Header des Axios Request muss das Format JSON definiert sein, damit das Backend mit den Daten arbeiten kann.

9.3.4.2 Kommunikation zwischen Backend und Webapplikation

Die Kommunikation erfolgt mittels Axios http-Requests. Die Daten werden auf den Seiten aktualisiert, sobald die Seite neu geladen wird. Um die Daten in der Datenbank auf den neuesten Stand zu bringen, gibt es in unserem Produkt eine Möglichkeit mit einem Reload-Button die neu hinzugefügten Lebensläufe im Backend einzulesen.

In der Kommunikation zwischen dem React-Frontend und dem Java-Backend kann es zu einem Problem kommen, da der Client auf einer anderen URL läuft und daher nicht auf bestimmte Endpunkte in der API zugreifen darf.
Bei diesem Problem kommt es zu einem „Cors-Policy“-Fehler. Damit diese behoben werden können, haben wir die Klasse CorsConfig geschrieben. Hier wird definiert, welche Clients auf den Restcontroller auf welche Art und Weise zugreifen dürfen.

9.3.4.3 Cors Policy

Cors steht für **Cross-Origin Resource Sharing** und bestimmt für Client-Webanwendungen, welche in einer Domain geladen werden, wie die Interaktion mit den Ressourcen eines anderen Ursprungs (Domain, Protokoll, Port,…) funktioniert. Beispielsweise die Verwendung der Daten aus dem Backend in der Benutzeroberfläche im Frontend. Damit dies funktioniert, gibt der Server eine spezielle Header in der Antwort zurück. [27]

Zu einem sogenannten „Cors Policy“-Fehler kommt es, wenn der bereits erwähnte Header nicht so zurückgegeben wird wie erwartet. Dann wird dem Browser aus Sicherheitsgründen der Zugriff auf diese Daten verweigert.

Die Fehler sehen in der Praxis meistens wie folgt aus:

- ➔ „No ‚Access-Control-Allow-Origin’ header is present on the requestet ressource.”

[28]

Um nun einen solchen Fehler beheben zu können, benötigt es die CorsConfig-Klasse im Backend. Hier können die Sharing Regeln definiert werden, von welchem Ursprung aus, welche Anfragen an die API gestattet sind. Somit wird eine sichere Kommunikation zwischen Back- und Frontend gewährleistet.

```
@Configuration
public class CorsConfig implements WebMvcConfigurer {
    no usages   LAPTOP-DIGVI66T\Katja *
    @Override
    public void addCorsMappings(CorsRegistry registry) {
        registry.addMapping( pathPattern: "/**")
                .allowedOrigins("*") //  z.B. "http://localhost:3000"
                .allowedMethods("GET", "POST", "PUT", "DELETE", "OPTIONS") // Erlaubte HTTP-Methoden
                .allowedHeaders("*")
                .exposedHeaders("Authorization")
                .allowCredentials(true) // Erlaubt das Senden von Cookies
                .maxAge(3600); // Gültigkeitsdauer
    }
}
```

Listing 13 CorsConfig Klasse

10 ERGEBNIS

10.1 WEBAPPLIKATION

Die Webapplikation besteht aus mehreren Pages, welche über eine Navigationsleiste im Header erreicht, werden können. Außerdem befindet sich im Header auch noch ein Reload-Butten, dieser ermöglicht das neue Laden der Seite. Dabei wird geprüft, ob der Name oder die E-Mail-Adresse der Bewerber nicht länger als ein Monat in der Datenbank gespeichert ist. Überschreiten diese Daten die zeitliche Begrenzung, werden diese aus der Datenbank gelöscht. Es wird auch überprüft, ob neue Lebensläufe im Filesystem abgelegt worden sind, welche dann automatisch im Backend eingelesen und die Daten in die Datenbank gespeichert werden.

Abbildung 31 Navigationsleiste

10.1.1 Overview

Als Hauptseite haben wir eine Übersicht, die alle für die Firma Dynatrace relevanteren Daten anzeigt. Wir haben uns hierbei für eine Tabelle entschieden, in welcher man nach bestimmten Kriterien suchen kann.

Wie in Abbildung 32 Overview ersichtlich handelt es sich bei den Daten um den Dateinamen des Lebenslaufes, das Alter sowie die Education- und Knowledge Type von jedem Bewerber. In jedem Eintrag gibt es einen Edit Button, dieser navigiert auf die Edit Seite. Dort wird der Lebenslauf des jeweiligen Bewerbers angezeigt und die Daten können bearbeitet werden.

Damit die Applikation benutzerfreundlich gestaltet ist, haben wir uns bei der Tabelle für Pagination entschieden, somit kann die Anzahl der angezeigten Bewerber individuell angepasst werden. Mithilfe des Scroll-Panels wird nur innerhalb der Tabelle gescrollt und nicht über die ganze Seite.

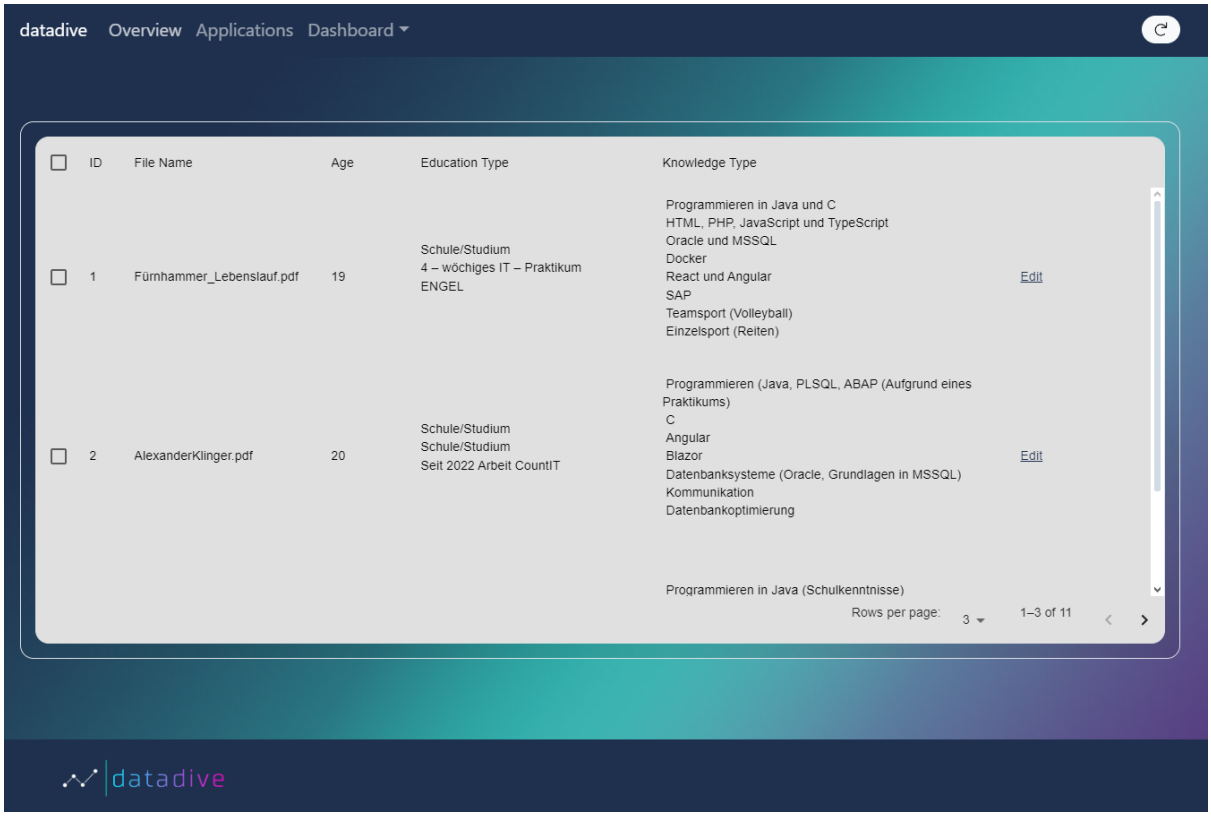


Abbildung 32 Overview

10.1.1.1 Files

Unsere File Page dient dazu, sich einen bestimmten Bewerber, falls diese oder dieser bereits in der Overview überzeugt hat, einzeln anzusehen.

Auf dieser Seite wird der Lebenslauf in PDF angezeigt. Außerdem befindet sich hier auch die Möglichkeit, die Daten noch einmal zu bearbeiten. Denn falls der Bewerber seinen Wohnort wechselt oder sich im darauffolgenden Jahr noch einmal bewirbt, können diese Daten in unserem Formular leicht angepasst werden.

Außerdem ist es nicht möglich, das Geschlecht anhand des Namens, der im Lebenslauf steht, zu erkennen. Darum gibt es die Möglichkeit, das Geschlecht nachträglich über unser Formular hinzuzufügen.

Zusätzlich erleichtert es diese Ansicht dem Recruiting Team bei der Überprüfung der Daten. Denn es kann auf einen Blick verglichen werden, ob die Daten, die in der Datenbank von dem potentiellen Kandidaten gespeichert sind, mit den Daten im Lebenslauf übereinstimmen.

Der „Save Changes“-Button in dem Formular ist dazu da, die Daten in der Datenbank zu aktualisieren. Wenn das Speichern erfolgreich gewesen ist, wird es dem Benutzer mit einer Message „Changes saved successfully!“ angezeigt.

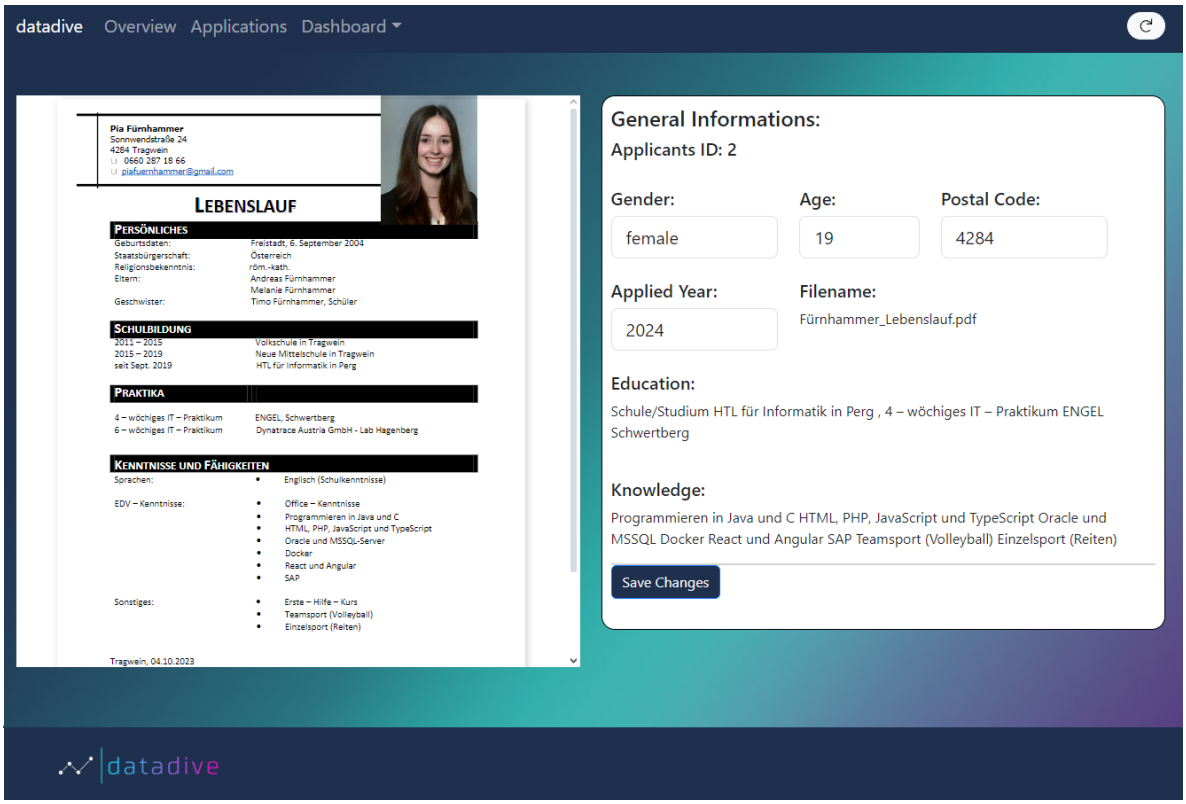


Abbildung 33 Edit Page

10.1.2 Applications

Die Application Seite veranschaulicht alle Daten, welche wir aus den einzelnen Lebensläufen herausgefiltert haben. Um unsere Applikation einheitlich zu gestalten, haben wir uns auch hier wieder für die gleiche Tabelle von Material UI entschieden.

Diese Tabelle ermöglicht uns ebenfalls das Suchen und Filtern nach bestimmten Kriterien. Es ist außerdem möglich, einzelne Attribute auszublenden. Somit können alle Bewerber gleichzeitig in ihren Fähigkeiten oder Ausbildungsstandpunkten verglichen werden. Auf diese Weise wird das Bewerbungsverfahren effizienter und kann schneller gestaltet werden.

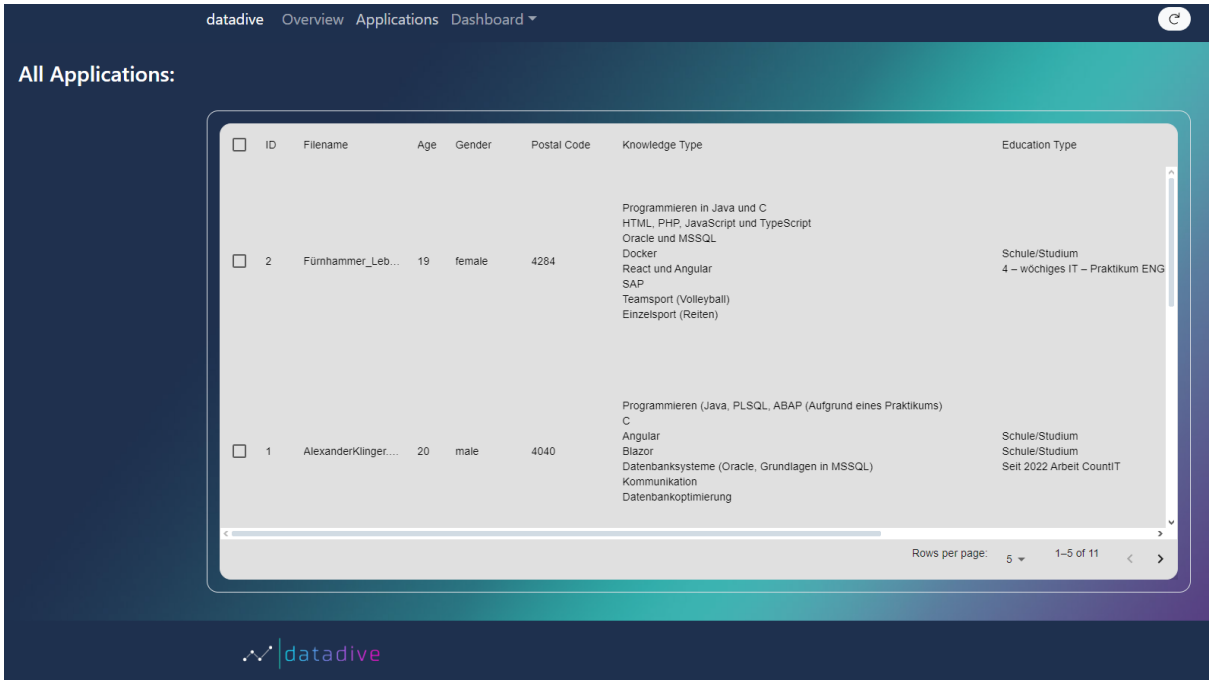


Abbildung 35 Applications

Wie bereits erwähnt, können die einzelnen Spalten in dem Material UI Data Grids individuell ein- und ausgeblendet werden. Das ermöglicht ein noch wirkungsvolleres Filtern der Daten.

Die Filterfunktion, die Material UI zur Verfügung stellt, beinhaltet die Möglichkeit, eine Spalte auszuwählen und eine Value einzugeben, nach welcher gesucht werden soll. Zusätzlich kann mit dem Operator festgelegt werden, ob die eingegebene Value in einem Attribut vorkommen soll, ob es damit starten oder enden oder ob es gleich der eingegebenen Value ist.

10.1.3 Dashboard

Um die gesamten Daten noch besser zu visualisieren, haben wir noch ein Dashboard entwickelt. Gemeinsam mit unserem Projektbetreuer Herrn Kreuzer haben wir uns für die Statistiken Yearly Applicants, Career Snapshot, Knowledge Snapshot und Gender Analytics, wie in Abbildung 36 Dashboard ersichtlich, entschieden.

Die Daten für die einzelnen Statistiken werden im Backend ausgewertet und danach mittels Get-Request an das Frontend gesendet. In den jeweiligen Child Components werden die Daten dann noch auf die benötigten Charts angepasst.

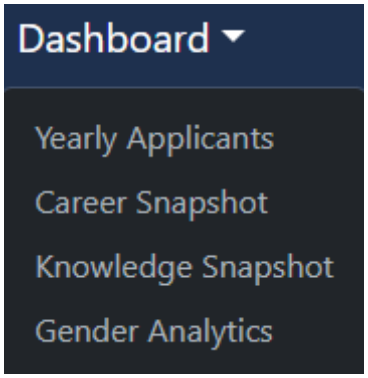


Abbildung 36 Dashboard

10.1.3.1 Yearly Applicants

Unser erstes Dashboard zeigt die jährlichen Bewerber der Firma Dynatrace. Das ist für unseren Auftraggeber von großer Bedeutung, da sie aufgrund unserer Analyse feststellen können, ob es mehr oder weniger Bewerbungen im aktuellen Jahr im Vergleich zum Jahr davor gibt.

Bei der Gestaltung haben wir uns für die Frameworks CoreUI, Google React Charts und React Apex Charts entschieden.

Von CoreUI verwenden wir in diesem Dashboard das Layout sowie die beiden Widgets, die die Bewerbungen im Jahr 2024, sowie die vom Jahr 2023 widerspiegeln. Um dem Gesamten eine persönliche Note zu geben, haben wir uns dafür entschieden, in die rechten oberen Ecken der Widgets das Dynatrace Logo, welches von CoreUI Icons zur Verfügung gestellt wird, zu platzieren.

Damit unser Dashboard lebendiger wirkt, haben wir uns bei den Farben, die in den Charts vorkommen, für Grün- und Blautöne entschieden, da diese Farben auch im Logo der Firma Dynatrace vertreten sind.

Um die Statistiken noch abzurunden, haben wir uns zusätzlich zu den Widgets für ein Säulen- und ein Tortendiagramm von React Apex Charts und Google React Charts

entschieden, welche die Daten noch einmal visualisiert darstellen.

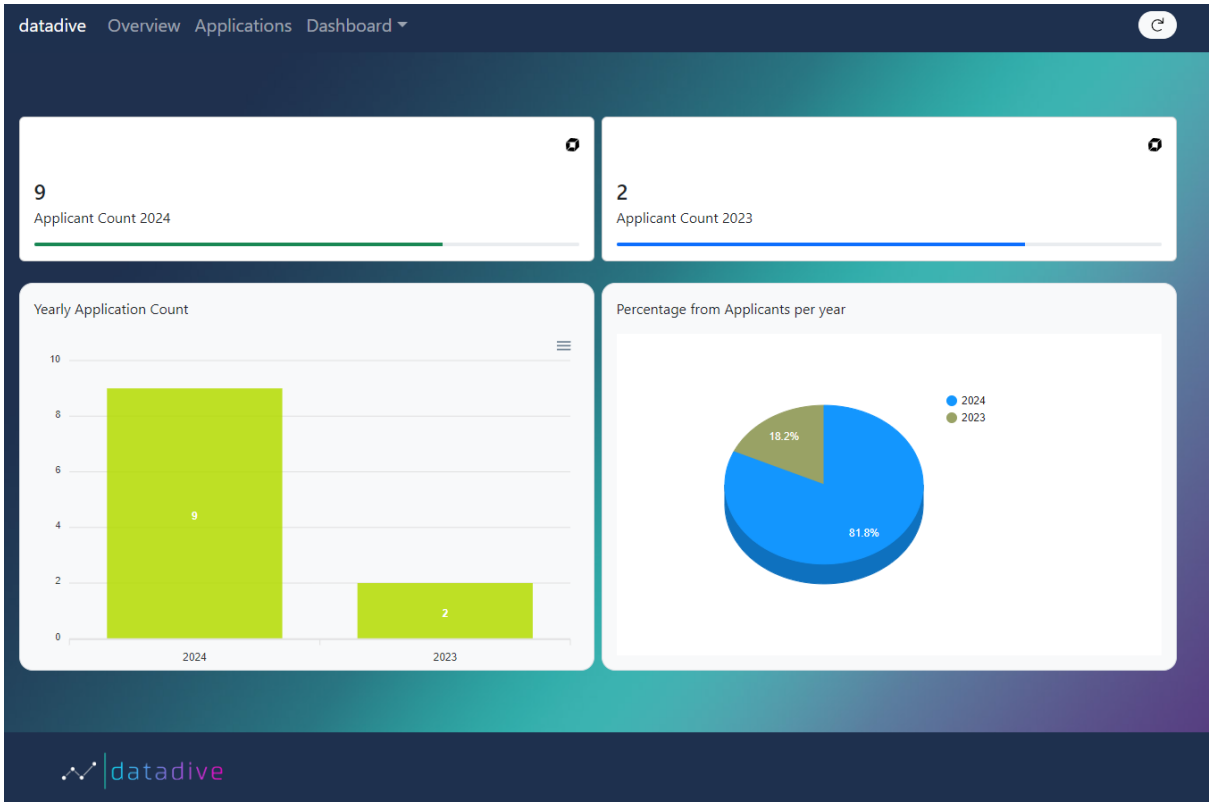


Abbildung 37 Yearly Applicants

10.1.3.2 Career Snapshot

Unter dem Reiter Career Snapshot werden alle schulischen und beruflichen Tätigkeiten in Diagrammen angezeigt. Das ermöglicht es unserem Auftraggeber Rückschlüsse zu bilden, in welchen Bereichen das Marketing am effektivsten gewesen ist.

Um das einheitliche Design der Dashboards beizubehalten, bevorzugten wir auch hier das Säulen- und ein Tortendiagramm von CoreUI und Google React Charts, ebenfalls in den Farbtönen der Dynatrace.

Unser Tortendiagramm stellt den prozentualen Wert dar, wohingegen das Säulendiagramm die tatsächliche Anzahl wiedergibt.

Damit diese analysierten Daten zusätzlich zu den Diagrammen noch einmal wiedergegeben werden, wählten wir eine Tabelle ohne Umrandung von CoreUI. So werden alle Ausbildungen, sowie die Anzahl der Bewerber, die diese Ausbildung absolviert haben, angezeigt.

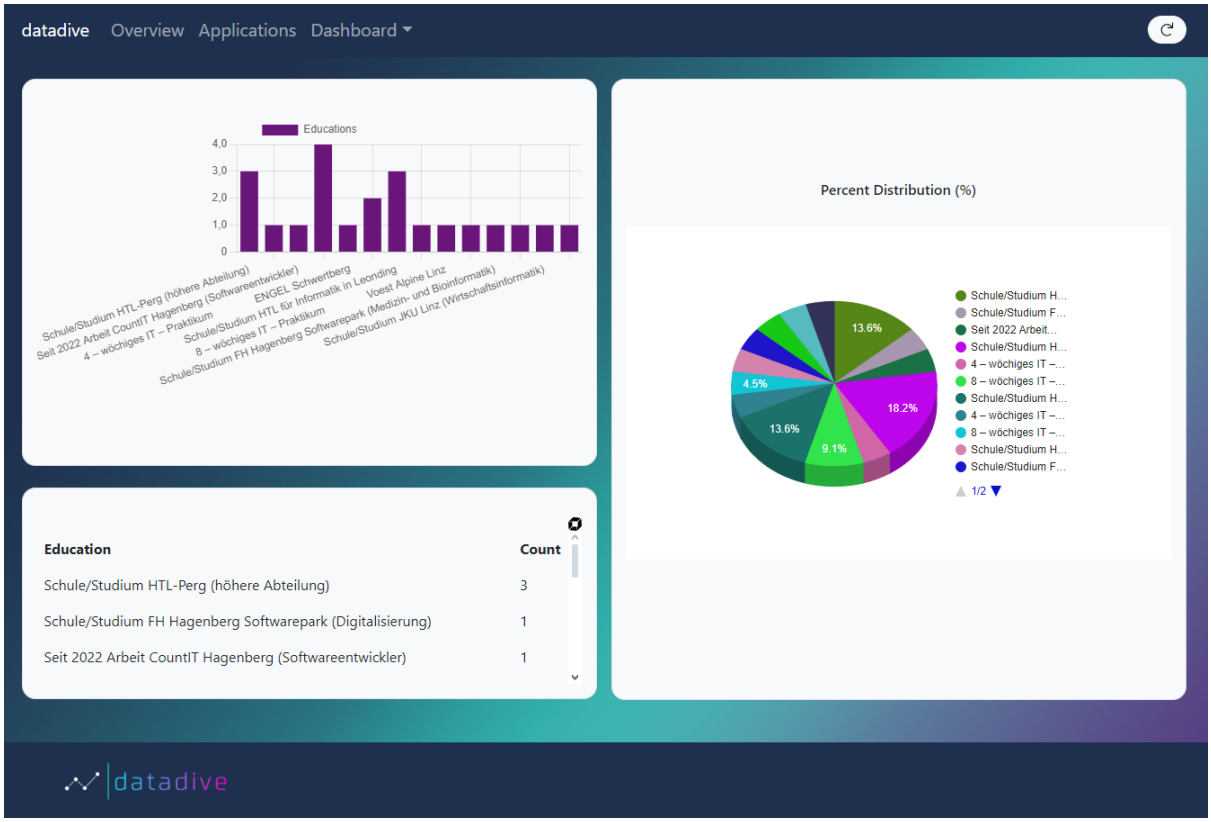


Abbildung 38 Career Snapshot

10.1.3.3 Knowledge Snapshot

Der Reiter Knowledge Snapshot soll alle Soft- und Hardskills der Bewerber visualisieren.

Da es sich bei den Knowledges um eine sehr große Menge an Daten handelt, haben wir uns dazu entschlossen, diese in Bereiche zusammenzufassen. Somit kann der Benutzer auf einen Blick sehen, welche Kenntnisse prozentual am häufigsten vorkommen.

Gemeinsam haben wir uns für folgende Gruppierungen entschieden:

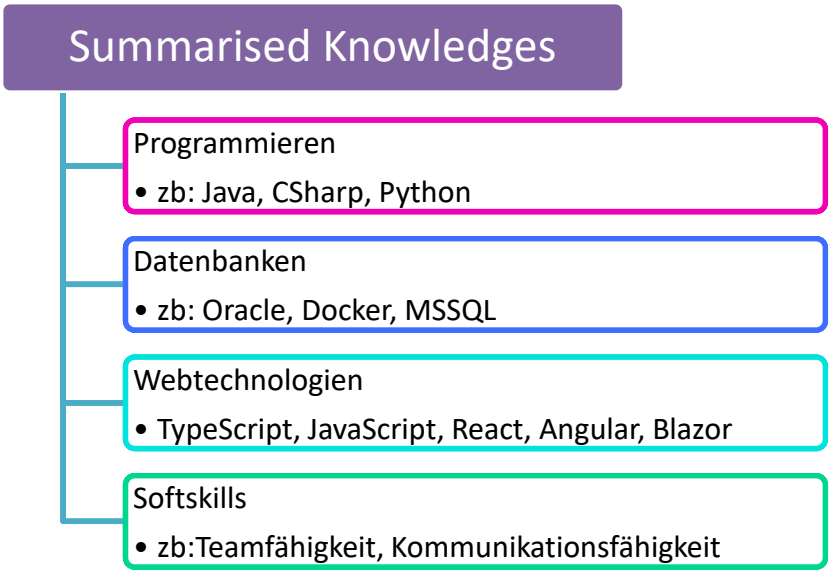


Tabelle 8 Summarised Knowledges

Diese Gruppierungen werden wie bei den Yearly Applicants mit CoreUI Widgets veranschaulicht. Aber im Gegensatz zu diesem Dashboard haben wir uns bei den Widgets für ein zur Gruppierung passendes Icon anstatt des Dynatrace Logos entschieden.

Zusätzlich zu den Widgets haben wir auch hier wieder ein Tortendiagramm, das nun alle Kenntnisse prozentual anzeigt. Wie in Abbildung 39 Knowledge Snapshot ersichtlich, sind wir auch hier den Dynatrace Farben treu geblieben.

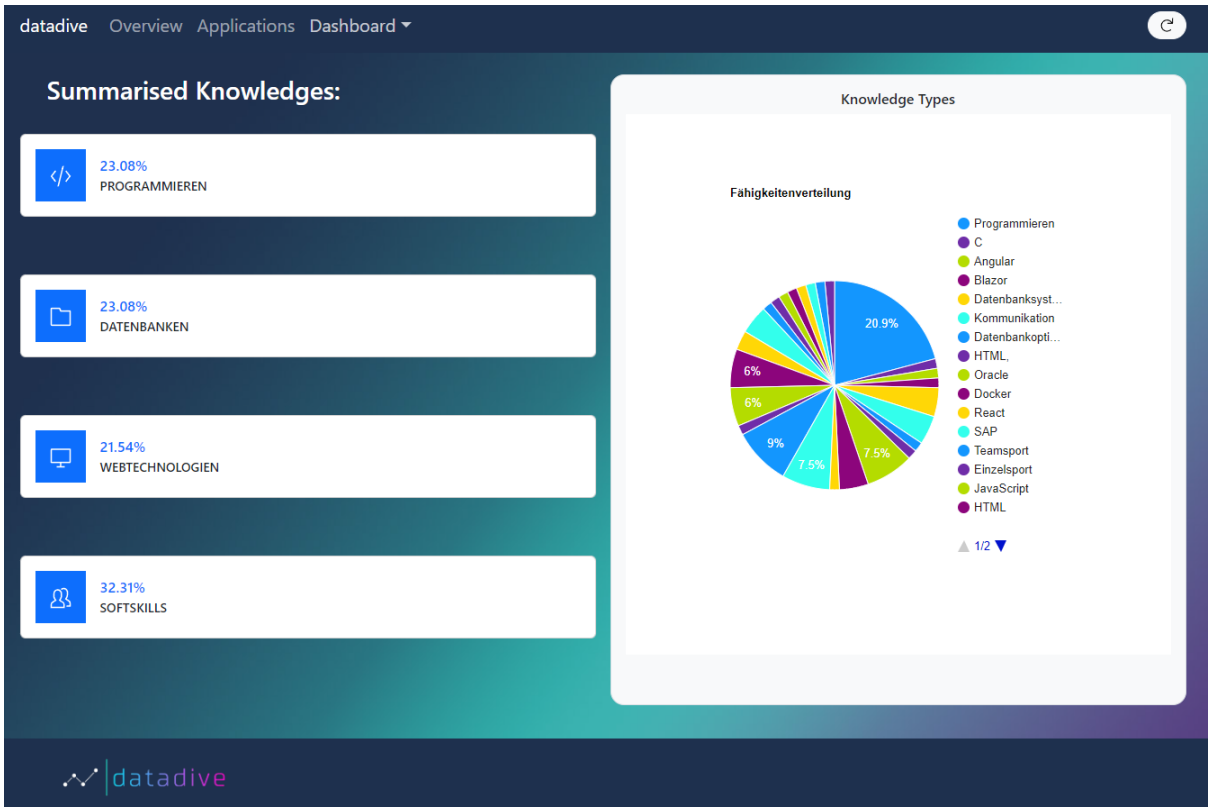


Abbildung 39 Knowledge Snapshot

10.1.3.4 Gender Analytics

Die Firma Dynatrace legt auch Wert auf Diversität bei ihren Mitarbeiter, darum ist im Reiter Gender Analytics genau zu sehen, bei welchem Geschlecht deren Marketing am besten ankommt.

Da der Prozentsatz von Frauen in der Technik im Vergleich zu den Männern immer noch sehr gering ist, bemüht sich die Firma Dynatrace auch die weiblichen Programmierinnen mit ihrem Firmenkonzept anzusprechen.

Wir haben uns bei diesem Dashboard auch wieder für die CoreUI Widgets entschieden, welche die weiblichen, männlichen und diversen Bewerber prozentual wiedergeben. Hier haben wir auch ein dazu passendes Icon gefunden.

Wie Abbildung 40 Gender Analytics zu erkennen ist, haben wir unterhalb mit denselben Farben wie bei den Icons zwei Charts platziert, welche die genderspezifischen Daten noch einmal visualisieren. Zusätzlich zu den Charts und den Widgets haben wir auch hier eine Tabelle bevorzugt, welche die Geschlechter und die genaue Anzahl an Bewerbern mit diesem Geschlecht anzeigt.

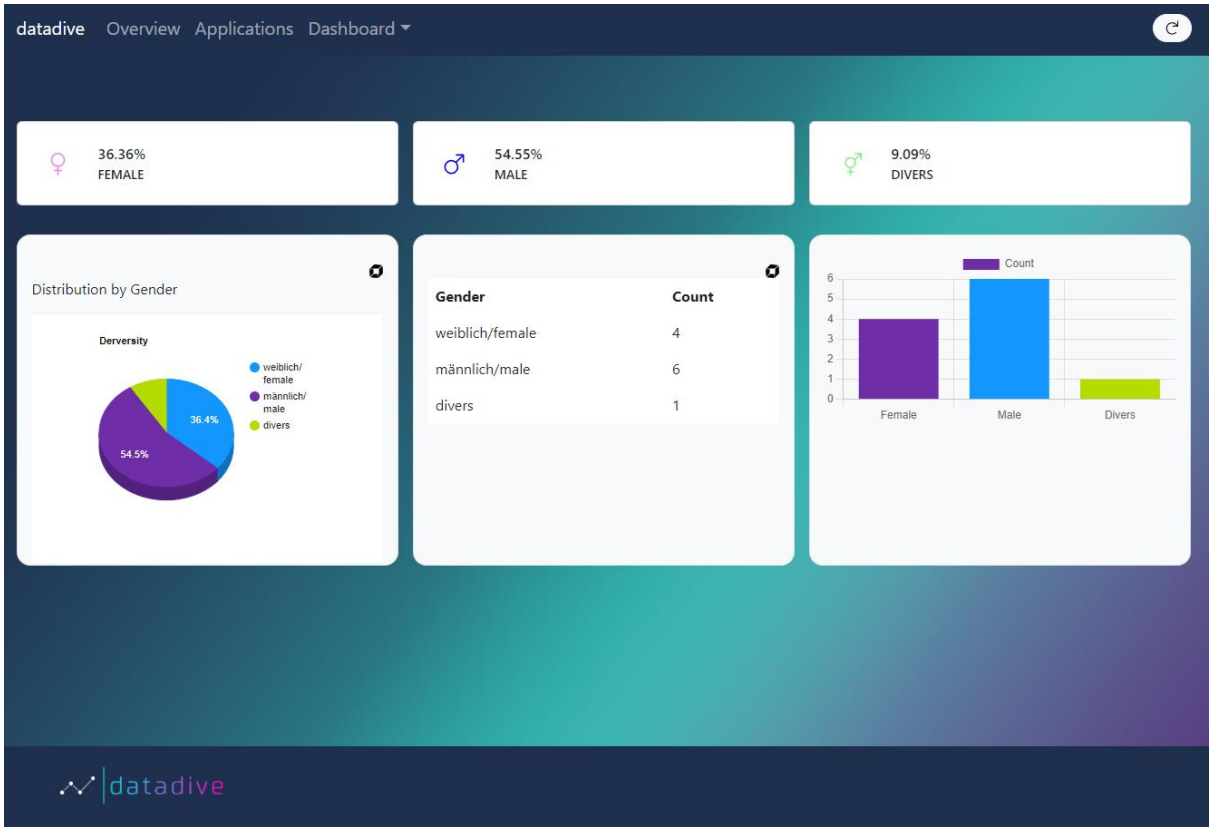


Abbildung 40 Gender Analytics

10.2 Java Spring Boot Backend

Daraus resultierend ist ein Java Sprinboot Backend entstanden, die diverse Endpunkte und Berechnungen der Daten für die Statistiken mitsamt allen Datenbankzugriffen beinhaltet. Lebenslaufdateien im PDF-Format müssen lediglich in einem Ordner im Verzeichnis des Backends abgelegt werden. Durch einen API-Aufruf über einen Button im Frontend werden anschließend die bisher abgespeicherten Informationen, die die Lebenslaufnamen beinhalten, mit den Namen neu hinzugefügter Dateien abgeglichen. All jene, die sich noch nicht in der Datenbank befinden, werden eingelesen und die für den Bewerbungsprozess interessanten Daten werden herausgefiltert und in der Datenbank persistiert.

Durch die hexagonale Struktur, in der sich das Backend befindet, ist die Applikation flexibel und strukturiert gestaltet. Somit hat der Auftraggeber die Möglichkeit, nach Belieben weitere bzw. andere Datenbanken, APIs und so weiter auszutauschen oder hinzuzufügen. Weiters kann die Firma bei anderen Anforderungen oder Gegebenheiten das Pattern, wonach die Informationen aus den Lebensläufen extrahiert werden, abändern und ergänzen.

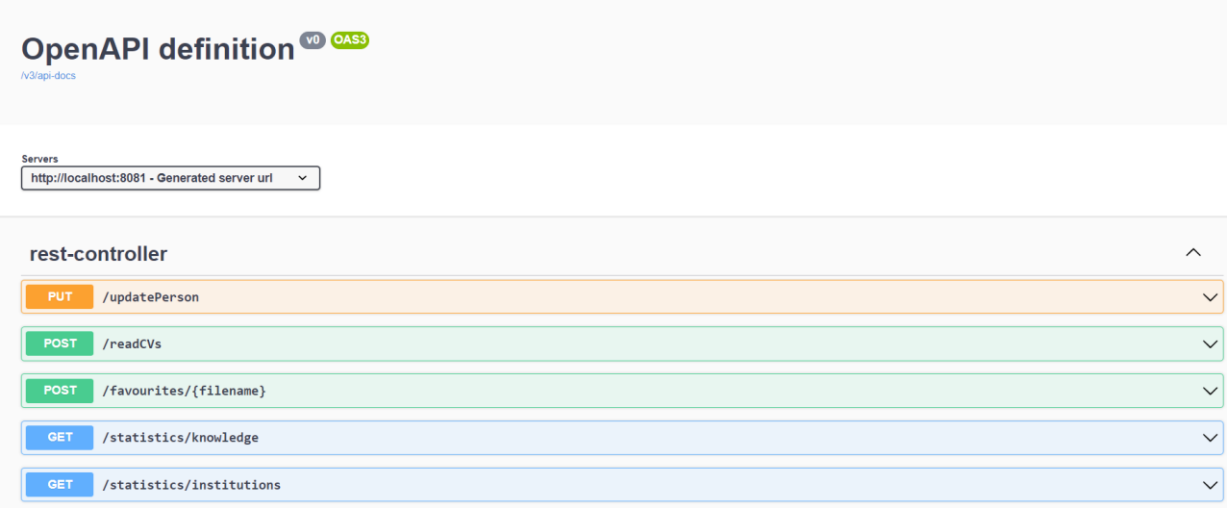


Abbildung 41 Endpunkte API

10.3 PostgreSQL Datenbank

Die PostreSQL Datenbank besteht aus einem an das Starschema angelehnten Modell, welches die Daten hält. Über die im Backend verwendete JPA-Bibliothek, ist es möglich, dass ohne SQL auf die Datenbank und deren Inhalt zugegriffen werden kann.

11 RESÜMEE

11.1 PROJEKTORGANISATION

Der Projektunterricht (SYPRE – System- und Projektentwicklung) hat vorwiegend dazu beigetragen, unser Projekt erfolgreich zu planen.

Den Scrumprozess, den wir bei der Umsetzung von datadive verwendet haben, haben wir ebenfalls bereits in diesem Unterrichtsgegenstand erlernt. Durch die praktische Anwendung in einer Firma konnten wir unseren Horizont diesbezüglich, aber sehr gut erweitern. Es war möglich, den Prozess praxisnahe zu erfahren und sich gedanklich auch mit den Tätigkeiten der anderen Entwickler auseinanderzusetzen und bei Problemen gemeinsam nach Lösungen zu suchen.

11.2 DATENBANK

Vor allem durch unseren Unterricht an der HTL Perg im Fach DBI – Datenbanken und Informationssysteme – ist es uns leichtgefallen, den ersten Entwurf des Datenmodells zu erstellen.

Dieses hat sich im ersten Drittel des Projektes laufend geändert, da vor allem während des Erstellens der Skripte zum Erstellen und Einfügen in die Datenbank, immer wieder Szenarien aufgetreten sind, bei welchen etwas am Modell geändert werden musste.

Weiteres haben wir in der Schule bisher nur in der Theorie von PostgreSQL gehört, somit brauchte es eine gewisse Eingewöhnungsphase mit der neuen Benutzeransicht pgAdmin sowie auch das Kennenlernen der Syntax bei den SQL-Befehlen.

Auch beim Verbinden vom Java Backend zur Datenbank hat es mehr als einen Versuch und einige Internetrecherchen gebraucht, bis dies vollendet war und funktioniert hat.
Als diese Hürden allerdings gemeistert waren, haben wir uns schnell an das neue Datenbanksystem gewöhnt und somit im Datenbankbereich viel dazulernen können.

11.3 Backend

Java ist die Programmiersprache, die uns in den fünf Jahren HTL Perg bereits am längsten begleitet hat. Dennoch haben wir im Backend nicht nur mit der für uns unbekannten Schnittstelle JPA, sondern auch mit dem Tool Java Spring Boot gearbeitet. Auch in diesem Bereich sind wir vor allem anfangs vor großen Herausforderungen gestanden.

Allerdings konnten wir nun in der fünften Klasse – als wir JPA im Programmierunterricht erlernt haben – unseren Vorteil daraus ziehen, dass wir uns bereits intensiv mit diesem Thema auseinandergesetzt haben.

Das wahrscheinlich Schwierigste im gesamten Backend war die hexagonale Projektstruktur. Von dieser haben wir noch nie zuvor etwas gehört, deshalb ist sehr viel Zeit in die Erklärung durch unseren Betreuer, Herrn Kreuzer, hineingeflossen. Auch während der Umsetzung des Produktes stießen wir hierbei immer wieder auf Probleme. Vor allem weil es durch die vielen verschiedenen Module und deren Verlinkung im pom.xml oft zu einer **CircularDependencyException** gekommen ist.

11.4 Frontend

Damit zukünftig auch neue Anforderungen im Frontend implementiert werden können, haben wir uns in Zusammenarbeit mit dem Unternehmen Dynatrace darauf geeinigt, die Programmiersprache React in Verbindung mit TypeScript im Frontend zu verwenden.

Da wir in der HTL diese Webtechnologie nicht lernen, haben wir uns diese Sprache zuerst selbst aneignen müssen, was anfänglich auch einige Zeit in Anspruch genommen hat.

Die Auswahl der verschiedenen Frameworks, die wir für das Design oder für das Rendern der Lebensläufe benötigt haben, ist nicht ganz so einfach gewesen. Doch wir wurden von unseren Ansprechpartnern in der Firma Dynatrace gut beraten und haben somit die für uns beste Auswahl treffen können.

Am herausforderndsten im Frontend waren die API-Calls, die wir mit Axios React umgesetzt haben, sowie der Aufbau einer übersichtlichen Projektstruktur mit Seiten, Components und Contexts. Anfänglich sind wir dabei auf Probleme gestoßen, wie wir die Daten am besten übertragen.

11.5 ZUSAMMENFASSUNG

Da nun alle Probleme gelöst sind, können wir stolz berichten, sehr viel bei diesem Produkt dazugelernt zu haben und wir uns nun nicht nur technisch, sondern auch logisch betrachtet um ein Vielfaches weiterentwickelt und viele neue Kenntnisse erlernt haben.
Darüber hinaus haben wir die Möglichkeit bekommen, in den Projektablaufprozess in der praktischen Anwendung in der Arbeitswelt kennenzulernen.

12 PLANUNG UND REALISIERUNG

12.1 LASTENHEFT

Das Lastenheft, das zu Beginn des Projektes erstellt worden ist, ist im Anhang zu finden.

Dort ist angemerkt, dass es möglich sein wird, gewisse Bewerber als Favoriten speichern zu können. Im Laufe der Diplomarbeit haben wir gemeinsam mit dem Auftraggeber beschlossen, noch mehr Statistiken einzufügen und sind deshalb gemeinsam zu dem Entschluss gekommen, das Favorisieren nicht zu implementieren, dafür aber mehr Endpoints und Statistiken zu erstellen. Somit gibt es jetzt statt einem Dashboard vier verschiedene.

Außerdem ist im Lastenheft die anfänglich geplante Overview Seite festgelegt. Doch auch in diesem Punkt gab es Änderungen. Zusätzlich zum Filename werden auch die Daten, welche für die Firma am relevantesten sind, angezeigt.

Die Vorschaumöglichkeit der Lebensläufe haben wir um eine Update-Möglichkeit erweitert. Somit wird der Lebenslauf nicht in einem Pop-Up Fenster geöffnet, sondern er wird auf einer eigenen Seite gerendert, wo rechts daneben in einem Formular die personenbezogenen Daten geändert werden können.

Außerdem haben wir das Lastenheft um eine weitere Funktion erweitert, denn im fertigen Produkt werden alle Daten aus allen Lebensläufen in einer Tabelle auf der Applicants Seite angezeigt.

12.2 PROJEKTORGANISATION

Wie bereits beschrieben wurde, haben sechs Wochen der Arbeitszeit an der Diplomarbeit direkt bei unserem Auftraggeber Dynatrace am Standort im Softwarepark in Hagenberg stattgefunden.

Das Projekt haben wir im Scrum-Modell gehalten, da auch die Firma Dynatrace dieses Vorgangsmodell alltäglich verwendet. Zusammen mit unseren beiden Betreuern Herrn Jürgen Riegler und Daniel Kreuzer, ergänzt noch mit einem anderen Diplomarbeitsteam aus der HTL Perg und einem Diplomarbeitsteam aus der HTL Leonding haben wir ein „Entwicklerteam“ gebildet.

Als dieses Team haben wir gemeinsam jeden Tag ein Standup-Meeting gehalten, um zu definieren, welche Dinge man an dem jeweiligen Tag erledigen möchte. Zusätzlich hat es jeden Mittwoch ein Zwischenstands-Meeting, sowie jeden Freitag ein Wochen-Resümee-Meeting gegeben. Da wir insgesamt sechs Wochen bei der Firma vor Ort waren, hat es so funktioniert, dass eine Woche eine Sprintdauer dargestellt hat. Mit dieser Vorgehensweise konnten wir immer einen genauen Zeitplan strukturieren, um zu bestimmen, welche Dinge wann erledigt werden müssen.

12.3 Meilensteine und Projektzeitplan

12.3.1 Projektverlauf

Wie bereits erwähnt worden ist, haben sechs Wochen der Umsetzung von datadive im Rahmen eines Praktikums direkt beim Auftraggeber vor Ort stattgefunden. In der vorletzten Woche ist der Betreuer aus der HTL Perg zu uns in die Firma gekommen, um sich unsere Fortschritte anzusehen.

Der restliche Teil ist im Sommer und im Schuljahr der fünften Klasse fertiggestellt worden. Währenddessen sind wir auch mit unserem Betreuer, Herrn Daniel Kreuzer, aus der Dynatrace in Kontakt gewesen und haben ebenfalls ein Meeting gehalten, indem wir ihm unseren Zwischenstand des Produktes veranschaulicht haben.

Am 20.03.2024 hat das finale Meeting mit unserem Auftraggeber stattgefunden. Hierbei war es für Herrn Kreuzer noch einmal möglich, sich ein Bild von der fertiggestellten Diplomarbeit zu machen. Die Übergabe ist ebenfalls an diesem Tag erfolgt. Über ein privates Git-Repository, auf das auch der Betreuer aus der Dynatrace Zugriff hat, ist das Produkt – fertiggestelltes Back- und Frontend – vollendet abgegeben worden.

12.3.2 Erkenntnisse

Wir sind zu dem Entschluss gekommen, dass wir das Favorisieren der Lebensläufe durch eine Editierfunktion der Daten ersetzen, da es beim Herauslesen der Daten zu ein paar Problemen im Backend gekommen ist. Zuerst haben wir im Backend für die Sortierung der Daten eine AI verwenden wollen, doch das hat sich als zu aufwendig herausgestellt, somit werden nicht alle Daten in den Lebensläufen gefunden, wie zum Beispiel Postleitzahl, denn nach einer vierstelligen Zahl kann nicht so einfach gesucht werden, da eine Jahreszahl ebenfalls vierstellig ist. Die Editierfunktion löst dieses Problem nun.

Außerdem sind wir zusammen mit unserem Auftraggeber zu der Erkenntnis gekommen, dass in der Hauptseite nicht nur die Filenamen angezeigt werden sollen, da diese nicht aussagekräftig sind. Wir haben den Filenamen nun auf die Attribute Alter, Edukation- und Knowledge Type erweitert.

Im Backend ist ursprünglich geplant gewesen, dass die Informationen aus den Lebensläufen nicht durch ein Pattern, sondern durch ein verwendetes Named Entity Relationship Model extrahiert werden sollen. Für diesen Ansatz ist das Core NLP von der Stanford University verwendet worden. Im Laufe des Projektes hat sich aufgrund vermehrter Testdurchläufe der Bibliothek gezeigt, dass diese nur circa 60% richtige Ergebnisse liefert. Deshalb haben wir uns während der Umsetzung des Projektes gemeinsam mit der Firma Dynatrace dazu entschieden, anstatt dieser Bibliothek ein String Pattern zum Filtern zu verwenden.

Dies ist nicht nur leicht anpassbar, auch die Klasse mit dem Namen **NERMapper** – der Klasse mit der Implementierung der Verwendung des Tools – existiert nach wie vor, obwohl sie aktuell nicht verwendet wird. Falls dieses Tool also weiterhin trainiert werden soll oder die Implementierung erweitert wird, um die Ergebnisse besser verwenden zu können, ist die Verwendung des Core NLP nach wie vor im Projekt enthalten.

12.3.3 Funktionalität

Das Produkt beinhaltet folgende Funktionalitäten:

12.3.3.1 Overview

Die Übersicht soll die relevanten Daten – Filename, Alter, Education- und Knowledge Type - in eine Übersichtsseite anzeigen. So sollen die wichtigsten Daten auf einem Blick ersichtlich sein. Jeder Eintrag soll bearbeitbar sein, darum soll man per Button auf eine weitere Seite navigiert werden, wo der Lebenslauf in PDF und die Daten zum Bearbeiten in einem Formular angezeigt werden.

12.3.3.2 Applicants

Mithilfe von unserer Applicants Page werden alle Daten aus dem Lebenslauf angezeigt. Diese Daten müssen sortierbar und filterbar sein, denn nur so können die einzelnen Bewerber effektiv miteinander verglichen werden.

12.3.3.3 Dashboard

Das Dashboard soll die eingelesenen Daten aus dem Backend grafisch darstellen. Die Statistiken Gender, Applicants per Year, Education, Academic Type haben wir uns gemeinsam mit dem Auftraggeber überlegt. So soll die Anzahl der jährlichen Bewerber mit dem Vorjahr verglichen werden, ebenso soll der Prozentsatz der Bewerberinnen und der Ausbildungs- bzw. akademische Grad der Bewerber angegeben werden.

12.3.4 Entwurf der Funktionalität

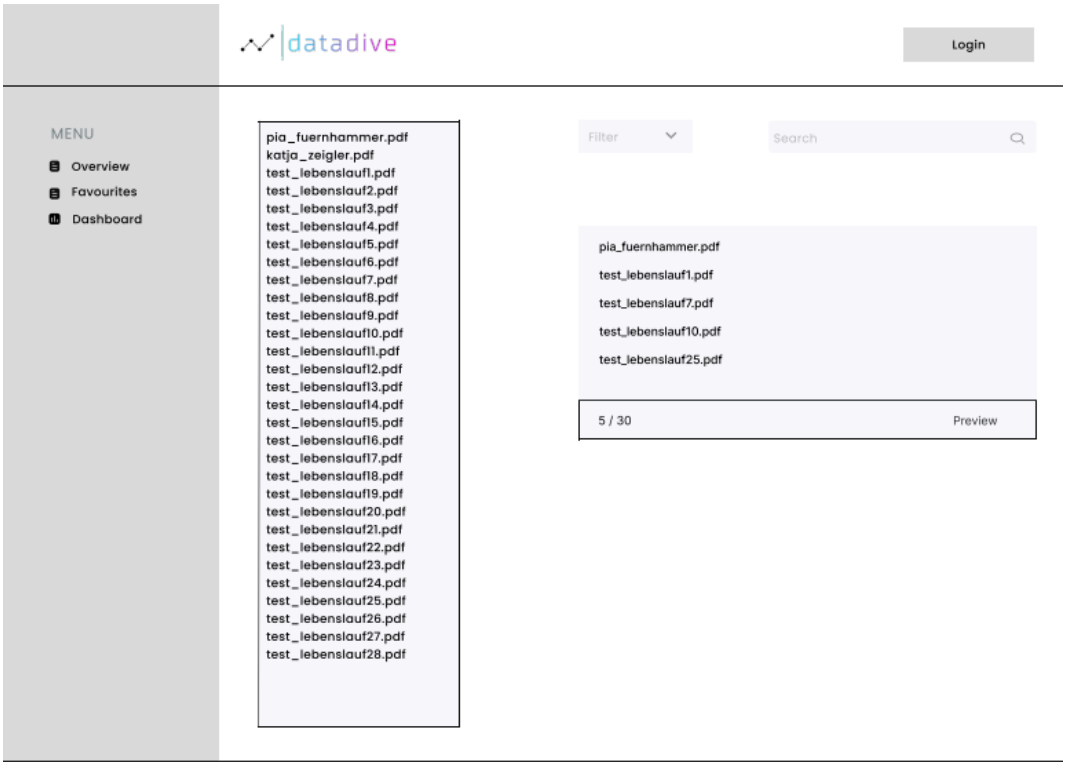


Abbildung 42 Overview Mockup

Zuerst war die Idee für unser Frontend, dass die Navigation rechts platziert ist und oben eine Login Button und unser Logo aufscheint. Wie in der Abbildung 42 zu erkennen, wollten wir in der Mitte nur die Filenamen der Lebensläufe anzeigen sowie eine Filter- und Suchfunktion.

In der Umsetzung hat sich dann doch einiges geändert, unser Auftraggeber benötigt kein Login, da die Applikation Lokal auf einem Rechner läuft. Deshalb haben wir uns dafür entschieden, dass wir die Navigation horizontal im Header positionieren und das Logo in den Footer geben. Es gibt im Header außerdem noch einen Reload-Button, wo alle Lebensläufe, die neu im lokalen Filesystem abgelegt werden, eingelesen werden. Wir vermieden nur die Filenamen anzugeben, da diese wenig aussagekräftig sind. Daher definierten wir mit unserem Betreuer Herrn Daniel Kreuzer die für die Firma wichtigsten Attribute, die in einem Lebenslauf stehen, nämlich Alter, Education- und Knowledge-Type. Diese werden zusätzlich zum Filenamen in einer Tabelle angezeigt. Die Tabelle kann sortiert und gefiltert werden.

Zusätzlich ist im Laufe der Diplomarbeit noch die Schwierigkeit aufgetreten, dass alle benötigten Daten aus dem Lebenslauf herausgefiltert werden. Darum haben wir jeden Eintrag in der Overview Tabelle mit einem Link zu einer Edit Page verlinkt. Dort wird der Lebenslauf rechts in PDF eingefügt und links daneben kommen in einem Formular alle Daten, die zu diesem Lebenslauf in der Datenbank gefunden werden. Sollte das nicht der Fall sein, kann man die leeren Felder noch ergänzen. Das Speichern wird mittels einem Save Button durchgeführt.

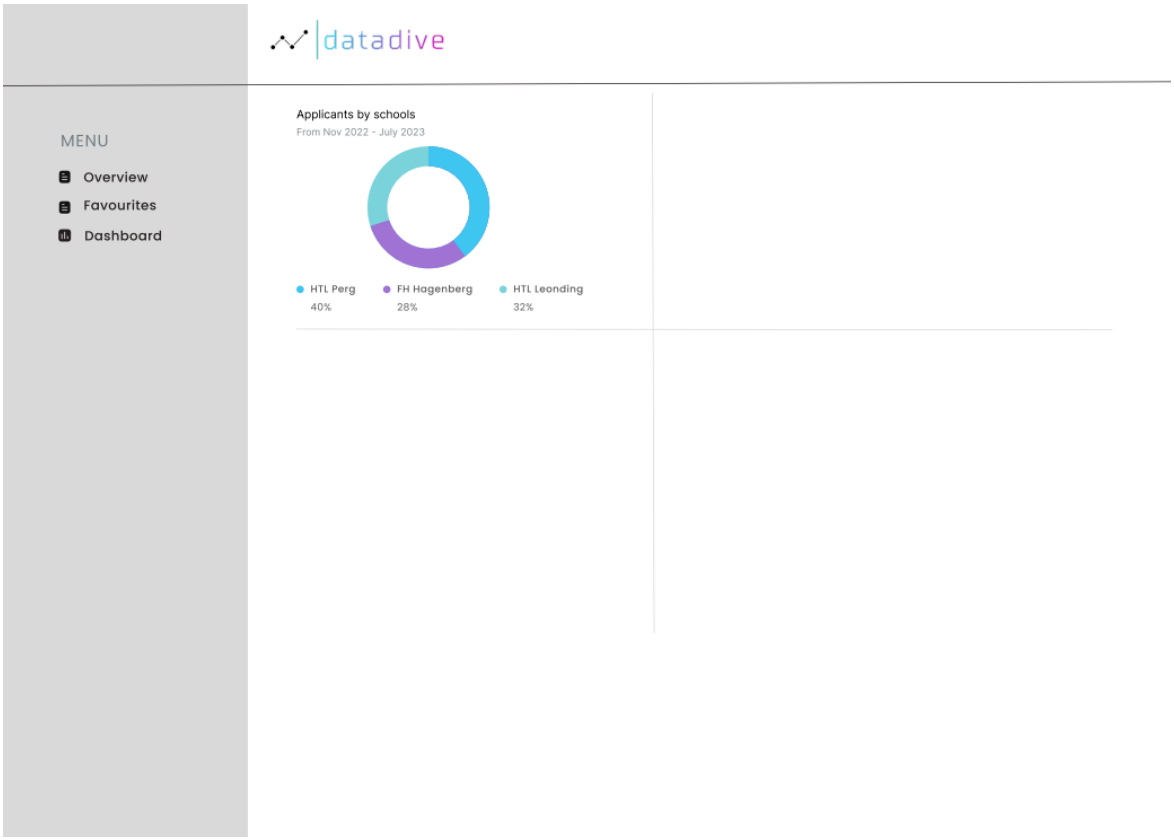


Abbildung 43 Dashboard Mockup

Da wir die Daten auch visualisieren sollen, haben wir uns überlegt, alle Auswertungen auf einem Dashboard anzuzeigen. Weil aber viele verschiedene Statistiken zu visualisieren sind, ist uns das zu unübersichtlich erschienen. Wie in Abbildung 42 ersichtlich, hätten wir nur vier Statistiken anzeigen können. Darum sind wir zu dem Entschluss gekommen, dass wir die möglichen Auswertungen gruppieren und für jede Gruppierung ein eigenes Dashboard erstellen.

Deshalb haben wir in der Navigationleiste bei der fertigen Applikation ein Dropdownmenü vorgesehen, das auf die verschiedenen Dashboards, nämlich Yearly Applicants, Carrer Snapshot, Academic Snapshot und Gender Analytics verweist.

Yearly Applicants vergleicht die Bewerber vom aktuellen Jahr mit dem Jahr davor. So kann unser Auftraggeber auf einen Blick sehen, wie sich die Bewerberzahlen verändern.

Carrer Snapshot zeigt übersichtlich, wo die Dynatrace Bewerber vor ihrer Bewerbung beschäftigt waren. Seien es verschiedene Jobs, Praktika oder Ausbildungen.

Academic Snapshot verdeutlicht den Ausbildungsstandpunkt der Bewerber, ob sie einen HTL-, HAK- Abschluss oder sogar eine Fachhochschule oder ein Studium absolviert haben.

Gender Analytics ist für die Visualisierung der verschiedenen Geschlechter der Bewerber zuständig. Damit kann verdeutlicht werden, bei welchem Geschlecht das Unternehmen am besten ankommt.

Unsere dritte Funktionalität ist erst im Laufe der Arbeit entstanden. Hierbei geht es um die Darstellung aller Daten, die wir aus den Lebensläufen eingelesen haben.

Mit dieser Funktionalität können die Bewerber gleichzeitig miteinander verglichen werden. Wir haben uns für eine Tabelle entschieden, welche die Daten übersichtlich darstellt. Diese ermöglicht dem Benutzer das Filtern jedes Attributs oder jede weitere Suche. Demnach können schnell und effizient die potenziellen Bewerber gefunden werden.

13 AUFGABENVERTEILUNG

13.1 FÜRNHAMMER PIA

3 Danksagung

7 Einleitung

- 7.1 Motivation
- 7.2 Zielsetzung
- 7.3 Projektinhalt – Überblick
  - 7.3.2 Daten auswerten
  - 7.3.3 Statistiken erstellen
- 7.4 Projektumfeld
  - 7.4.1 Projektteam
  - 7.4.3 Auftraggeber

8 Theoretische und Fachpraktische Grundlagen und Methoden

- 8.1 Verwendete Technologien
  - 8.1.5 React
  - 8.1.6 TypeScript
- 8.2 Verwendete Bibliotheken und Plug-Ins
  - 8.2.1 Bootstrap
  - 8.2.2 Material-UI
  - 8.2.3CoreUI
  - 8.2.4 React Google Charts
  - 8.2.5 ApexCharts
  - 8.2.6 Nodejs
  - 8.2.7 Node Package Manager
  - 8.2.9 Axios
  - 8.2.10 React-PDF-Viewer

9 Implementierung

- 9.3 Frontend
  - 9.3.1 Projektstruktur
  - 9.3.2 Routing

13.2 ZIEGLER KATJA

4 Kurzfassung

5 Abstract

7 Einleitung

7.2 Zielsetzung

7.3 Projektinhalt – Überblick

7.3.1 Daten der Lebensläufe in Datenbank speichern

7.3.2 Daten auswerten

7.4 Projektumfeld

7.4.2 Betreuung

8 Theoretische und Fachpraktische Grundlagen und Methoden

8.1 Verwendete Technologien

8.1.1 Java

8.1.2 Java Spring Boot

8.1.3 PostgreSQL & pgAdmin

8.1.4 JPA & Hibernate

8.1.7 Git und GitHub Desktop

8.2 Verwendete Bibliotheken und Plug-Ins

0

Swagger & OpenAPI Specifikation

9 Implementierung

9.1 Datenbank

9.1.1 Container

9.1.2 Datenmodell

9.1.2.1 Entität person

9.1.2.2 Entität contact

9.1.2.3 Entität knowledge

9.1.2.4 Entität education

9.1.2.5 Entität institution

14 LITERATURVERZEICHNIS

[1] Dynatrace, Wir alle benötigen Software, die perfekt funktioniert, Dynatrace, 2024.

[2] Oracle, 2024. [Online]. Available: https://www.java.com/en/download/help/whatis_java.html. [Zugriff am 22 03 2024].

[3] Microsoft Corporation, „Azure,“ 29 02 2024. [Online]. Available: https://azure.microsoft.com/de-de/resources/cloud-computing-dictionary/what-is-java-spring-boot.

[4] Microsoft Corporation, „Azure,“ 01 März 2024. [Online]. Available: https://azure.microsoft.com/de-de/resources/cloud-computing-dictionary/what-is-postgresql#:~:text=PostgreSQL%20ist%20eine%20relationale%20Open,PostgreSQL-Datentypen%20und%20weiteren%20Themen..

[5] baeldung, „Baeldung,“ 19 Oktober 2023. [Online]. Available: https://www.baeldung.com/learn-jpa-hibernate.

[6] kinsta, Was ist React.js? Ein Blick auf die beliebte JavaScript-Bibliothek, 2023.

[7] Platri IT, TypeScript – Was ist es und wann wird es verwendet?, 2024.

[8] BootstrapWorld.de, Was ist Bootstrap?, 2023.

[9] entwickler.de, Material-UI 1.0 ist da: Material Design für React, 2018.

[10] Material UI, Move faster with intuitive React UI tools, 2024.

[11] CoreUI, Introduction.

]

[CoreUI, How does coreUI Dashboard Template cut development time?.
1
2
]

[DhiWiseBETA, React Google Charts: A Tool for Effective Data Visualization, 2023.
1
3
]

[APEXCHARTS, APEXCHARTS.JS Modern & Interactive Open-source Charts.
1
4
]

[TechTarget, Node.js (Node), 1999-2024.
1
5
]

[Biteno, WAS IST NPM? DER PAKETMANAGER FÜR NODE.JS ERKLÄRT..
1
6
]

[I. Tasdelen, „Medium,“ 29 07 2023. [Online]. Available:
1 https://aws.amazon.com/de/what-is/java/#:~:text=Java%20is%20a%20multi-
7 platform,applications%20and%20server-side%20technologies.. [Zugriff am 14 03
] 2024].

[DigitalOcean, How To Use Axios with React, 2021.
1
8
]

[DigitalOcean, How To Use Axios with React, 2021.
1
9
]

[REACT PDF VIEWER, Basic usage - Setting up the worker, 2019 - 2024.
2
0
]

[Wikipedia, „Hexagonale Architektur,“ 22 10 2023. [Online]. Available:
2 https://de.wikipedia.org/wiki/Hexagonale_Architektur#:~:text=Die%20hexagonale%2

[1] 0Architektur%20unterteilt%20das,eine%20Alternative%20zur%20etablierten%20Schichtenarchitektur.. [Zugriff am 03 23 2024].

[2] mehmoodGhaffar, „JpaRepository and EntityManager in Spring Data JPA,“ 14 03 2023. [Online]. Available: https://medium.com/@mgm06bm/jparepository-and-entitymanager-in-spring-data-jpa-e12daad579a7. [Zugriff am 18 03 2024].

[3] The Apache Software Foundation, „pdfbox.apache,“ The Apache Software Foundation, 2017. [Online]. Available: https://pdfbox.apache.org/docs/2.0.7/javadocs/org/apache/pdfbox/text/PDFTextStripper.html. [Zugriff am 18 03 2024].

[4] W3 Schools, 2024. [Online]. Available: https://www.w3schools.com/java/java_regex.asp#:~:text=The%20matcher()%20method%20is,if%20it%20was%20not%20found.. [Zugriff am 18 03 2024].

[5] Axios, Einleitung - Was ist Axios ?.

[6] Mmdn web docs _, How to use promise, 1998-2024.

[7] Amazon web Services, „What is Cors?,“ 2023. [Online]. Available: https://aws.amazon.com/de/what-is/cross-origin-resource-sharing/#:~:text=Cross-Origin%20Resource%20Sharing%20(CORS,Ressourcen%20in%20einer%20anderen%20Domain.. [Zugriff am 28 03 2024].

[8] Amazon Web Services, „Wie behebe ich CORS-Fehler von meinem API Gatewai API aus?,“ 2024. [Online]. Available: https://repost.aws/de/knowledge-center/api-gateway-cors-errors. [Zugriff am 28 03 2024].

[9] JET BRAINS, Funktionsübersicht, 2000-2024.

[30] Amazon, „Amazon,“ 2023. [Online]. Available: https://aws.amazon.com/de/what-is/java/#:~:text=Java%20is%20a%20multi-platform,applications%20and%20server-side%20technologies.. [Zugriff am 14 03 2024].

[Oracle, 2024. [Online]. Available:

3 https://www.java.com/en/download/help/whatis_java.html. [Zugriff am 22 03 2024].
1
]

15 ABBILDUNGSVERZEICHNIS

16 LISTINGS

17 TABELLENVERZEICHNIS

18 STICHWORTVERZEICHNIS

19 ANHANG

19.1 LASTENHEFT

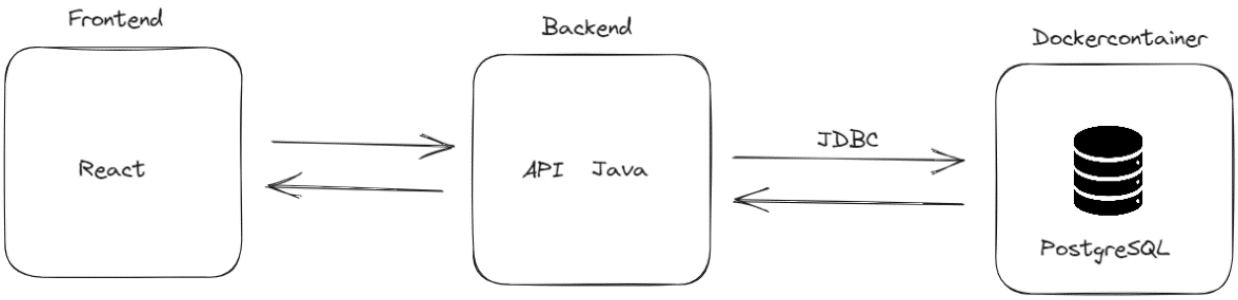
Lastenheft

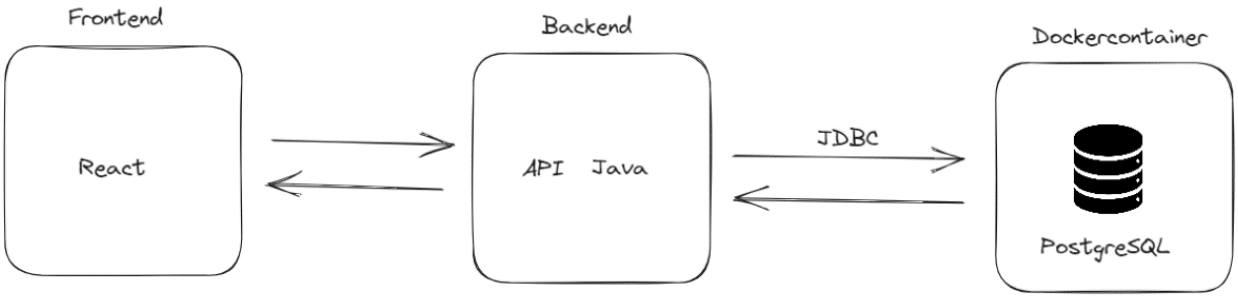
Allgemein

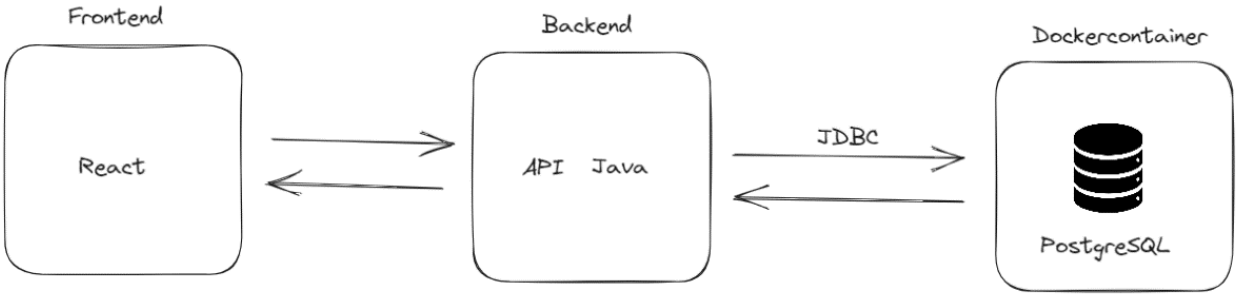
Durch unsere Diplomarbeit “datadive” sollen Statistiken über die Lebensläufe der Bewerber erstellt werden. Aufgrund dessen kann dann entschieden werden, ob der Werbeprozess optimiert werden sollte bzw. ob er beibehalten wird. Des Weiteren kann veranschaulicht werden, woher – zum Beispiel aus welchen Schulen – die Bewerber kommen, die am besten in das Team passen oder woher die meisten Bewerbungen kommen, Frauenanteil etc...

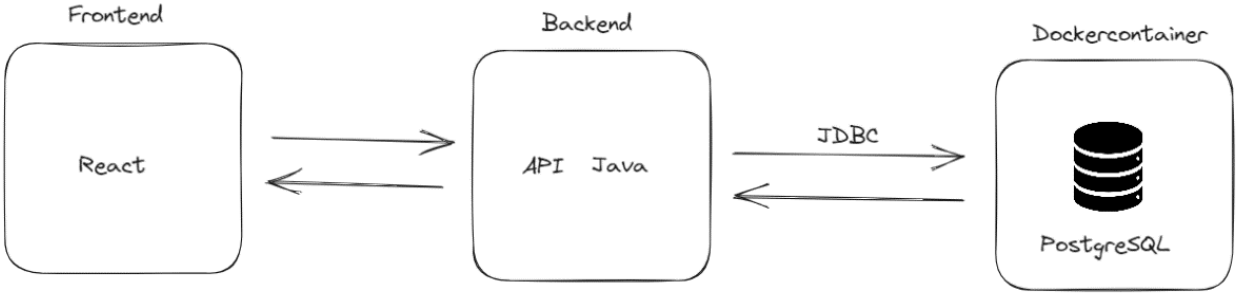
Es ist kein Server notwendig.

Architektur









Lebensläufe

Aufgrund von Datenschutz etc. kann der Auftraggeber keine Lebensläufe für Testprozesse zur Verfügung stellen. Deshalb werden „Test – Lebensläufe“ für Testzwecke erstellt, hierbei sollen so viele verschiedene Arten von Lebensläufen wie möglich abgedeckt werden.
Die Lebensläufe in späterer Folge sind dann in einem lokalen Verzeichnis abgelegt. Das hierbei verwendete Format ist .pdf.

Text Mining

Mithilfe eines Tools soll das PDF gelesen werden. Es werden dann relevante Daten in einer Datenbank persistiert.

Datenbank

Hier sind alle Informationen zu finden, die für Analysezwecke und Statistiken verwendet werden.

Datenbank → PostgreSQL

Läuft in einem Dockercontainer.

Attribute:

- Ausbildung
  - Schulen, Unis ...
  - Sortiert nach Schulstufe
- Erfahrung
  - Praktika
  - Kenntnisse
  - Letzter Arbeitgeber, wenn vorhanden
- Wohnort
  - Nur Postleitzahl, KEINE Adresse
- Alter
- Sprachen
- Name
  - Nur für 1 MONAT speichern, danach automatisiert löschen
- E-Mail
  - Nur für 1 MONAT speichern, danach automatisiert löschen
- Geschlecht
  - Männlich
  - Weiblich
  - Divers
- Bild
  - Das Bild wird NICHT gespeichert

Darüber hinaus können gewisse Lebensläufe als Favoriten gespeichert werden. Deren Name, E-Mail und Referenz auf den Eintrag in der Datenbank werden dann in einer eigenen Tabelle gespeichert. Es werden auch die Filter gespeichert.

Backend - API

Für die Verbindung zur Datenbank wird JDBC verwendet. Die Informationen aus dem Textmining kommen aus der Datenbank und werden für das Frontend so aufbereitet, dass eine übersichtliche Ansicht erstellt werden kann. Ebenfalls werden weitere Anfragen aus dem Frontend bearbeitet.

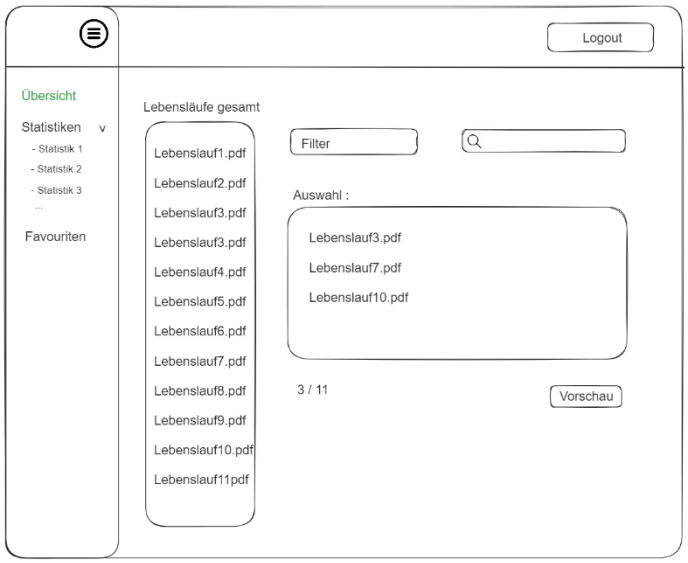
Frontend

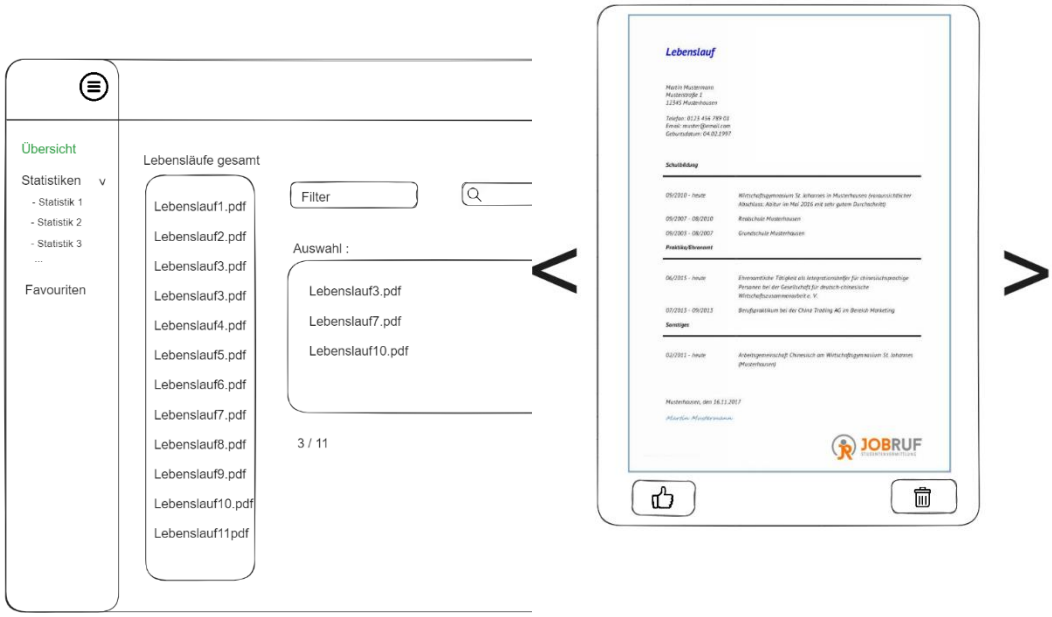
Funktionalitäten des Frontends sind:

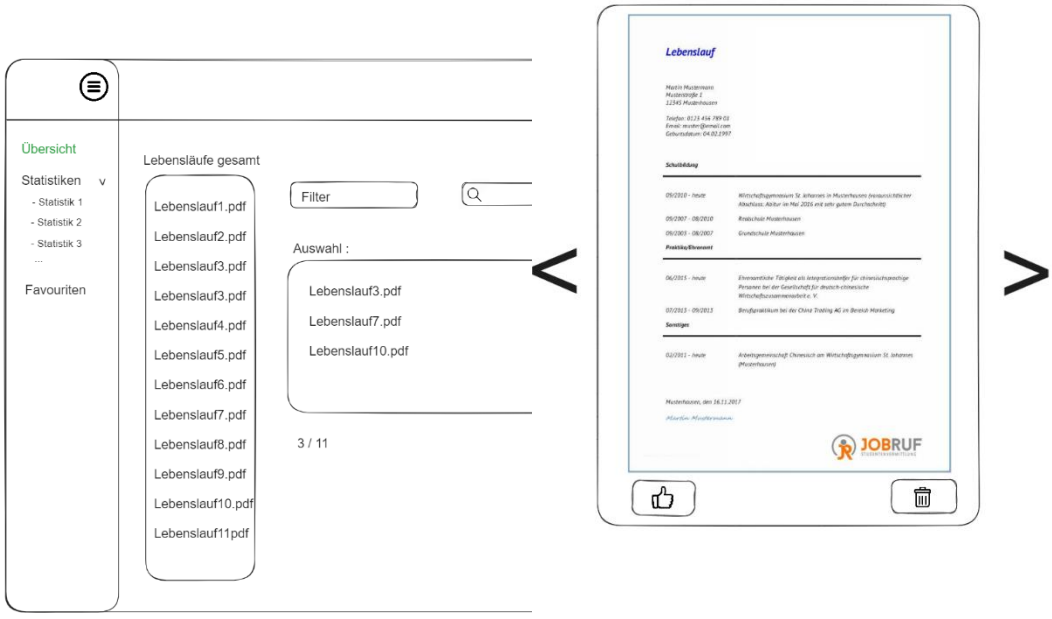
- Filtern
- Suchen
  - Nach Dateiname

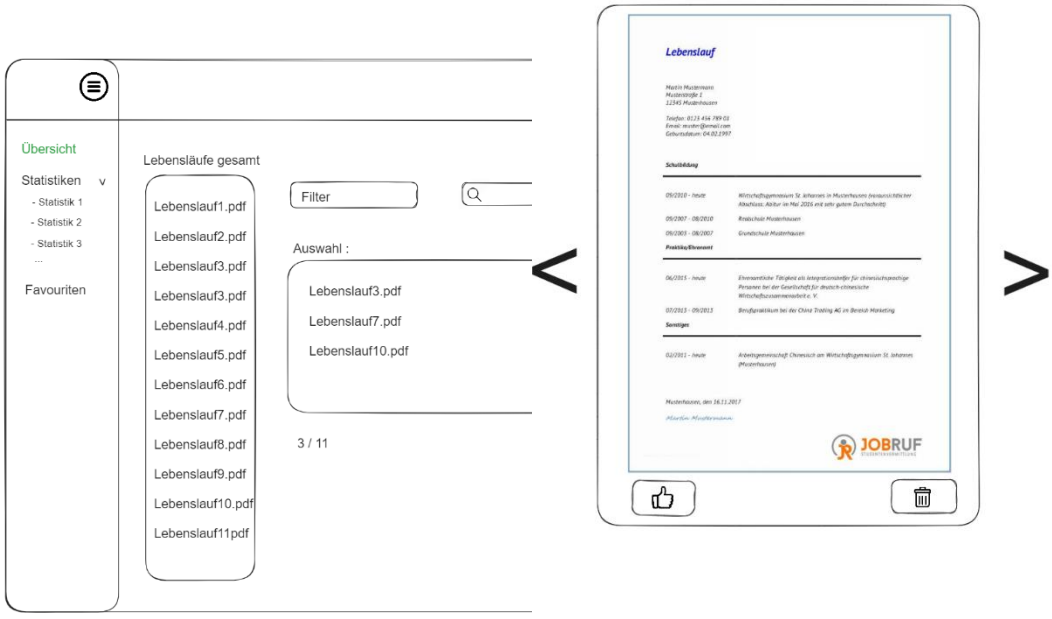
Seiten:

- **Übersicht:**
  - Filtern
  - Suchen
  - Alle Dateien via Dateinamen in Fenster anzeigen
  - Gefilterte bzw. gesuchte Dateien anzeigen
  - Anzahl der gefunden vs. der gesamten Anzahl an Lebensläufe
  - Vorschau Button




Übersicht
Statistiken
- Statistik 1
- Statistik 2
- Statistik 3
Favouriten
Lebensläufe gesamt
Lebenslauf1.pdf
Lebenslauf2.pdf
Lebenslauf3.pdf
Lebenslauf3.pdf
Lebenslauf4.pdf
Lebenslauf5.pdf
Lebenslauf6.pdf
Lebenslauf7.pdf
Lebenslauf8.pdf
Lebenslauf9.pdf
Lebenslauf10.pdf
Lebenslauf11pdf
Filter
Auswahl :
Lebenslauf3.pdf
Lebenslauf7.pdf
Lebenslauf10.pdf
3 / 11
Lebenslauf
JOBRUF


Übersicht
Statistiken
- Statistik 1
- Statistik 2
- Statistik 3
Favouriten
Lebensläufe gesamt
Lebenslauf1.pdf
Lebenslauf2.pdf
Lebenslauf3.pdf
Lebenslauf3.pdf
Lebenslauf4.pdf
Lebenslauf5.pdf
Lebenslauf6.pdf
Lebenslauf7.pdf
Lebenslauf8.pdf
Lebenslauf9.pdf
Lebenslauf10.pdf
Lebenslauf11pdf
Filter
Auswahl :
Lebenslauf3.pdf
Lebenslauf7.pdf
Lebenslauf10.pdf
3 / 11
Lebenslauf
JOBRUF


Übersicht
Statistiken
- Statistik 1
- Statistik 2
- Statistik 3
Favouriten
Lebensläufe gesamt
Lebenslauf1.pdf
Lebenslauf2.pdf
Lebenslauf3.pdf
Lebenslauf3.pdf
Lebenslauf4.pdf
Lebenslauf5.pdf
Lebenslauf6.pdf
Lebenslauf7.pdf
Lebenslauf8.pdf
Lebenslauf9.pdf
Lebenslauf10.pdf
Lebenslauf11pdf
Filter
Auswahl :
Lebenslauf3.pdf
Lebenslauf7.pdf
Lebenslauf10.pdf
3 / 11
Lebenslauf
JOBRUF

- **Vorschau:**
  - Keine eigene Seite, sondern Popup Fenster mit Buttons für rechts links
  - Button Ansicht schließen

- **Statistiken:**