CHALMERS

Webbapplikation för CV generering i molnet

En molnbaserad webbapplikation som genererar CV för konsulter

Examensarbete inom Data- och Informationsteknik

DANIEL CEBE
NURHUSEIN ABDULKADER

Institutionen för Data- och Informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2019

EXAMENSARBETE

Webbapplikation för CV generering i molnet

En molnbaserad webbapplikation som genererar CV för konsulter

DANIEL CEBE
NURHUSEIN ABDULKADER

Institutionen för Data- och Informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET

Göteborg 2019

Webbapplikation för CV generering i molnet
En molnbaserad webbapplikation som genererar CV för konsulter
DANIEL CEBE
NURHUSEIN ABDULKADER

Examinator: Peter Lundin, Data- och informationsteknik.
Handledare: Sakib Sistek, Data- och informationsteknik.

Institutionen för Data- och Informationsteknik
Chalmers Tekniska Högskola / Göteborgs Universitet
412 96 Göteborg
Telefon: 031-772 1000

Webbapplikation för CV generering i molnet
En molnbaserad webbapplikation som genererar CV för konsulter

Institutionen för Data- och Informationsteknik
Göteborg 2019

SAMMANFATTNING

Företaget FrontEdge IT har från början använt sig av en CV-hantering där alla Cv:n för de anställda är samlade i en mapp på en lokal dator. Målet med detta projekt var att skapa en molnbaserad webbapplikation som genererar ett CV för alla anställda på företaget. Denna rapport beskriver hur implementeringen av ett sådant system kan åstadkomma en effektivare CV-hantering. Det konstruerade systemet använder Springs ramverk med Java som programmeringsspråk och MySQL som databashanteringssystem samt React med JavaScript som programmeringsspråk för användargränssnittet. Resultatet blev en fungerande molnbaserad webbapplikation med tillhörande databas och ett användarvänligt användargränssnitt som underlättar CV-hanteringen och skapandet av enhetliga Cv:n på ett mycket effektivare sätt för företaget.

Nyckelord: Spring Boot, Java, MySQL, React, Webbapplikation.

ABSTRACT

The company FrontEdge IT has from the beginning used a CV management where all CVs for the employees are gathered in a folder on a local computer. The goal of this project was to create a cloud-based web application that generates a CV for all employees in the company. This report describes how implementation of such a system can achieve more efficient CV management. The constructed system uses the Springs framework with Java as the programming language and MySQL as the database management system and the React with JavaScript as the programming language for the user interface. The result was a functioning cloud-based web application with associated databases and a user-friendly user interface that facilitates CV management and creation of unified CV in a much more efficient way for the company.

Keywords: Spring Boot, Java, MySQL, React, Web Application.

FÖRORD

Den här rapporten är skriven av Daniel Cebe och Nurhusein Abdulkader och gjordes som ett examensarbete motsvarande femton högskolepoäng på högskoleingenjörsprogrammet datateknik på Chalmers tekniska högskola under de sista tretton veckorna under vårterminen 2019.

Vi vill tacka Fredrik Lundé på FrontEdge IT som tog emot oss för att utföra ett examensarbete och för den tid som han har ägnat åt oss under examensarbetet. Vi vill även tacka honom för all stöttning han har gett oss under vårt examensarbete.

Vi ger Niclas Hammar ett tack som konsult på FrontEdge IT som har gett oss hintar över mejl vid frågor gällande programmering och molntjänster.

Slutligen vill vi ge ett stort tack till vår handledare Sakib Sistek för hans goda personlighet. Tack för att du har motiverat och stöttat oss.

TERMINOLOGI

AWS	En förkortning för Amazon Web Services som är en molntjänstplattform.
API	Applikationsprogrammeringsgränssnitt, ett sätt att hämta och uppdatera data mellan olika system.
Back-end	Allt som sker bakom kulisserna för en applikation, det vill säga ett bakomliggande system med servrar och databaser.
CSS	Förkortning för Cascade Styling Sheet, används inom webbutveckling som utformar utseendet för en webbsida.
Front-end	Tillför vad användaren ser i sin webbläsare, det vill säga gränssnittet för webbplatsen som användaren interagerar med och koden som körs på klientsidan.
GUI	Graphical User Interface. Ett grafiskt användargränssnitt det vill säga kontaktyta mellan användare och dator.
HTTP	Hypertext Transfer Protocol.
IDE	Integrerad utvecklingsmiljö för mjukvaruutveckling.
JDBC	Java DataBase Connectivity, en standard Java API för hantering av information mot SQL-databaser.
JPA	Java Persistence API, är ett Java applikationsprogrammeringsgränssnitt.
JSON	JavaScript Object Notation (JSON). Ett kompakt, textbaserat format som används för datautbyte.
JUnit	Ett ramverk för enhetstestning i Java.
ODBC	Open DataBase Connectivity. Ett API för åtkomst till databashanteringssystem det vill säga kommunikation mellan applikation och databas.
ORM	Object-relational mapping, en programmeringsteknik för att koppla objektkod till en relationsdatabas.
PDF	Portable Document Format, ett digitalt dokumentformat.
URI	Uniform Resource Identifier (URI)**.** Dataprogrammeringsterm som består av en sträng av tecken som används för att namnge eller identifiera en resurs.
URL	Uniform Resource Locator(URL), internetadress, det vill säga en teckensträng som identifierar en viss resurs på nätet.

Innehåll

1. Inledning

I det här kapitlet presenteras uppdragsgivaren och bakgrunden till arbetet. Därefter följer beskrivning av syfte och mål för projektet och slutligen beskrivning av eventuella avgränsningar.

1.1 Bakgrund

Projektförslaget utgavs av Fredrik Lunde; VD på konsultföretaget FrontEdge IT och går ut på att skapa en molnbaserad webbapplikation. FrontEdge IT arbetar med mjukvaruutveckling som tar emot uppdrag vilka utvecklas “in-house”, det vill säga inom företaget med hjälp av företagets tillgångar och anställda. Företaget kan även hyra ut sina anställda för uppdrag ute hos kund. För att en kund ska få rätt kännedom om att konsultföretaget har rätt personal för det typ av uppdrag som ska genomföras kan ett CV stärka beviset.

Ett problem som förekommer regelbundet är att chefen för FrontEdge IT måste lägga ner onödig tid för att hantera CV uppdateringen manuellt för anställda. I nuvarande fall begär chefen att de anställda alltid ska skicka över den senaste versionen av sitt CV. Detta är ett problem i och med att det kan vara svårt att hålla koll på alla Cv:n och vilka som har uppdaterats. I sin tur är det tidskrävande samt att det finns en risk för att alla anställdas Cv:n inte är enhetliga. Ett system ska utvecklas för att kunna lösa detta problem genom att skapa en webbapplikation där varje anställd själv ska kunna uppdatera sitt CV i molnet. Tanken är att man som anställd håller sitt CV uppdaterat med sina senaste uppdrag och vilka kompetenser man behärskar och att detta ger tillgång till chefen att kunna se de anställdas CV-profil mycket effektivare samt alla uppdateringar som sker automatiskt.

1.2 Syfte

Syftet med projektet är att utveckla ett molnbaserat IT-system för hantering av anställdas Cv:n. Systemet skall utöver att underlätta och effektivisera administrationen av anställdas Cv:n skapa ett standardiserat format på Cv:n och alltid kunna visa den senaste uppdaterade profilen.

1.3 Mål

Målet med projektet är att utveckla en molnbaserad webbapplikation som kan generera en mall för ett CV, möjliggöra editering av Cv:n, lagra tillhörande data samt skapa en PDF version av Cv:n. Systemet skall utvecklas med RESTful API:er, samt sträva efter kontinuerlig integration.

1.4 Avgränsningar

Projektet kommer anpassas till företaget FrontEdge IT vilket innebär att webbapplikationens innehåll är strikt anpassad för att generera CV för personer på företaget.

För att inte överskrida tiden för planeringen kommer inget större fokus ligga på att designa ett fint grafiskt användargränssnitt då detta kan resultera till att viktiga funktioner inte hinner implementeras.

2. Teknisk bakgrund

2.1 Back-end

Det här avsnittet kommer att behandla de tekniker som har använts i det bakomliggande systemet av projektet. Här beskrivs ramverk, RESTful API, bibliotek och utvecklingspraxis.

2.1.1 Spring och Spring Boot

Spring är ett ramverk med öppen källkod. Ramverket ger ett omfattande infrastrukturstöd för att utveckla Java-applikationer. Med detta sagt, hanterar ramverket infrastrukturen för att låta utvecklarna fokusera på applikationsutveckling. Användning av ramverket Spring gör det möjligt att skapa en högpresterande, lätt testbar, återanvändbar kod, när man utformar och utvecklar Java applikationer [1].

Spring Boot är ett projekt byggt på toppen av Spring. Liksom en extension av ramverket Spring. Den ger ett enklare och snabbare sätt för att komma igång med minimal konfiguration och skapa fristående, produktionskvalificerade webbaserade applikationer [2].

2.1.2 RESTful API

RESTful eller REST står för Representational State Transfer och är ett IT-arkitekturbegrepp som redogör för hur tjänster via webbteknologi kan tillhandahållas för maskin-till-maskin-kommunikation [3]. Ett RESTful API bygger på REST teknologi. RESTful API är till för att utföra förfrågningar (GET) och få svar (POST) via HTTP-protokoll, det vill säga ett kommunikationsprotokoll för överföring av webbsidor på informationsnätverket på internet [4].

2.1.3 Spring Data och Spring Data JPA

Spring Data är en del av Spring ramverket och är till för att tillhandahålla välkänd och konsekvent Spring-baserad programmeringsmodell för dataåtkomst. Det gör det enkelt att använda teknologier för dataåtkomst, relationsdatabaser och icke relationsdatabaser [5].

Spring Data JPA är en del av Spring Data och förenklar tillgång till datalager och minskar antalet rader kod, samtidigt som att bibehålla en rik och komplett uppsättning av funktioner. För att göra det möjligt tillåter Spring Data JPA enkel implementation av JPA baserade repositories. Dessa repositories är Java-gränssnitt som gör det möjligt för utvecklaren att bland annat skriva anpassade sökmetoder, och Spring kommer hantera implementeringen automatiskt [6].

2.1.4 IText

IText är ett bibliotek i bland annat Java för att skapa och manipulera PDF-dokument. Med IText är det möjligt att designa och bestämma hur informationen i dokumentet ska vara. Exempelvis kan man med hjälp av IText inkludera bilder, tabeller och teckensnitt. Det är även möjligt att spara PDF-dokument som bildfiler [7].

2.1.5 Kontinuerlig integration

Kontinuerlig integration är en utvecklingspraxis som går ut på att utvecklare integrerar källkod regelbundet till ett gemensamt repository. Det vill säga en plats där alla utvecklare kan checka in kod med versionshantering och kan få tillgång till källkoden. Varje ändring av källkod checkas in och verifieras sedan av ett automatiserad byggsystem. Detta gör det möjligt för utvecklarna att upptäcka problem tidigt. Genom att integrera regelbundet kan man upptäcka fel snabbare och hitta felen lättare [8].

2.2 Front-end

Det här avsnittet kommer att beskriva de bibliotek som har använts för att utforma ett användargränssnitt i front-end.

2.2.1 React

React är ett open-source JavaScript bibliotek som används för att bygga användargränssnitt och är specifikt för *"single page"* applikationer. Single page-applikationer eller så kallade enkelsida applikationer laddar upp en enda HTML-sida och uppdaterar den sidan dynamiskt när användaren interagerar med applikationen. React underlättar uppbyggnad av användargränssnitt på ett effektivt sätt. Detta genom att dela upp varje sida i olika delar så kallade React-komponenter. En React komponent är en JavaScript funktion som representerar en del av en sida. För sammanställning av en hel sida anropas funktionerna i en viss ordning, och resultatet visas för användaren [9].

2.2.2 Redux

Redux är ett JavaScript bibliotek med öppen källkod och är ett hjälpmedel för att hantera tillståndet för applikationen. I React där data delas mellan komponenter kan det vara rörigt att veta var tillståndet ska vara. Med Redux som hjälpmedel hålls applikationens tillstånd i en behållare som kallas för Store. Varje React Komponent kan få tillgång till det lagrade tillståndet de strävar efter från Store. Redux består huvudsakligen av tre delar. Utöver Store är de två andra delarna Actions och Reducers. Actions är händelser, vilket är det enda sätt för att skicka data från applikationen till Store i Redux. Data kan komma från RESTful API:er, användarinteraktioner eller formulär som fylls med data [10]. Reducers är funktioner som anger hur applikationens tillstånd ändras som svar på Actions som skickas till Store [11].

2.3 AWS

En molntjänst kan uttryckas som leveransen av datatjänster vilka kan vara servrar, lagring, databaser över internet(“molnet”) för att bland annat erbjuda snabbare innovation och flexibla resurser [12]. AWS är ett exempel på en molntjänstplattform som är bland de mest populära i världen. Den erbjuder över 165 fullt utrustade tjänster från datacenter globalt [13]. Utöver det erbjuder AWS fördelar som hög prestanda, skalbarhet, kosteffektiv, säkerhet, flexibilitet och enkel användning [14].

3. Metod

I det här avsnittet kommer de arbetssätt som använts under projektet att förklaras. Kapitlet beskriver först det förberedande arbetet som planeras inför projektstart. Därefter följer en förklaring av den agila arbetsmetod för mjukvaruutveckling som kommer att användas.

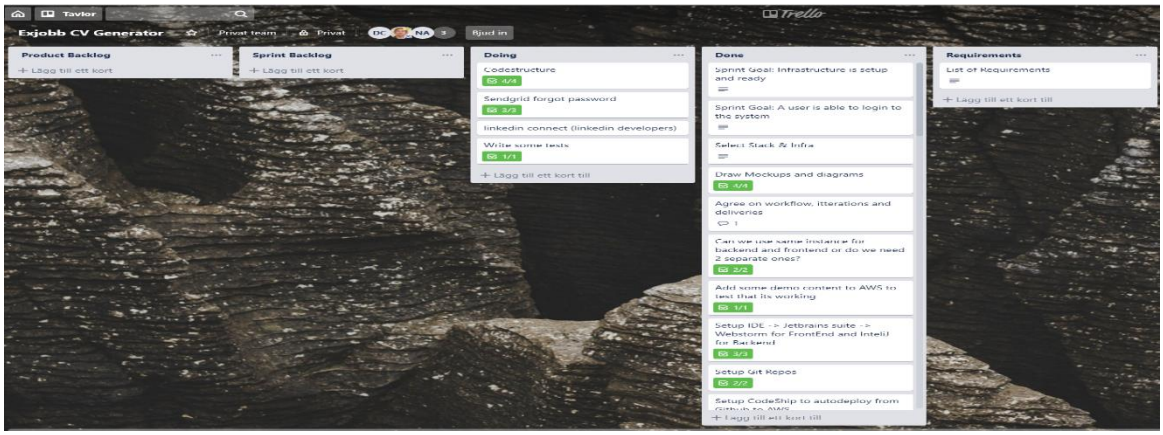
3.1 Förberedande arbete

Ett förberedande arbete utförs innan någon utveckling av webbapplikationen sker. Det förberedande arbetet omfattar fyra väsentliga saker där det första är att skapa en lista av krav för webbapplikationen som ska fastställas tillsammans med företaget. Dessutom kommer alla arbetsuppgifter och mål veckovis att dokumenteras. Utöver det ska även en tidsplan läggas upp som sträcker sig i cirka tio veckor. Tidsplanen innehåller start- och slutdatum för utvecklingen av projektet men även områden som projektet kommer bestå av vilka kan ses i Gantt-schemat i appendix A. Det tredje blir att rita upp preliminärt utseende för applikationen innan påbörjad utveckling. Slutligen kommer projektmiljön att sättas upp där val av integrerade utvecklingsmiljöer, relevanta ramverk och infrastrukturer ska bestämmas för att underlätta utvecklingen av webbapplikationen.

3.2 Agilt arbetssätt med Scrum

Utvecklingen av applikationen kommer att ske med ett agilt arbetssätt som är passande för system- och mjukvaruutveckling. Det agila arbetssättet används i projektet för att uppnå snabb leverans, iterativ och inkrementell process. Till detta har en agil projektmetodik Scrum använts som hjälp. Dock kommer inte alla teorier inom Scrum att appliceras på grund av projektgruppens storlek. Arbetet kommer att delas upp i korta iterationer så kallade sprintar som sträcker sig under en vecka. Under sprintarna fördelas arbetet med fyra dagar utveckling, en dag för verifiering att de implementerade funktionerna funkar, samt en dag för Sprint Review(möte) och dokumentation. Sprint Review hålls i slutet av varje iteration för att demonstrera funktionaliteter som utvecklats under veckan. Diskussioner över vad som behöver förbättras och vilka uppgifter som bör lösas till nästa sprint är annat som lyfts fram under varje möte.

Under arbetets gång kommer en digital Scrum Board att användas på hemsidan Trello där alla parter som är inblandade får åtkomst till all källkod. Trello är tänkt att användas som hjälp för att fullfölja arbetsprocessen i arbetet. Scrum Board kommer bland annat innehålla de listade kraven som fastställts under det förberedande arbetet men även mål som skrivs respektive vecka och prioriteringsordning för de olika funktioner som ska implementeras. Figur 3.1 visar en klarare illustration av Scrum Board för projektet.



Figur 3.1: *Illustration av Scrum Board som används i arbetet.*

4. Kravspecifikation

En kravspecifikation togs fram tillsammans med chefen på företaget. Detta med hjälp av en diskussion om förslag som berör funktionaliteter vilka kan passa in i en kravspecifikation. Förslagen som lyftes upp visade sig vara lämpliga till att forma en lista med krav med tanke på tidsestimeringen och nivån på webbapplikationen. Ett av de förslag som lyftes upp var att både en administratör och en användare ska kunna skapa ett CV och utifrån det generera en PDF som ska kunna laddas ner från applikationen. Det lämpade sig därför att ha ett krav som låter applikationens användare att bestå av två olika roller: vanlig användare eller administratör. Till följd av detta var det även lämpligt att ha ett inloggningssystem som krav.

Vanliga användare i detta fall är anställda på företaget som kan få en översikt över sin CV-profil vid inloggning i systemet. De ska alltså med applikationens hjälp uppnå detta, men även kunna ladda ner Cv:t, lägga till, uppdatera eller ta bort information om sitt CV samt ändra sitt lösenord för inloggning. Utöver detta har vanliga användare inga andra rättigheter i systemet.

Användare med rollen administratör ska ha åtkomst till alla funktioner systemet besitter. Det innefattar full tillgång till sin egen CV-profil och en egen administratör sida som ger åtkomst till alla CV-profiler som tillhör vanliga användare. Administratören kan utöver det även gå in och ändra, ta bort eller lägga till information i en CV-profil som tillhör en vanlig användare. Till följd av detta kan administratören ladda ner både sitt eget CV och en vanlig användares CV.

<u>Funktionella krav:</u>

- En administratör eller vanlig användare kan logga in i systemet.
- Skapa ett CV genom att lägga till information i systemet som sparas till databasen.
- Lägga till, ändra, ta bort information från sitt CV.
- Ladda ner CV i PDF- och Word format från applikationen.
- Om en administratör eller vanlig användare glömt sitt lösenord ska detta kunna återställas med ett nytt lösenord via email.
- En administratör kan bjuda in användare via inbjudningslänk som ger möjlighet för att registrera ett konto.

<u>Icke-funktionella krav på systemet:</u>

- Systemet ska köras på Amazon Web Services.
- Systemet ska kräva inloggning som styr vilka funktioner en administratör och användare kan använda.
- Flexibelt system som gör det möjligt vid eventuell vidareutveckling av systemet.

5. Analys

Det här kapitlet beskriver det befintliga systemet som företaget använder. Utöver det innehåller kapitlet val av plattform, molntjänst och verktyg, beskrivning av applikationens domänmodell och applikationens prototyp.

5.1 Analys av befintligt system

Hittills har företaget använt sig av en CV-hantering där alla Cv:n för de anställda är samlade i en mapp på en lokal dator. Följande brister finns på nuvarande CV-hantering;

- Nuvarande Cv:n för anställda har olika utseende vilket gör att de inte uppfyller enhetlighet. Företaget har försökt att lösa detta genom att en nyanställd får fylla i en mall i form av Word dokument för att uppnå enhetliga Cv: n. Detta har inte varit optimalt på grund av att anställda på företaget har tyckt att det är rörigt att anpassa sitt gamla CV till företagets mall samt att mallen ger en ganska otydlig förklaring på vad för innehåll som ska finnas under respektive sektion.

- Cv:n behöver ständigt uppdateras med nya uppdateringar då anställda som har jobbat med uppdrag hos kund får nya meriter. Meriterna behöver läggas till på de anställdas personliga Cv:n. Företaget har försökt att lösa detta problem genom att de anställda som har genomfört uppdrag hos kund ska uppdatera sitt CV på egen hand och senare skicka den uppdaterade versionen av Cv:t till chefen via mejl. Detta har inte fungerat fullt ut då det har hänt att mejl inte hamnat på inkorgen men även att formateringen har ändrats vid nedladdning av filen.

FrontEdge IT vill alltid vara i framkant när det gäller uthyrning av anställda, där meriter för anställda värderas högt. Det befintliga systemet med hantering av Cv:n gör processen onödigt tidskrävande med tanke på bristerna ovan. För ett nystartat företag som FrontEdge IT är det ytterst viktigt att ha ett arbetssätt som är modernt och eftertraktat för att locka till sig kunder. Därför vill företaget satsa på ett modernare sätt att jobba på. De har försökt att använda sig av Linkedin och liknande applikationer men dessa uppfyller inte de mest viktiga funktionerna som företaget är ute efter.

5.2 Analys av val av plattform

Två separata applikationer ska byggas som sedan ska integreras. För att göra det möjligt kommer projektet att delas upp i back-end och front-end. Till dessa finns ett stort val av ramverk och bibliotek när det gäller utveckling av webbapplikationer. Spring är det ramverk som ska användas för back-end medan React tillsammans med Redux är det bibliotek som ska användas i front-end. Syftet med detta är att sträva efter god modularitet, snabbare implementering, resursoptimering, bättre skalbarhet i systemet vilket gör att man kan ändra och utveckla en applikation utan påverkan av en annan applikation [15].

5.2.1 Val av back-end plattform

För att kunna gå tillväga att utföra en back-end applikation finns det ett antal valmöjligheter. Utgångspunkten var att hitta ett ramverk som är passande för webbutveckling med egenskaper som uppfyller god prestanda, enkel integration med databaser och bra dokumentation som underlättar för nya utvecklaren att komma igång. Två alternativ som är passande för just detta projektet är ExpressJs och Spring. Valet föll på Spring i utvecklingsprocessen för back-end. Detta på grund av att Spring är snabbare, komplett och mer konfigurerbart än ExpressJs [16]. Soni (2015) nämner att Spring erbjuder omfattande infrastrukturstöd för att utveckla webbapplikationer och att det är ett kraftfullt ramverk som bland annat stöder JDBC ramverk som förbättrar produktiviteten och minskar fel. Annat som nämns är att Spring ger abstraktion på ORM för att utveckla ORM persistenslogiken som gör att integreringen med databasen blir lättare. Spring är även modulärt på det sättet att man använder bara de delar som behövs. Fördelarna som Spring erbjuder visar att de egenskaper som eftersöktes uppfylls.

För att underlätta utvecklingen av webbapplikation har Spring Boot använts som en extension av ramverket Spring. I sin tur har Spring Boot använts för minimal konfiguration och en snabb uppstart för att komma igång med utvecklingsprocessen.

5.2.2 Val av front-end plattform

När det gäller front-end finns en del befintliga alternativ att välja mellan. React och Vue är två olika alternativ. Både React och Vue är bra verktyg för att bygga interaktiva användargränssnitt. Dokumentationen för Vue är utmärkt och innehållet förklaras detaljerat. React är dock utmärkt för att bygga komplexa applikationer i företagskvalitet och är en veteran inom JavaScript-fältet med företagsstöd, en stor gemenskap och ännu större ekosystem [17]. Valet blev att använda React för att bygga webbapplikationens användargränssnitt. Den främsta anledningen till att det blev just React är för att chefen på företaget har erfarenhet inom detta område och var övertygad om att det skulle gå fort att bekanta sig med React för att utveckla ett användargränssnitt. React erbjuder ett flertal olika fördelar som gynnar utvecklandet. Exempelvis utvecklingseffektivitet, det vill säga högre kvalité och mindre tid för att utveckla. React ger även möjlighet att bryta ner projektet i separata komponenter samt skriva modulär och ren kod vilket möjliggör kodåteranvändning. Andra viktiga fördelar är att React erbjuder hög prestanda som gör det möjligt spara tid vid utveckling och dessutom skapa en snabbare applikation [18]. Med tanke på fördelarna och chefens åsikt ledde det till att React var ett intressant val för projektet.

Redux har använts tillsammans med React. Anledningen är för att Redux underlättar hantering av tillståndet för applikationen. Redux ger möjligheten att lättare förstå hur dataflödet fungerar i applikationen samt att Redux erbjuder ett verktyg som gör det enkelt att spåra när, var, varför och hur tillståndet för applikationen har ändrats [19].

5.3 Analys av val av molntjänst

För att komma åt data från applikationen när och var som helst över internet krävs att applikationen ligger på en server. Med detta sagt finns en hel del olika molnplattformar att

välja emellan. AWS är den plattform som valdes. Anledningen till att just AWS valdes är för att den rekommenderades av chefen på företaget samt att den erbjuder gratis användning av vissa tjänster så länge användningen av tjänsterna inte överstiger 750 timmar per månad. Utöver det, erbjuder AWS flexibilitet. Med detta sagt är det möjligt att välja operativsystem, fritt val av programmeringsspråk, webbapplikationsplattform, databas och andra tjänster som behövs. AWS är också enkelt att använda då plattformen erbjuder en hanteringskonsol som ger en god vägledning tack vare ett grafiskt gränssnitt för åtkomst till ett brett utbud av AWS tjänster samt att det finns väl dokumenterade webbtjänster. AWS Elastic Beanstalk är en lättanvänd gratistjänst som har använts för att distribuera och skala webbapplikationer med Java och JavaScript. Både back-end och front-end har använt två separata AWS Elastic Beanstalk tjänster. Man kan helt enkelt ladda upp sin kod och Elastic Beanstalk hanterar distributionen automatisk, från kapacitettillhandahållande, automatisk skalning till övervakning av applikationshälsa [20]. Elastic Beanstalk erbjuder integration med Amazon Relational Database Service som använts [21]. Amazon Relational Database Service är också en gratistjänst som tillämpats för att lägga till en databasinstans i Elastic Beanstalk miljön. Tack vare nyttjandet av denna tjänst ger det möjlighet att konfigurera, driva och skala en relationsdatabas i molnet [22].

5.4 Analys av val av verktyg

5.4.1 InVisionapp

InVisionapp är ett digitalt prototypverktyg som låter dig att designa dina idéer [23]. Anledningen till att verktyget valdes är för att InVisionapp ger möjlighet att skissa upp hur det grafiska användargränssnittet för ett system ska se ut innan påbörjad utveckling. Detta på ett effektivt sätt då flera utvecklare kan designa samtidigt med internetuppkoppling.

5.4.2 Db diagram

Db diagram är ett verktyg för att designa ett relationsdatabasdiagram [24]. Detta verktyg valdes för att det är gratis och enkelt att använda för att rita Entity-Relationship diagram genom att skriva några få rader kod.

5.4.3 MySQL

MySQL är ett databashanteringssystem som är fritt att använda. Databashanteringssystemet använder sig av språket SQL för att lägga till, komma åt och hantera innehåll i en databas [25]. Anledningen till att MySQL valdes är för att databashanteringssystemet har använts i tidigare projekt samt att det valda ramverket för projektet som är Spring erbjuder en enkel integration med MySQL för att komma åt data.

5.4.4 Git

Git är en gratis programvara som används för distribuerad versionshantering. Git är utformat för att hantera allt från små till stora projekt med prestanda som utgångspunkt [26]. Detta verktyg valdes för att förenkla utvecklingsmetodiken för projektet då Git

erbjuder att flera utvecklare kan arbeta med samma filer och på ett enkelt sätt slå samman förändringar med projektets huvudgren.

5.4.5 CodeShip

CodeShip är en kontinuerlig integrationsplattform för att bygga och distribuera applikationen från GitHub till en vald molnplattform [27]. Anledningen till att CodeShip valdes är för att slippa distribuera applikationen manuellt vid varje ändring av källkoden. CodeShip upptäcker ändringar av källkoden och distribuerar senaste versionen till molntjänsten.

5.4.6 JetBrains

JetBrains är ett företag specialiserat för att skapa intelligenta utvecklingsverktyg för mjukvaruutveckling. Exempel på utvecklingsverktyg som JetBrains erbjuder är IntelliJ IDEA och WebStorm. IntelliJ IDEA är ett Java integrerat utvecklingsmiljö (IDE) och är det utvecklingsverktyg som har valts för back-end. Anledningen till att IntelliJ valdes är för att den är gratis för studenter och erbjuder flera fördelar såsom snabb och intelligent upplevelse genom att ge relevanta förslag i alla sammanhang och tillförlitliga refaktoreringsverktyg [28].

WebStorm IDE är ett JavaScript integrerat utvecklingsmiljö och är det utvecklingsverktyg som har valts för front-end. Anledningen till att WebStorm valdes är för att den är gratis för studenter och ger ett perfekt stöd för JavaScript, HTML och CSS samt för ramverk eller bibliotek som React [29].

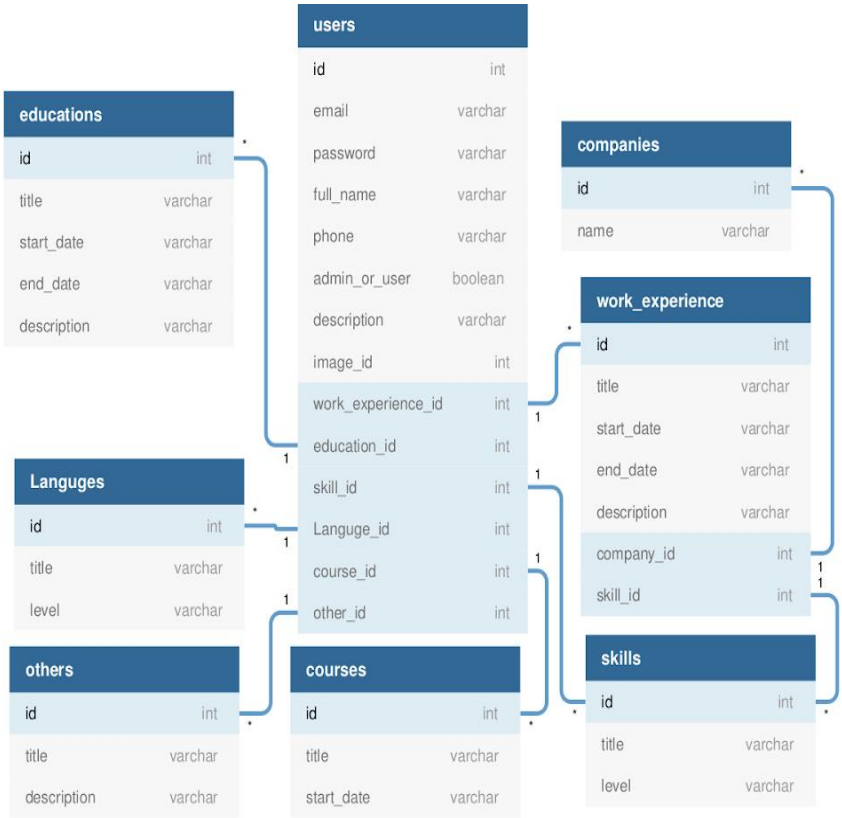
5.4.7 Postman

Postman är ett verktyg för att testa RESTful API:er. Anledningen till att Postman har valts är för att verktyget erbjuder ett användarvänligt GUI för att konstruera förfrågningar och svar för att säkerställa kvalité och funktion för RESTful API:er. Genom att skicka förfrågningar i form av GET, PUT, POST och DELETE kan man få ett svar tillbaka som verifierar om funktionalitet fungerar som det ska [30].

5.5 Domänmodell

I tidigt skede påbörjades utformning av domänmodell för applikationen med kravspecifikation som utgångspunkt. Huvudmålet med modellens uppbyggnad är att uppnå en bra struktur genom att dela upp modellen i flera entiteter. Tillsammans med chefen på företaget har en domänmodell med tillhörande entiteter kunnat utformas vilka illustreras i figur 5.1. En användare som motsvarar user entity utgör modellens centrala del i domänen. Entiteter som har tagits ut är användare, arbetslivserfarenhet, utbildning, färdighet, språk och företag. Relationer mellan användaren och de resterande entiteter är *many-to-one*, detta för att se till att en användare har en kompakt och komplett domän.

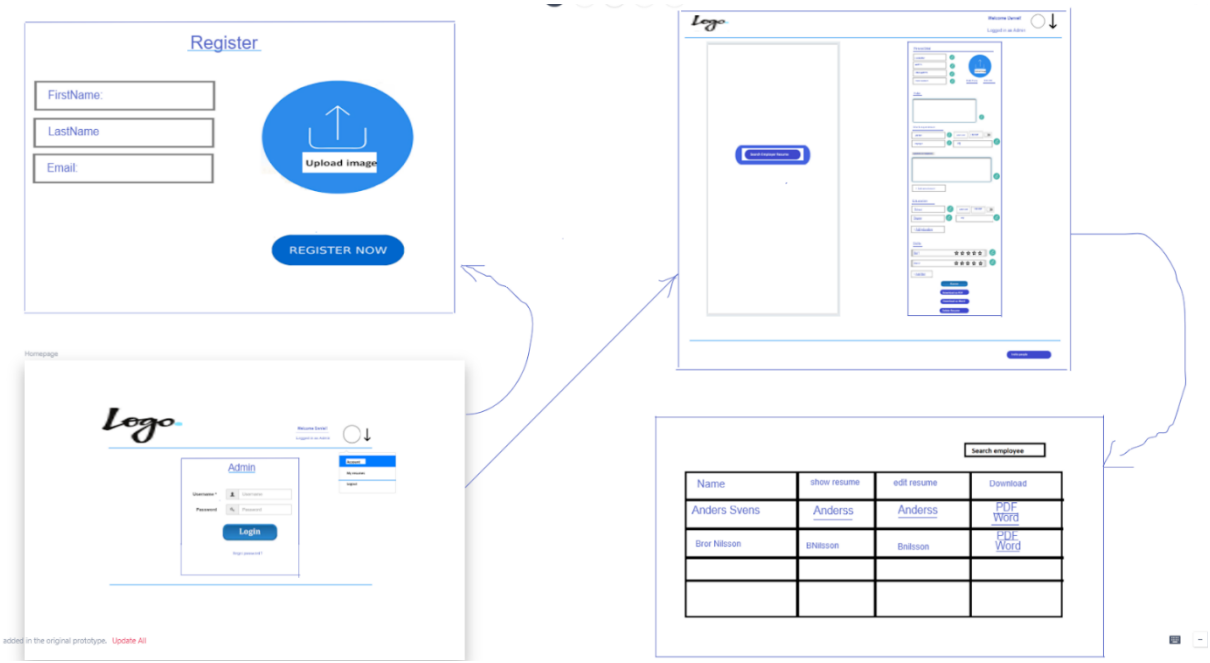
Vid konstruktion av modellen dök en fråga upp huruvida bilder ska ha en egen entitet. Detta tycktes inte vara en stor prioritet på grund av att det är smidigare att användaren har en sträng i form av *URL*.



***Figur 5.1** - applikationens domänmodell*

5.6 Prototyp för användargränssnitt

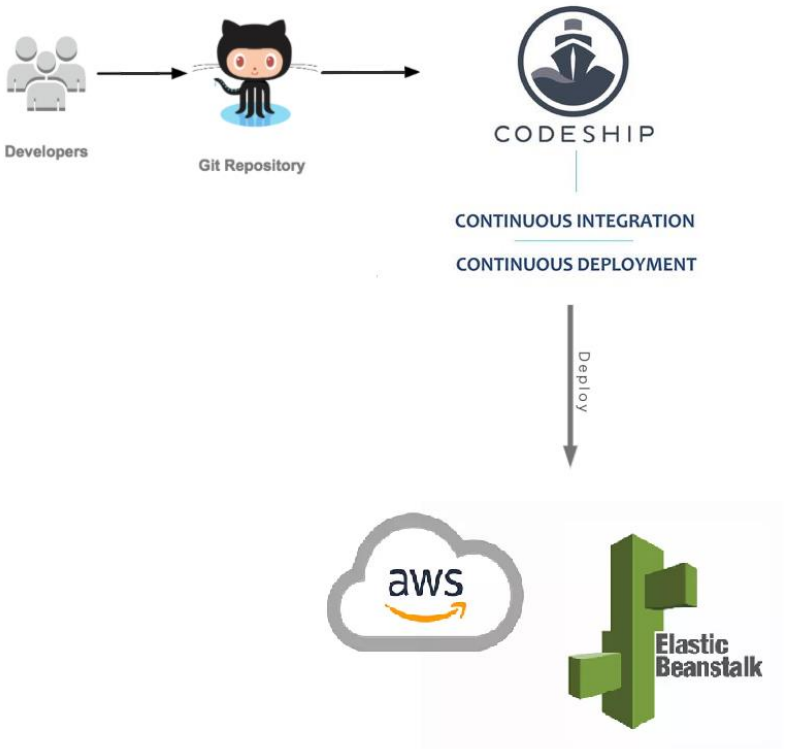
Med kravspecifikationen som utgångspunkt blev den tänkbara lösningen att skapa en preliminär prototyp med InVisionapp för applikationens användargränssnitt som figur 5.2 illustrerar. Prototypen är tänkt att fungera som ett underlag för att kunna följa hur det tänkta användargränssnittet ska utvecklas i senare skede. Därför skapades en skiss som omfattar en huvudsida med inloggningsformulär, en registreringssida, en egen sida för CV-hantering samt en administrationsdel där en administratör kan se alla Cv:n för de anställda.



Figur 5.2 - *Preliminär skiss av applikationens användargränssnitt*

6. Kontinuerlig integration

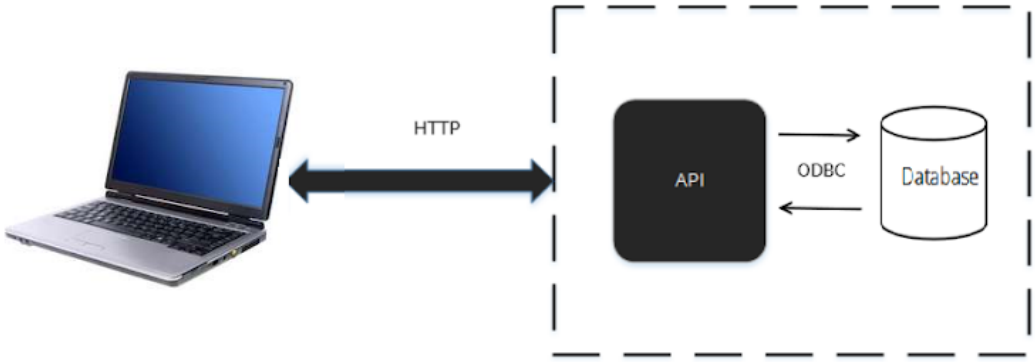
Kontinuerlig integration är en utvecklingspraxis som har använts redan från början i projektet. Första steget för att sträva efter kontinuerlig integration är att integrera kod i ett gemensamt Git Repository på GitHub med hjälp av versionshantering. I och med att projektet består av både back-end och front-end skapades två olika Git Repositories där meningen är att båda delarna eftersträvar kontinuerlig integration. Med detta sagt eftersöktes en lämplig plattform som kan distribuera senaste versionen av källkoden automatiskt från GitHub till den valda molntjänsten för projektet. Utifrån det som eftersöktes visade det sig att CodeShip var passande som fokuserar på snabbhet, tillförlitlighet och enkelhet [31]. Proceduren är helt enkelt att utvecklaren checkar in en förändring eller ett tillägg i källkoden och programvaran byggs automatiskt i CodeShip. När programvaran har byggts distribueras senaste versionen av källkoden till AWS då CodeShip stödjer kontinuerlig distribuering med en vald molntjänst. Figur 6.1 visar en klarare illustration av hur flödet för kontinuerlig integration har tillämpats i projektet.



***Figur 6.1** - Flödet för kontinuerlig integration*

7. Arkitektur beskrivning

Webbapplikationen kommer att kommunicera med klienten och databasen i ett färdigt system. Klienten skickar API-förfrågningar till back-end genom HTTP-förfrågningar (Request-Response). Kommunikationen mellan server(back-end) kommer i sin tur att ske med ODBC som visas i figur 7.1.



***Figur 7.1** - Arkitektur översikt*

8. Genomförande av back-end applikation

8.1 Kommunikations med relationsdatabas

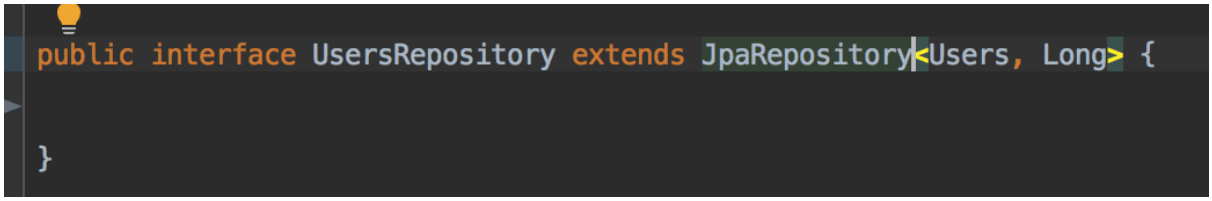
För att kunna skapa ett CV för en användare behöver domänmodellen som nämnts i tidigare kapitel att tillämpas. Informationen från användaren behöver lagras i en MySQL databas och för att få tillgång till användarens information används JPA. Figur 8.1 illustrerar hur ett användarobjekt kommer att lagras i en databas.

```
import ...
@Entity
public class Users implements UserDetails {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "id")
    private Long id;
    @Column
    private String currentTitle;
    @NotBlank(message = "Please enter full name")
    @Column(name = "fullName")
    private String fullName;
    @Email(message = "Username needs to be an email")
    @NotBlank(message = "Email is required")
    @Column(unique = true, length = 250)
    private String username;//this instance represents the ema
    @NotBlank(message = "Password is required")
    @Column(name = "password")
    private String password;
    @Column(name="Address")
    private String address;
    @Column(name = "PHONE")
    private String phone;
    @Column(name = "ADMINORUSER")
    private boolean adminOrUser;
    @Column(name = "UserProfile")
    private String userProfile;
    @Column(name = "Image")
    private String Image;
    private String theOwner;
    @OneToMany(cascade = CascadeType.ALL)
    @JoinColumn(name = "id", referencedColumnName = "id")
    private List<Workexperience> workExperience;
    @OneToMany(cascade = CascadeType.ALL)
    @JoinColumn(name = "id", referencedColumnName = "id")
    private List<Courses> courses;
    @OneToMany(cascade = CascadeType.ALL)
    @JoinColumn(name = "id", referencedColumnName = "id")
    private List<Education> educations;
    @OneToMany(cascade = CascadeType.ALL)
    @JoinColumn(name = "id", referencedColumnName = "id")
    private List<Others> others;
    @OneToMany(cascade = CascadeType.ALL)
    @JoinColumn(name = "id", referencedColumnName = "id")
    private List<Skills> skills;
    @OneToMany(cascade = CascadeType.ALL)
    @JoinColumn(name = "id", referencedColumnName = "id")
    private List<AboutFedgeIT> aboutFedgeIT;
    @OneToMany(cascade = CascadeType.ALL)
    @JoinColumn(name = "id", referencedColumnName = "id")
    private List<Languages> languages;
```

***Figur 8.1** - Översikt av ett användarobjekt i java.*

JPA annotationer används för att kunna lagra data i en databas. @Entity är den JPA-annotation som gör att användarobjektet mappas till MySQL databasen.
Id, currentTitle, fullName, username, password, address, phone, adminOrUser, image är attributen som används i användarobjektet och dessa är markerade med @column annotation som representerar namnen för kolumnerna i tabellen.
Andra JPA annotationer som har använts är @JoinColumn för att koppla tabellen för användarobjektet med andra tabeller som består av education, skills, workexperience.
Mellan användare och ovan nämnda tabeller används relationer i form @OneToMany just för att en användare refererar till flera andra tabeller.

Utöver att Spring Data JPA fokuserar på att spara data i en databas, är dess mest övertygande funktion att skapa repository implementationer automatiskt vid körtid från ett repository gränssnitt. Som ett exempel illustrerar figur 8.2 hur ett repository gränssnitt har skapats som fungerar med ett användarobjekt.



***Figur 8.2** - Repository gränssnitt.*

Detta gränssnitt utökar JpaRepository. Typen av domännamnet och Id som det fungerar med, Users och Long, specificeras i de generella parametrarna på JpaRepository. Genom att utöka JpaRepositroy, förvärvar JpaRepository flera metoder som stöder läsning, uppdatering och radering mot en back-end databas. Exempel på metoder som använts är Save, delete, findAll. Samtliga för att kunna spara och ta bort information till databasen samt läsa information från databasen.

8.2 API anrop

I Springs tillvägagångssätt hanteras HTTP-förfrågningar av en “controller” för att på så sätt skapa RESTful API:er. Dessa komponenter identifieras av en @RestController-annotation. @RequestMapping-annotationen kan tillämpas på en klassnivå där mappnings strategin är att mappa ett specifikt URI-mönster till “controller” klassen eller appliceras på metodnivå där mappnings strategin är att mappa särskild HTTP-metod för varje metod i “controller” klassen. Figur 8.3 visar hur UsersController hanterar GET-förfrågningar, helt enkelt genom att returnera User objektet och objektdata skrivs direkt till HTTP-svaret som ett kompakt, textbaserat format (JSON).

```
@RestController
@CrossOrigin
@RequestMapping("api/users")
public class UsersController {
    @Autowired
    private UserService userServices;
    @GetMapping(value = "/all")
    public Iterable<Users> getAll(Principal principal) {
        return userServices.getAllUsers(principal.getName());
    }

    @GetMapping("{us_id}")
    public ResponseEntity<?> getUsByID(@PathVariable Long us_id, Principal principal){

        Users users = userServices.findById(us_id, principal.getName());
        return new ResponseEntity<Users>(users, HttpStatus.OK);
    }
```

***Figur 8.3** - UsersController som hanterar GET-förfrågningar.*

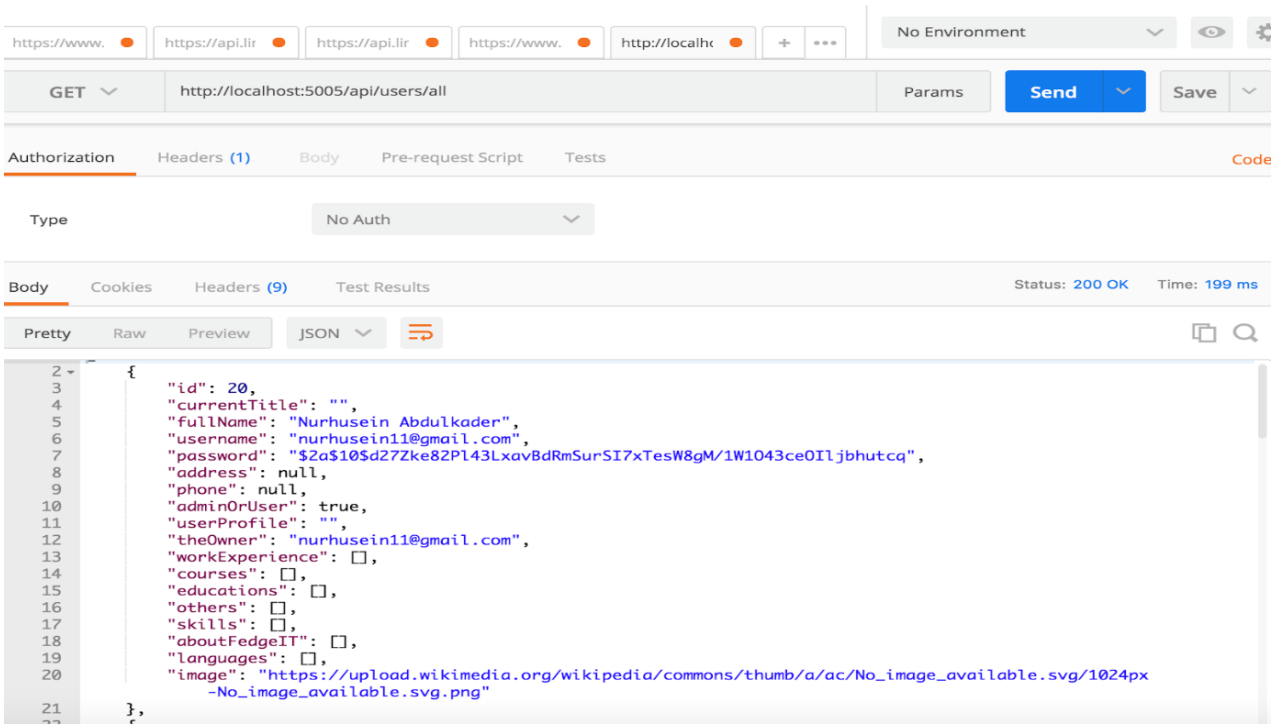
@RequestMapping-annotationen på klassnivå använder en URI, api/users, som delegerar alla förfrågningar under api/users URI till klassens metoder. Metoderna i klassen använder

sig av @GetMapping. När en HTTP GET-förfrågan görs kan en av de två metoderna att påkallas beroende på hur förfrågan utförs. Exempelvis påkallas getAll metoden när en HTTP GET-förfrågan görs på api/users/**all.** Utöver @GetMapping-annotation använder getUsByID metoden @PathVariable för att specifiera värdet av @GetMapping, som kommer att skicka in det värde som matas in i URL för metoden. getUsByID deklarerar @PathVariable Long us_id, Principal principal. Om en HTTP-förfrågning är i form av api/users/1 kommer metoden att få tillgång till us_id variabeln med värdet för första användaren i databasen.

Spring stöder flera typer av inbyggda annoteringar för hantering av olika typer av inkommande HTTP-förfrågningsmetoder. @PostMapping, @PutMapping och @DeleteMapping, som motsvarar POST, PUT och DELETE anrop specificeras inte i figur 8.3. Dock har dessa också implementerats och logiken fungerar på liknande sätt som GetMapping.

8.3 Testning och verifikation

För att säkerhetsställa funktionalitet i REST API:er har dessa testats med hjälp av Postman som är ett verktyg för att testa RESTful API:er. Varje RESTful API testas genom att skicka en begäran till webbservern med ett specifikt URL som är unikt för varje API och få svar tillbaka för verifikation. GET, POST, DELETE och PUT är de förfrågningsmetoder som använts för att testa RESTful API:er. För att testa om data kan hämtas från servern har GET förfrågningar använts. POST förfrågningar har använts för att testa om information i form av data har lyckats att läggas till på servern. DELETE förfrågningar har använts för att testa om information kan tas bort från servern. Därmed har PUT förfrågningar använts för att testa om data kan uppdateras på servern. Figur 8.4 illustrerar hur funktionaliteten av ett RESTful API testas och verifieras genom att GET förfrågan skickas till webbservern med URL, http://localhost:5005/api/users/all.

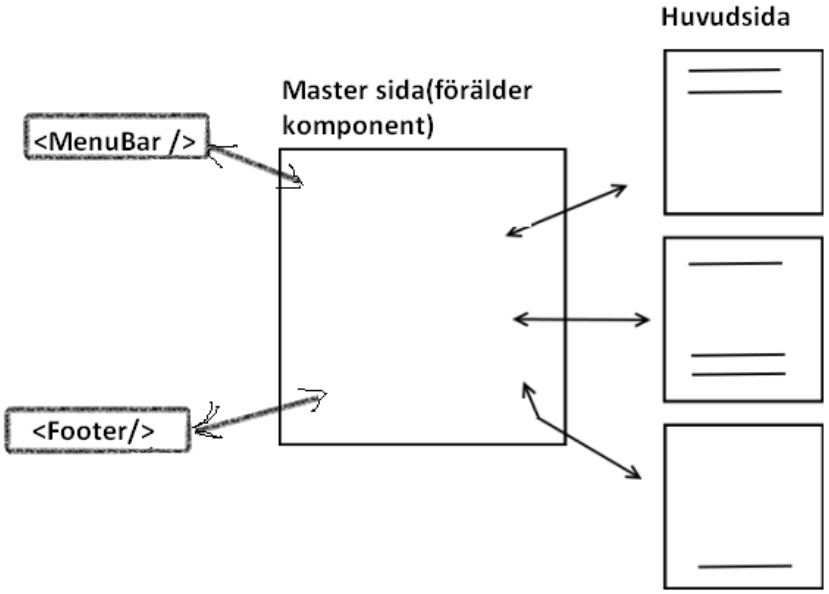


***Figur 8.4** - Testning av RESTful API med GET-förfrågningsmetod.*

9. Genomförande av front-end applikation

9.1 Master sida

Användargränssnittet för applikationen är uppdelad i flera enskilda bitar som är komponenter. Dessa utvecklas separat och sammanfogas i en Master sida eller så kallad förälder komponent som representerar hela användargränssnittet för applikationen när den körs. En huvudsida utformades i form av en komponent i React som är en av applikationens byggsten. Meningen med huvudsidan är att representera applikationens första sida med ett användargränssnitt som ger enkel vägledning för att komma åt applikationens funktioner och gå vidare till andra sidor. Genom att matcha komponenternas identifierare kan huvudsidan på så sätt kopplas med andra sidor. Figur 9.1 illustrerar hur huvudsidan sammanfogas i en förälder komponent. På det viset vet React var innehållet ska hamna när webbsidan genereras.



***Figur 9.1** - Samlingsplats för alla komponenter(sidor) som utformar applikationens användargränssnitt.*

9.2 Design för användargränssnitt

I React är det väldigt smidigt att designa ett användargränssnitt med få rader HTML- och CSS kod. Varje enskild sida har kodats med HTML- och CSS för att forma ett användargränssnitt utifrån prototypen som skapades i kapitel 5.6. För att göra applikationens utseende mer attraktivt samtidigt som att upplevelsen blir användarvänlig har ramverket Bootstrap tillämpats. Bootstrap underlättar webbutvecklingen genom att erbjuda en del HTML- och CSS-baserade designmallar för bland annat knappar, tabeller, ikoner, positionering och formulär som är färdiga för användning. Figur 9.2 illustrerar hur en sida för applikationens användargränssnitt har programmerats med HTML och CSS samt tillämpning av Bootstrap. I

figur 9.2 är “card-body” ett exempel på ett kort som hämtats från Bootstrap vilken ger en flexibel och utdragbar innehållsbehållare.

```
<div className="container">
    <div className="row">
        <div className="col-md-8 m-auto">
            <div className="card border-dark mb-3">
                <div className="card-header">
                    <h1 className="display-4 text-center">Edit education</h1>
                </div>
                <div className="card-body">
                    <form onSubmit={this.onSubmit}>
                        <div className="input-group" style={{padding: '5px'}}>
                            <div className="input-group-prepend">
                                <span className="input-group-text" id=""
                                      style={{width: '90px'}}>Title
                                </span>
                            </div>
                            <input type="text" className="form-control"
                                   placeholder=" title"
                                   name="title"
                                   value={this.state.title}
                                   onChange={this.onChange}/>
                        </div>
```

***Figur 9.2** - Design för en av applikationens sida med HTML, CSS samt Bootstrap.*

9.3 React Router

I React finns en enda HTML-fil inblandad vilket gör att applikationen blir begränsad till en enkel sida som kan visas. Med en enkelsidig applikation som innehar en enda sida hade inte kravspecifikationen uppfyllts då applikationen behöver mer än en sida. Reacts routing bibliotek, “React Router” har använts som gör det möjligt att bygga en enkelsidig webbapplikation som innehar flera komponenter med navigation samt utan siduppdatering när användaren navigerar. Tack vare navigationen återanvänds samma html sida som visar olika komponenter exempelvis huvudsidan i figur 9.1, beroende på navigeringen. Detta förhindrar siduppdateringar som istället ger en bättre användarupplevelse i form av sömlös övergång mellan olika sidor. React router har också använts för att göra det möjligt för användaren att använda webbläsarens funktionalitet som uppdateringsknappen och bakåtknappen samtidigt som den korrekta visningen av applikationen upprätthålls. Figur 9.3 illustrerar hur routing används för att kunna kartlägga webbadresser till destinationer som inte är fysiska sidor utan närmare de enskilda sidorna i den enkelsida webbapplikationen.

```
<Router>
    <div className="App">

        <Header/>

        {
            //Public Routes
        }
        /*HuvudSida*/
        <Route exact path="/" component={Login}/>
        <Route exact path="/Register" component={Register}/>
        <Route path="/ForgotPasswordPage " component={ForgotPasswordPage}/>
```

Figur 9.3 *- Rutter för olika sidor i applikationen.*

Alla "rutter" definieras i mastersidan och befinner sig inom <Router>. Detta för att när URL-adressen ändras i webbläsaren kommer </Router/> byta till en annan komponent för varje ny URL-begäran. Med detta sagt får användaren en uppfattning om att webbläsaren skiftar mellan olika sidor vid en ny URL-begäran. Men i verkligheten är det en enda sida som visas där React Router väljer vilken komponent som ska visas och vilken som ska döljas beroende på vad som efterfrågas i webbläsarens adressfält.

För att erbjuda användaren en enkel navigering genom applikationens sidor används länkar så kallade "Link" från React Router för att undvika att skriva in specifika URL-adresser i adressfältet. I figur 9.4 visas hur Link implementerats som navigerar en användare till sidan för glömt lösenord (/ForgotPasswordPage) samtidigt som URL kommer att ändras utan siduppdatering.

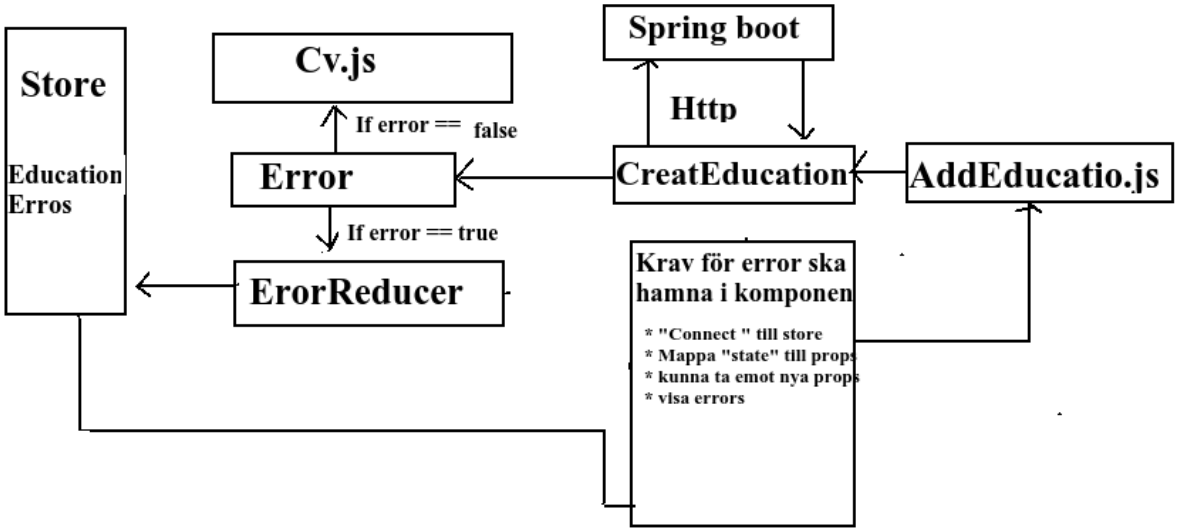
```
<Link className="fa" to={`/ForgotPasswordPage`} style={{padding: '20px'}}>
    Forgot password
</Link>
```

Figur 9.4 *- Navigeringslänk för applikationen.*

9.4 Redux

För att underlätta dataflödet- och tillståndet i applikationen har Redux använts. I figur 9.5 ges ett exempel för hur arkitekturen har använts för att tillämpa Redux i applikationen. AddEducation.js är en komponent som har använts för att kunna lägga till utbildningar till Cv:t. AddEducation.js anropar en händelse med en funktion, "createEducation", där http förfrågan skickas till back-end (Spring Boot). Detta för att interagera med RESTful API:er som har skapats i Spring Boot. Om förfrågningen lyckas kommer back-end att svara med det Education objekt som har skapats och detta kommer i sin tur att kunna läggas till CV sidan (CV.js). CV sidan erhåller all information om en användare som visas. Vid misslyckande kommer Spring Boot att svara med ett felmeddelande eller så kallad "error". Dessa errors

kommer att hamna i “Store”. AddEducation.js kommer i sin tur fånga alla errors som har sparats i Store och visa upp de.

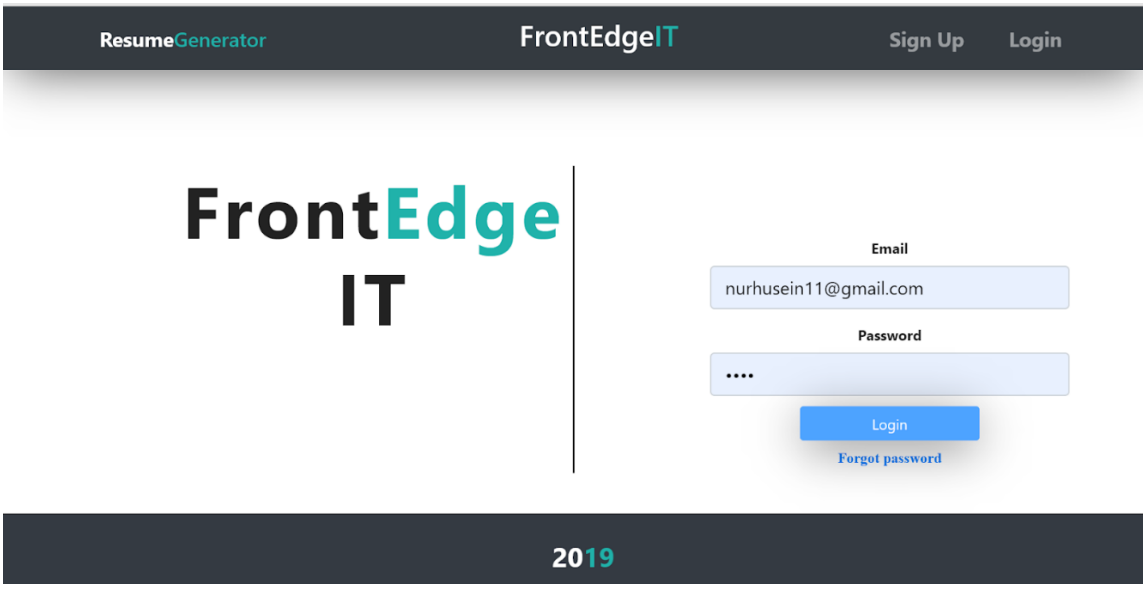


Figur 9.4 - *Översikt över arkitektur för att tillämpa Redux.*

10. Interaktion mellan back-end- och front-end applikation

10.1 Inloggningssida

En inloggningssida utformades som är huvudsidan för applikationen, illustreras i figur 10.2. För att en administratör eller vanlig användare ska kunna logga in på systemet sker en validering. Detta genom att skicka en http-förfrågan från front-end till back-end applikationen. Utöver det kontrolleras även att de inmatade textsträngarna under email-fältet i figur 10.1 möter krav i form av att klienten måste skriva in en sträng som är en mailadress.



Figur 10.1 *- Webbapplikationens inloggningsportal.*

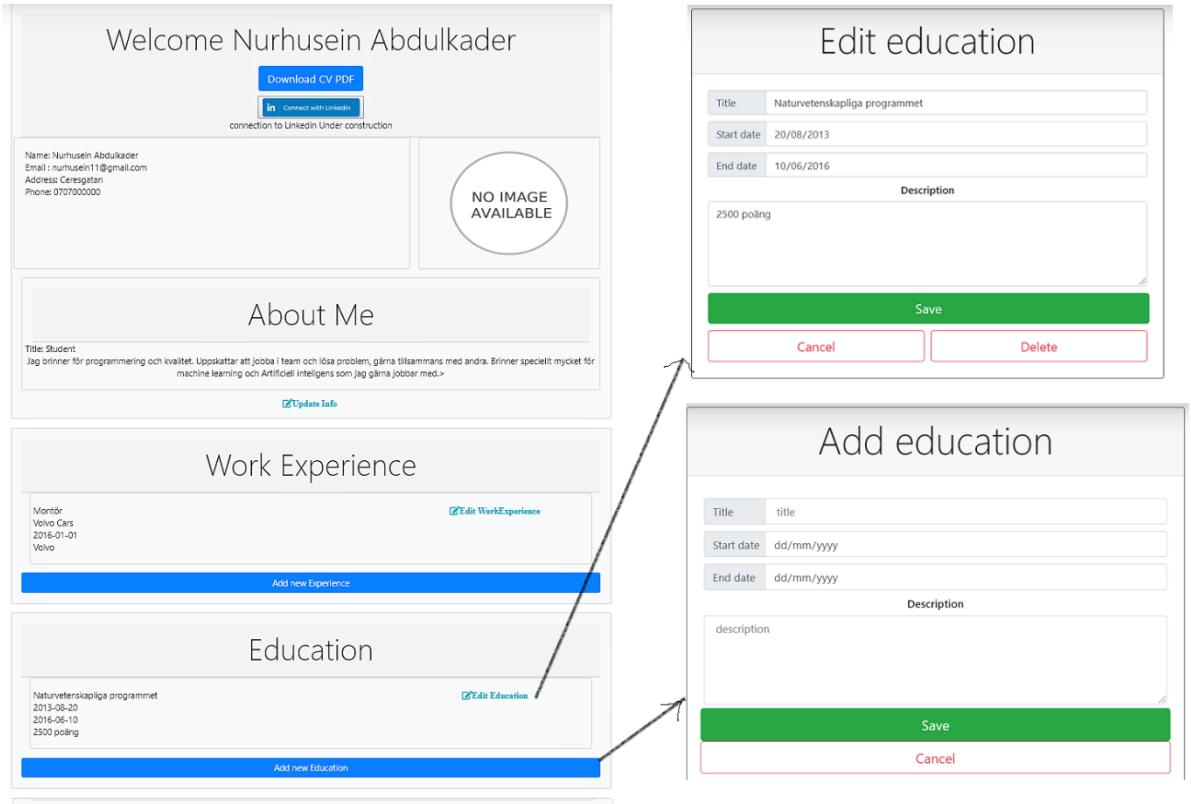
Figur 10.2 visar koden som exekveras när en http-förfrågan sker från front-end applikationen. back-end tar emot förfrågan och svarar genom att skicka tillbaka data för en användare om indatan från front-end som är mejladressen och lösenordet är korrekt. Annars skickas ett http-svar bestående av ett felmeddelande. Lösenord krypteras med hjälp av en färdig algoritm, så kallat BCryptPasswordEncoder, som kastar om textsträngar till oläsbara tecken.

```
export const login = LoginRequest => async dispatch => {
    try {
        const res = await axios.post( url: `${BASE_URL}/users/auth/login`, LoginRequest);
        dispatch({
            type: SET_CURRENT_USER,
            payload: res
        });
    } catch (err) {
        dispatch({
            type: GET_ERRORS,
            payload: err.response.data
        });
    }
};
```

Figur 10.2 *- Kod som exekveras vid http-förfrågan.*

10.2 CV sida

CV sidan utformades för att sammanställa information för en administratör eller vanlig användare i form av kunskaper, erfarenheter och färdigheter. Varje box är en React komponent som består av huvudrubrikerna “Welcome”, “About me”, “Work Experience” och “Education” som illustreras i figur 10.3. I första komponenten kan användaren generera ett CV i pdf och komma åt Linkedin. Förutom första komponenten kan information läggas till, uppdateras eller tas bort i alla komponenter.



Figur 10.3 *- Webbapplikationens CV-sida.*

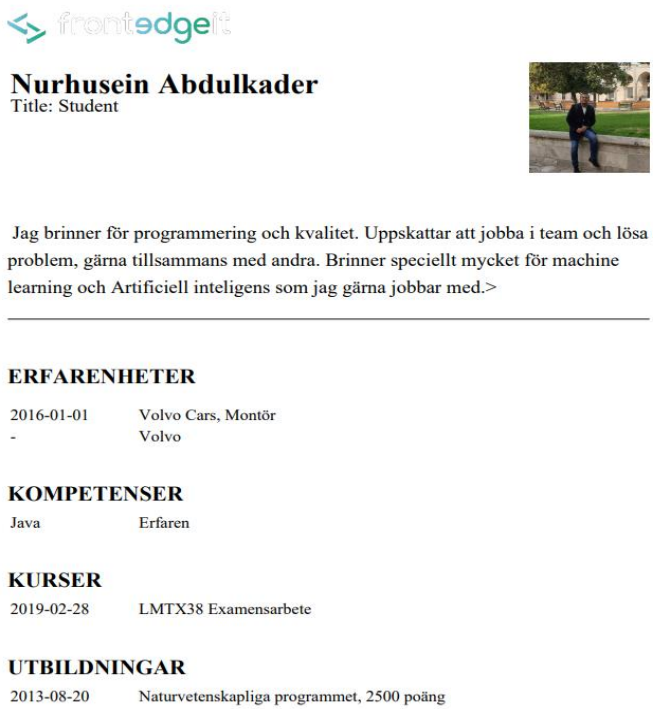
Beroende på den inloggade användarens id körs en JavaScript funktion som utför en http-förfrågan för att kunna hämta den data som innehar användarens information. Hur detta går till när koden exekveras illustreras i figur 10.4. Back-end tar emot användarens id och skickar tillbaka data som tillhörande en användare som visas på front-end.

```
export const getUsersById = (id) => async dispatch => {
    const res = await axios.get( url: `${BASE_URL}/users/${id}`);
    dispatch({
        type: FETCH_USER,
        payload: res.data
    });
};
```

Figur 10.4 *- Kod som exekveras vid http-förfrågan för att hämta data för en användare.*

10.2.1 Utformning av CV

PDF-generering är en funktion som är skapad för att varje användare ska ha möjlighet att ladda ner sitt CV i PDF-format. PDF styling hanterades i back-end med IText för att kunna manipulera innehållet på ett sätt som passar företagets mall vilken illustreras i figur 10.5



Nurhusein Abdulkader
Title: Student

Jag brinner för programmering och kvalitet. Uppskattar att jobba i team och lösa problem, gärna tillsammans med andra. Brinner speciellt mycket för machine learning och Artificiell inteligens som jag gärna jobbar med.>

ERFARENHETER

2016-01-01	Volvo Cars, Montör
-	Volvo

KOMPETENSER

Java	Erfaren

KURSER

2019-02-28	LMTX38 Examensarbete

UTBILDNINGAR

2013-08-20	Naturvetenskapliga programmet, 2500 poäng

Figur 10.5 *- CV i PDF som laddas ner från applikationen.*

10.3 Administratör sida

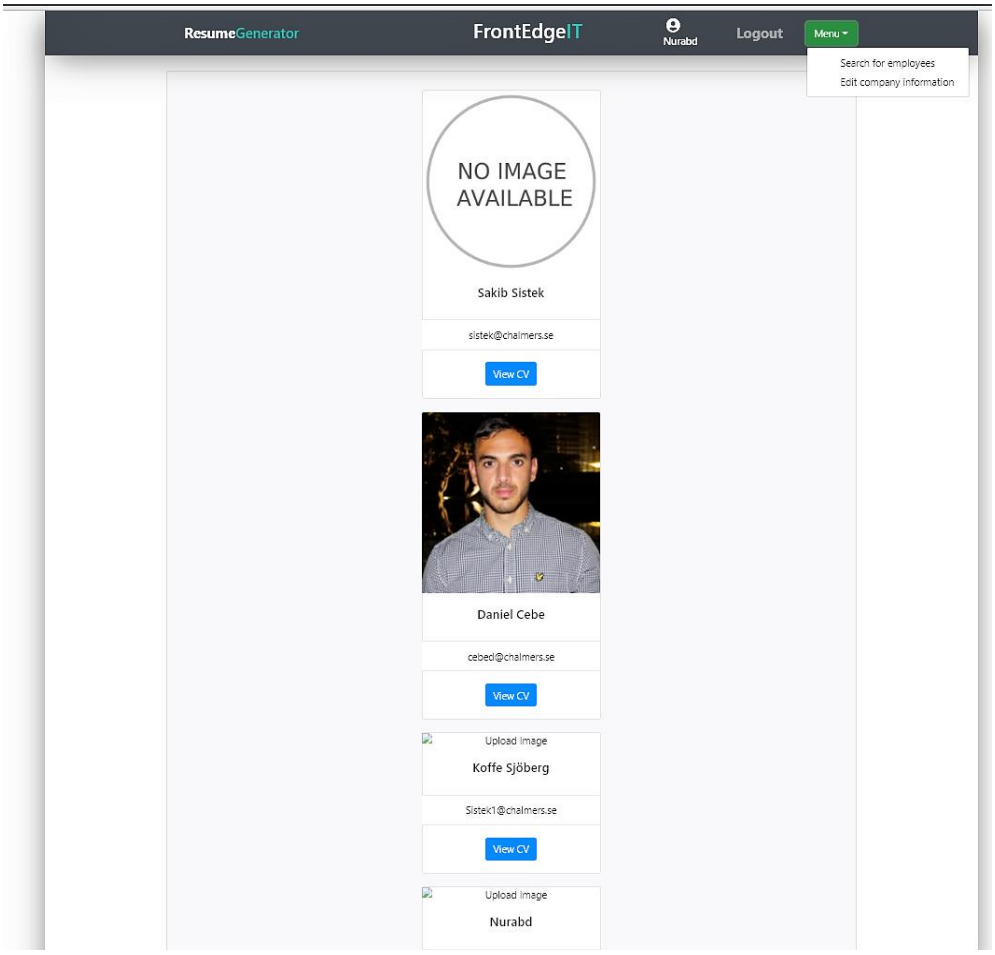
Vid inloggning som administratör får man tillgång till alla användare som är registrerade i systemet. Detta sker genom att en JavaScript-funktion körs. Funktionen skickar en http-förfrågan till back-end och hämtar alla användare. Kodavsnittet i figur 10.6 illustrerar hur

front-end skickar en förfrågan till back-end för att hämta data som i detta fall är alla användare.

```
export const fechUsers = () => async dispatch => {
    const res = await axios.get( url: `${BASE_URL}/users/all`);
    dispatch({
        type: FETCH_ALL_USER,
        payload: res.data
    });
};
```

Figur 10.6 *- JavaScript-funktion för att hämta alla användare i systemet.*

För att på något sätt ge en administratöröversikt över alla användare som hämtats, har en egen administratör sida utformats som illustreras i figur 10.7. Administratör sidan ger åtkomst till alla användare där bild, namn och mailadress visas för respektive användare. Utöver det kan administratören komma åt användarens CV-profil genom att klicka på “view CV” och därmed kunna ladda ner CV för en specifik användare. Men även ändra, ta bort och lägga till information för en användare.



Figur 10.7 *- Webbapplikationens administratör sida.*

11. Resultat

Arbetet resulterade i en fungerande webbapplikation med ett grafiskt användargränssnitt och en serverdel med tillhörande databasstruktur. Webbapplikationen utvecklades i WebStorm för att hantera det grafiska användargränssnittet med React och Redux. Dessutom utvecklades Webbapplikationen i IntelliJ med Spring Boot där endast bearbetning på servernivå hanterades med MySQL som databashanteringssystem. Relationer definierades i databasen med hjälp av JPA för att förse applikationen med information som täcker allt en användare behöver för att forma ett komplett CV.

Systemet uppfyller nedanstående funktionella krav:

- En administratör eller vanlig användare kan logga in i systemet.
- Skapa ett CV genom att lägga till information i systemet som sparas till databasen.
- Lägga till, ändra, ta bort information från sitt CV
- Ladda ner CV i PDF format från applikationen.
- Glömt lösenord vid inloggning

Systemet uppfyller nedanstående Icke-funktionella krav:

- Systemet ska köras på Amazon Web Services och ska kunna vara åtkomlig när som helst, var som helst.
- Systemet ska kräva inloggning som styr vilka funktioner en administratör eller användare kan använda.
- Flexibelt system som gör det möjligt vid eventuell vidareutveckling av systemet.

12. Slutsats

12.1 Kritisk diskussion

Det fanns redan från början en uppfattning om att uppdraget för att utveckla en molnbaserad webbapplikation från grunden skulle innebära ett hårt arbete med mycket nedlagd tid. Dessutom innebar den saknade erfarenheten av att arbeta med molnplattform samt back-end och front-end separat att detta projekt kunde antas ta mer tid än vad som planerats. Så blev fallet och medförde att två funktionella krav inte implementeras på grund av tidsbrist. Nedladdning av CV i Word format och möjligheten för en administratör att bjuda in användare via inbjudningslänk är de funktioner som inte har implementerats. Dock hann de mest prioriterade funktionerna att implementeras i tid. Med detta sagt följdes tidsplanen som väntat fram till sista veckan. Då insågs att planeringen för att implementera alla funktioner inte skulle uppfyllas helt. Tidsplanen för sista veckan fick ändras då veckan ägnades åt att spendera mer tid till att refaktorera systemet för att på bästa sätt anpassa det till att andra utvecklare har möjlighet att förstå koden för vidareutveckling av systemet.

Då projektet har omfattats av både back-end och front-end, har koden för användargränssnitt och funktionalitet varit lätt att separera på ett bra sätt. Trots detta har det försvårat möjligheterna att hinna med att implementera automatiserade tester såsom JUnit tester för den funktionella koden då det har varit svårt att fördela vikten av utvecklandet mellan back-end och front-end. Att jobba fullstack, det vill säga med både back-end och front-end visade sig vara väldigt tidskrävande speciellt när man jobbar med olika ramverk och två olika programmeringsspråk. Tiden som avsattes för testning användes istället till ytterligare implementering i både back-end och front-end. Därmed möjliggjorde det att webbapplikationen utvecklades till den grad som beskrivs i resultatdelen. I och med att någon form av testning ändå är viktigt för att säkerställa funktion och kvalitet har istället manuella tester under utvecklingen av applikationen utförts med hjälp av Postman. I detta fall är projektet alltså inte beroende av automatiska tester. För att på ett enkelt sätt kunna skriva automatiserade tester hade det rätta valet varit att använda sig av ramverket JUnit som Spring Boot erbjuder. Dock hade inlärningstiden blivit ännu längre för att bekanta sig med ramverket för att skriva automatiserade tester. På grund av tidsramen för projektet skulle applikationens funktionaliteter begränsas ytterligare om testerna skulle implementeras.

Jämfört med det nuvarande systemet som företaget använder för att hantera Cv:n, har det system som utvecklats med molnbaserade lösning övervägande antal fördelar. Det nya systemet ger alla Cv:n garanterad enhetlighet, ett stort underlättande för chefen att ha koll på alla Cv:n för anställda samt alltid tillgång till att ladda ner senaste versionen av Cv:n för en anställd.

12.2 Tekniken och miljön

Molntjänster bidrar negativt till miljön och är bland annat väldigt energikrävande och bidragande till koldioxidutsläpp. Med åtanke till detta hostas applikationen endast vid dagtid mellan 7–16. Utöver dessa tider är applikation inte i drift [32].

12.3 Vidareutveckling av webbapplikation

Webbapplikationen har förmåga att utvecklas i stor omfattning. Flera funktioner och förbättringar har diskuterats under projektets gång, men tiden har inte räckt till för att uppnå alla förbättringar samt möjliga implementationer. När det gäller förbättringar kan användargränssnittet utvecklas. Detta genom att lägga till ikoner på knappar och inmatningsfält, positionering av knappar samt bättre anpassning av upplösning till användarens skärm med CSS. Nedan följer ett par funktioner som inte blivit implementerade under projektet, men skulle kunna vara intressanta att implementera vid ett fortsatt arbete och skulle förbättra applikationen:

- Implementera en funktion som gör det möjligt att ladda ner sitt CV även i WORD format.
- Lägga till en egen profilsida för användaren där möjlighet för namn, lösenord och mejladress kan ändras istället för att allt ska ligga på CV sidan.
- Strukturera om koden, för möjliggörande av JUnit tester
- Implementera funktion för uppladdning av bild från filsystemet utöver befintliga funktion med uppladdning av personlig bild från URL.
- Kunna skicka inbjudningslänk via mejl till anställda på företaget för att registrera ett konto på applikationen.

Referenser

[1] Soni, R. (2015). *Learning Spring Application Development*. Birmingham: Packt Publishing.

[2] Baeldung. (2019). *Spring Boot Tutorial - Bootstrap a Simple App | Baeldung*. [online] Available at: https://www.baeldung.com/spring-boot-start [Accessed 8 Jun. 2019].

[3] Websystem.se. (2019). *REST-API:er | Websystem en teknikbyrå som jobbar med Drupal*. [online] Available at: https://websystem.se/sv/rest-apier [Accessed 29 May 2019].

[4] Rouse, M. (2019). *What is RESTful API? - Definition from WhatIs.com*. [online] SearchMicroservices. Available at: https://searchmicroservices.techtarget.com/definition/RESTful-API [Accessed 29 May 2019].

[5] Spring.io. (2019). *Spring Projects*. [online] Available at: https://spring.io/projects/spring-data [Accessed 5 Jun. 2019].

[6] Skobalj, V. (2018). *Guide to Spring Data JPA*. [online] Stack Abuse. Available at: https://stackabuse.com/guide-to-spring-data-jpa/#disqus_thread [Accessed 5 Jun. 2019].

[7] www.tutorialspoint.com. (n.d.). *iText Overview*. [online] Available at: https://www.tutorialspoint.com/itext/itext_overview.htm [Accessed 8 Jun. 2019].

[8] Amazon Web Services, Inc. (2019). *What is Continuous Integration? – Amazon Web Services*. [online] Available at: https://aws.amazon.com/devops/continuous-integration/ [Accessed 8 Jun. 2019].

[9] Domes, S. (2017). *Everything You Should Know About React: The Basics You Need to Start Building*. [online] freeCodeCamp.org News. Available at: https://www.freecodecamp.org/news/everything-you-need-to-know-about-react-eaedf53238c4/ [Accessed 25 May 2019].

[10] LogRocket Blog. (2018). *Why use Redux? Reasons with clear examples - LogRocket Blog*. [online] Available at: https://blog.logrocket.com/why-use-redux-reasons-with-clear-examples-d21bffd5835/ [Accessed 26 May 2019].

[11] Redux.js.org. (2018). *Reducers · Redux*. [online] Available at: https://redux.js.org/basics/reducers
[Accessed 26 May 2019].
[12] Azure.microsoft.com. (n.d.). *What is cloud computing? A beginner's guide | Microsoft Azure*. [online] Available at: https://azure.microsoft.com/en-in/overview/what-is-cloud-computing/ [Accessed 24 May 2019].

[13] Amazon Web Services, Inc. (n.d.). *What is AWS*. [online] Available at: https://aws.amazon.com/what-is-aws/ [Accessed 8 Jun. 2019].

[14] Amazon Web Services, Inc. (n.d.). *Benefits at a glance*. [online] Available at: https://aws.amazon.com/application-hosting/benefits/ [Accessed 8 Jun. 2019].

[15] Joshi, V. (2018). Council Post: Seven Reasons Why A Website's Front-End And Back-End Should Be Kept Separate. [online] Forbes.com. Available at: https://www.forbes.com/sites/forbestechcouncil/2018/07/19/seven-reasons-why-a-websites-front-end-and-back-end-should-be-kept-separate/#73616ad94fca [Accessed 14 May 2019].

[16] Plebani, A. and Plebani, A. (2019). *Web development comparison: SpringBoot vs ExpressJS*. [online] Devamountain.com. Available at: https://www.devamountain.com/post/web-development-comparison-springboot-vs-expressjs/ [Accessed 15 Sep. 2019].

[17] Monterail.com. (2019). *Vue vs. React in 2019: Which Framework to Choose and When*. [online] Available at: https://www.monterail.com/blog/vue-vs-react-2019 [Accessed 15 Sep. 2019].

[18] Timokhina, V. (2017). *Why Choose ReactJS for your Next Project | Eastern Peak*. [online] Eastern Peak. Available at: https://easternpeak.com/blog/why-choose-reactjs-for-your-next-project/ [Accessed 10 May 2019].

[19] Redux.js.org. (n.d.). *Redux · A Predictable State Container for JS Apps*. [online] Available at: https://redux.js.org/ [Accessed 12 May 2019].

[20] Amazon Web Services, Inc. (n.d.). *AWS Elastic Beanstalk – Deploy Web Applications*. [online] Available at: https://aws.amazon.com/elasticbeanstalk/ [Accessed 13 May 2019].

[21] Docs.aws.amazon.com. (n.d.). *Adding a Database to Your Elastic Beanstalk Environment - AWS Elastic Beanstalk*. [online] Available at: https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/using-features.managing.db.html [Accessed 13 May 2019].

[22] Amazon Web Services, Inc. (n.d.). *Amazon Relational Database Service (RDS) – AWS*. [online] Available at: https://aws.amazon.com/rds/?ft=n&trk=ft_card&trk=ft_card [Accessed 15 May 2019].

[23] InVision. (n.d.). *InVision | Digital product design, workflow & collaboration*. [online] Available at: https://www.invisionapp.com/ [Accessed 19 May 2019].

[24] Dbdiagram.io. (2018). *dbdiagram.io - Database Relationship Diagrams Design Tool*. [online] Available at: https://dbdiagram.io/home [Accessed 19 May 2019].

[25] Dev.mysql.com. (n.d.). *MySQL :: MySQL 8.0 Reference Manual :: 1.3.1 What is MySQL?*. [online] Available at: https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html [Accessed 19 May 2019].

[26] Git-scm.com. (2019). *Git*. [online] Available at: https://git-scm.com/ [Accessed 19 May 2019].

[27] Inc., C. (2019). *Continuous Integration, Deployment & Delivery with Codeship*. [online] Codeship.com. Available at: http://codeship.com/ [Accessed 19 May 2019].

[28] JetBrains. (2019). *IntelliJ IDEA: The Java IDE for Professional Developers by JetBrains*. [online] Available at: https://www.jetbrains.com/idea/ [Accessed 19 May 2019].

[29] JetBrains. (2019). *WebStorm: The Smartest JavaScript IDE by JetBrains*. [online] Available at: https://www.jetbrains.com/webstorm/ [Accessed 19 May 2019].

[30] Postman. (2019). *Postman API Development Environment | Learn More & Download*. [online] Available at: https://www.getpostman.com/postman [Accessed 19 May 2019].

[31] Inc., C. (2019). *Continuous Integration, Deployment & Delivery with Codeship*. [online] Codeship.com. Available at: http://codeship.com/ [Accessed 19 May 2019].

[32] Binero.Cloud. (2018). *”Smutsig data” ett snabbt växande problem – men det finns hopp | Binero.Cloud*. [online] Available at: https://binero.cloud/smutsig-data-ett-snabbt-vaxande-problem-men-det-finns-hopp/ [Accessed 5 Jun. 2019].

Appendix A – Gantt-schema

Webbapplikation för CV generering i molnet