CHALMERS

GÖTEBORGS UNIVERSITET

inSource

Utveckling av ett socialt webbaserat arbetsverktyg för rekryterare inom data- och IT-branschen

Kandidatarbete inom Data- och Informationsteknik

Martin Chemander
Oskar Holmberg
Linus Johansson
Adrian Nilsson
Erik Rundén

CHALMERS UNIVERSITY OF TECHNOLOGY
GÖTEBORGS UNIVERSITET
Institutionen för Data- och informationsteknik
Göteborg, Sverige, Juni 2016

inSource
Utveckling av ett socialt webbaserat arbetsverktyg för rekryterare inom data- och IT-branschen

MARTIN CHEMANDER
OSKAR HOLMBERG
LINUS JOHANSSON
ADRIAN NILSSON
ERIK RUNDÉN

© MARTIN CHEMANDER, 2016.
© OSKAR HOLMBERG, 2016.
© LINUS JOHANSSON, 2016.
© ADRIAN NILSSON, 2016.
© ERIK RUNDÉN, 2016.

Examinator: Arne Linde

Kandidatarbete 2016:31

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

The Author grants to Chalmers University of Technology and University of Gothenburg the non-exclusive right to publish the Work electronically and in a non-commercial purpose make it accessible on the Internet. The Author warrants that he/she is the author to the Work, and warrants that the Work does not contain text, pictures or other material that violates copyright law.

The Author shall, when transferring the rights of the Work to a third party (for example a publisher or a company), acknowledge the third party about this agreement. If the Author has signed a copyright agreement with a third party regarding the Work, the Author warrants hereby that he/she has obtained any necessary permission from this third party to let Chalmers University of Technology and University of Gothenburg store the Work electronically and make it accessible on the Internet.

Institutionen för Data- och Informationsteknik
Göteborg 2016

inSource

Utveckling av ett socialt webbaserat arbetsverktyg för rekryterare inom data- och IT-branschen

Martin Chemander
Oskar Holmberg
Linus Johansson
Adrian Nilsson
Erik Rundén

Institutionen för Data- och Informationsteknik, Chalmers Tekniska Högskola och Göteborgs universitet

Kandidatarbete

Sammanfattning

Att hitta lämpliga personer med rätt kompetens till jobb inom data- och IT-branschen är inte alltid enkelt, varför Software Skills, ett rekryteringsföretag i Göteborg, har gjort det till sin affärsidé. Eftersom de använder egenutvecklade tester för att utvärdera kandidater har behovet av ett skräddarsytt arbetsverktyg där testerna kan utgöra en integrerad del av systemet vuxit sig stort.

Denna rapport beskriver utvecklingen av inSource, en prototyp av ett nytt webbaserat mjukvarusystem för rekrytering framtaget i nära samarbete med personalen på Software Skills. Resultatet är en prototyp som integrerats i Software Skills nya webbplattform. Prototypen består av ett meddelandesystem, hantering av enskilda kandidater samt hantering av alla kandidater som sökt till ett specifikt jobb. Även funktioner så som en aktivitetslogg, omnämningar och kandidatomdömen har implementerats för att främja samarbete mellan rekryterare.

Projektets stora utmaning har varit att utveckla prototypen utan vetskap om hur den slutgiltiga designen skulle komma att se ut. Designen utvecklades nämligen i ett separat projekt och färdigställdes inte förrän i det här projektets slutskede. Detta kombinerat med en begränsad tidsram innebar att det inte var möjligt att implementera den slutgiltiga designen med all önskad interaktion på alla sidor. Därför är prototypen i nuläget inte praktiskt användbar som arbetsverktyg. Däremot nådde meddelandesystemet en alfaversion, och Software Skills har även uttryckt att de kommer gå vidare i utvecklingen av prototypen.

Abstract

Finding and recruiting skilled software professionals is not always easy, which is why Software Skills, a recruitment company based in Gothenburg, dedicate themselves to this task. They make use of in-house developed tests to evaluate candidates, and the need for a tailored applicant tracking system that integrates with their tests has grown apparent.

The following report describes the development of inSource, a prototype of a web-based applicant tracking system created in close cooperation with Software Skills. The result is a prototype that has been integrated with Software Skills new website. The prototype consists of views and functions for finding and administrating candidates as well as an instant messaging (IM) service. Features such as an activity log, mentions and candidate reviews have also been implemented to support collaborative work between recruiters.

This project faced the challenge of developing the prototype without a finished design. The design and layout of the prototype was developed in another project and was not finalized until close to the end of this project. This, combined with the limited time available, meant that it was not possible to implement the final design with the desired functionality on every view. Therefore the prototype has limited use as a tool for recruitment in its present state. Nevertheless, the IM-service reached an alpha state and Software Skills has expressed an intention to further develop the prototype.

Förord

Projektet utfördes som ett kandidatarbete under våren 2016 av studenter från civilingenjörsprogrammet i datateknik vid Chalmers tekniska högskola och programmet Datavetenskap vid Göteborgs Universitet.

Gruppen vill rikta ett särskilt tack till Henrik Enström, ägare av Software Skills i Göteborg, som upprättat projektbeskrivningen och som både stöttat och trott på oss. Vi vill även passa på att tacka Luuk van Egeraat för kontinuerligt stöd under arbetets gång med de tekniska detaljerna och kommit med förslag till lösningar då gruppen fastnat, masterstudenterna Anna Weiss och Johannes Lundqvist för den fina design som vi sedan kom att implementera och slutligen vår handledare Emil Axelsson som kommit med råd och synpunkter under projekts gång.

Innehåll

Ordlista

API *Application Programming Interface*, en specifikation på hur applikationer kan kommunicera.

ATS *Applicant Tracking System*, ett system för hantering av jobb och sökande.

CSS *Cascading Style Sheets*, en stilmall för webbplatser.

Google Drive En molnbaserad tjänst för att spara data och dokument.

HTTP *HyperText Transfer Protocol*, ett protokoll för kommunikation över internet.

JSON *JavaScript Object Notation*, ett textbaserat sätt att spara data.

MVC *Model-View-Controller*, ett designmönster som ofta används vid utveckling av webbapplikationer.

PM *Personal Message*, privat meddelande mellan företag och kandidater.

Scrum En typ av agil arbetsmetod.

SPA *Single Page Application*, en applikation som dynamiskt ändrar utseende vid behov.

Trello En webbaserad projekthanteringsapplikation.

1. Inledning

I takt med att IT-eran fick sitt genombrott kom stora fundamentala bitar i rekrytering av arbetskraft att förändras. Från att tidigare varit en sekventiell process, där jobbannonser publicerades i tidningar och ansökningar hanterades i pappersformat finns nu helt andra möjligheter. E-rekrytering har kommit att bli en kontinuerlig process, som kan utföras varsomhelst, närsomhelst för såväl jobbsökande som rekryterare, där aktiviteter sker online och kan i viss mån även uträttas parallellt med varandra [1]. Dagens e-rekryteringssystem har utvecklats till att inkludera hantering av kandidater genom hela rekryteringsprocessen, med möjlighet att både erbjuda och avvisa sökande jobb [2]. Ett sådant system benämns ofta ATS, från engelskans *Applicant Tracking System*, eller ansökningshanteringssystem i en fri översättning.

Fördelarna med övergången till en onlinebaserad rekrytering inkluderar reducerade kostnader för företaget i fråga samt ökad potential att jobben når ut till en bredare publik, medan den största nackdelen är att fler otillräckliga eller ointressanta ansökningar kan komma skickas in [3]. En annan aspekt är att e-rekrytering i många fall reducerat tiden det tar från att en jobbmöjlighet uppdagats till en faktiskt anställning, exempelvis har företaget Dow Chemical lyckats reducera sin genomsnittliga rekryteringsprocess från 90 till 34 dagar [1].

Eftersom ansökningshanteringssystem ofta används av rekryterare som jobbar i lag [2] kan inte enbart människa-datorinteraktion tas i beaktande under utvecklingen av ett ATS. Hänsyn måste även tas till hur rekryterare kommunicerar med varandra för att skapa en så effektiv rekryteringsprocess som möjligt.

1.1 Bakgrund

Software Skills är ett Göteborgsbaserat rekryteringsföretag med affärsidén att kombinera specialanpassade tester och intervjuer för att hjälpa sina kunder hitta lämpliga och kvalificerade kandidater till tjänster inom data och IT. Deras kunder är svenska företag som önskar hitta rätt person till en viss tjänst.

Med en vision om en ny webbapplikation ser Software Skills en möjlighet att utgöra en

del av den digitala arbetsmarknaden. Ett hinder för denna vision är att delar av deras nuvarande arbetsprocess är beroende av tredjepartsprogrammen Google Drive och Trello, vilket i praktiken betyder att tre helt separata gränssnitt måste användas för administrering av rekryteringen. I takt med att företaget och mängden data de hanterar växer blir en tredjepartslösning som inte är anpassad helt efter deras behov väldigt ineffektiv, och till slut ohanterbar.

1.2 Syfte och mål

Projektets syfte är att dokumentera utvecklingen av inSource, en ATS-prototyp som i framtiden är tänkt att ersätta Software Skills tredjepartslösning för att hantera rekryteringsprojekt. Utöver utvecklingen av inSource syftar rapporten till att undersöka vilken funktionalitet som är central i en webbapplikation som främjar samverkan mellan användare, och mer specifikt hur denna funktionalitet appliceras på ett ansökningshanteringssystem.

Målet med projektet är att ersätta de externa tjänsterna Trello och Google Drive med en oberoende ATS-lösning och en tillhörande databas som integreras direkt i Software Skills befintliga webbplattform. Prototypen skall hantera alla delar av rekryteringsprocessen, allt från kommunikation och upprättande av jobbannonser till att hantera ett stort antal sökande mellan olika steg i ansökningsprocessen.

1.3 Avgränsningar

Vår ATS-prototyp är främst skapad för att fungera på datorskärmar och är inte anpassad för mobila enheter. Dagens stora variation i skärmstorlekar och alternativa inmatningsmetoder (pekskärm eller mus och tangentbord) har skapat nya designmässiga utmaningar för webbutvecklingen och blir ett särskilt stort hinder i detta tidsbegränsade projekt. Eftersom ett ATS är ett arbetsverktyg kommer applikationen främst användas i webbläsare på datorer och laptops. Det är därför av tidsintresse lämpligt att bortse från användning på mobila enheter i en första prototyp. Fokus ligger istället på att implementera funktionaliteten, d.v.s. användarens interaktion med hemsidan.

På grund av projektets omfattning har det delats upp i två delar: design och implementation. Detta kandidatarbete innefattar enbart implementationen, men vi kommer emellertid behöva resonera kring webbapplikationen som helhet. Själva designen tas fram av två masterstudenter från Interaktionsdesign, med uppgiften att skapa sidornas layout och grafiska profil (färgschema, typsnitt och dylikt) i form av wireframes. Det är denna design vi sedermera kom att implementera i kod. Således har ingen kod producerats av masterstudenterna.

2. Problembeskrivning

Problemet som projektet ska lösa är att skapa ett webbaserat arbetsverktyg till rekryterarna på Software Skills som kan ersätta deras tidigare system. Följande kapitel syftar till att identifiera faktorer som är viktiga i en e-rekryteringsprocess, samt specificera vilka uppgifter ATS-prototypen skall utföra. Detta sker genom en analys som skapar en ram för att specificera problemet. Det specificerade problemet beskriver sedan en lösning som tillsammans med förstudien ligger till grund för hur webbapplikationen slutligen utformades.

2.1 Problemanalys

Från att rekryterare tidigare behövt behandla stora grupper av kandidater på papper i en sekventiell process karakteriseras istället e-rekrytering av en process där nya ansökningar behandlas kontinuerligt [1]. Fördelarna med e-rekryteringens kontinuerliga process är att den typiskt är snabbare, billigare och når ut till en bredare publik [3]. En nackdel är att företagen riskerar få större andel ointressanta eller otillräckliga ansökningar [3], men trots detta kan införandet av e-rekrytring minska den administrativa bördan hos företag [4].

I en studie under 2008-2010 där man kollade på hur införandet av e-rekrytering påverkade rekryteringsprocessen hos danska företag [2] fann man stöd för en tidigare teori [5] att den nya, kontinuerliga, e-rekryteringsprocessen kan delas in i tre huvudsakliga steg:

1. Attrahera kandidater:
   - Skapa en tilltalande och lämplig design av webbsidorna.
   - Spara kandidater.
2. Sortera fram kandidater
   - Genomför tester för att sålla bort okvalificerade eller olämpliga kandidater.
3. Kommunikation med kandidater

Just kommunikationssteget har visat sig vara särskilt viktig inom e-rekrytering. Kan-

didater som lämnar in en jobbansökan på webben förväntar sig snabb och kontinuerlig återkoppling [4]. Dessutom kan kontinuerlig kontakt med kandidater under rekryteringens gång bidra till att skapa en positiv bild av företaget [4]. Möjligheten att i en webbapplikation direkt kontakta och kommunicera med kandidater är därför eftertraktad.

2.1.1 Gemensamt arbete

I utvecklingen av en applikation där användare, i ett ATS fall rekryterare, kommer arbeta i lag finns det vissa aspekter som påverkar hur användbar applikationen i slutändan blir. Målet med en applikation som främjar gemensamt arbete är att assistera kommunikation, främja samarbete samt koordinera användarnas aktiviteter [6]. Det som skiljer en sådan applikation från alternativet (en applikation med bara en aktiv användare) är att sättet som människor kommunicerar på, gruppdynamik samt sociala roller har betydelse för hur applikationen bör utformas. De flesta applikationer som uppfyller dessa kriterier innehåller funktionalitet så som användarkommunikation, notifieringar, gruppgränssnitt och åtkomstkontroll [6]. Något annat som är centralt i sådana samverkande applikationer är att det ska ges möjlighet att få kännedom om vad som händer i applikationen [7], alltså vetskap om vem som utför vad och när det utfördes.

2.1.2 Problemets utgångspunkt

Vid projektstart hade Software Skills redan påbörjat arbetet med deras nya webbplattform. Det system som ska utvecklas är därför inte byggt från grunden, utan är en vidareutveckling av den kodbas som redan fanns. Detta betyder att vårt inflytande över vilka teknologier webbapplikationen bygger på var begränsat, men även att viss grundläggande funktionalitet var implementerad sedan tidigare. Dessa funktioner var:

- Inloggningssystem.
- Användarnivåer för företag, kandidat och administratör.
- Ett företag kan ha flera användare.
- Skapa och redigera jobbannonser.
- Ansöka till ett jobb.
- Automatiska mejlutskick.
- Uppladdning av bilagor, typiskt CVn.

2.2 Problemspecifikation

Med utgångspunkt i analysen från föregående avsnitt specificeras nu vilka funktioner den slutgiltiga ATS-prototypen idealt bör klara av för att den skall utgöra ett användbart arbetsverktyg i Software Skills rekryteringsprocess:

- Kandidater är alltid placerade i exakt en fas av rekryteringen, exempelvis *New* direkt efter att kandidaten lämnat in ansökan eller *Interviewed* om kandidaten har intervjuats.
- Kandidater kan ha olika färdigheter.
- Kandidater är antingen headhuntade av en rekryterare eller så har kandidaten själv lämnat in en ansökan.
- Rekryterare ska kunna söka och filtrera bland kandidater.
- Rekryterare inom samma företag kan arbeta tillsammans med rekryteringsprojekt.
- Rekryterare ska för varje jobbannons kunna anpassa vilka faser som rekryteringen består av.
- Rekryterare ska ha möjlighet att skriftligt kommunicera med kandidater.
- Rekryterare ska kunna kommunicera med andra rekryterare inom företaget.
- Rekryterare kan internt i rekryteringslaget ge omdömen till kandidater.
- Alla rekryterare i ett företag är behöriga att flytta kandidater mellan faser i rekryteringen.

Lösningen är en samverkande applikation som behandlar alla faser av rekryteringen: från ansökan, eventuella tester, intervju och slutligen anställning eller avslag. Systemet som beskrivits har även sociala funktioner som främjar kommunikation mellan rekryterare och kandidater samt mellan rekryterare inom samma företag.

3. Teknisk Bakgrund

Följande kapitel beskriver den teknik och de bakomliggande designmönster som utgör arkitekturen för ATS-systemet inSource.

3.1 HTML och CSS

HTML och CSS är två standardiserade språk som används i alla hemsidor, där HTML utgör hemsidans struktur medan CSS manipulerar strukturen för att skapa sidans visuella design och layout [8]. Vi använder oss av de i skrivande stund senaste standarderna: HTML5 och CSS3.

3.2 MVC

Model-View-Controller, förkortat MVC, är ett designmönster som bygger på tanken att datamodellen (M), vyn som användaren ser (V) och logiken som arbetar med datamodellen (C) bör separeras från varandra [9]. Exempelvis skall inte data manipuleras direkt i användargränssnittet i en klientapplikation som applicerar MVC-mönstret. Istället är det controllern som ska leverera, och eventuellt modifiera, data som sedan presenteras direkt i vyn (View).

3.3 Single-Page Application

En single-page application (SPA) är ett webbaserat program eller applikation som körs direkt via webbläsaren. Detta medför att applikationen är helt plattformsoberoende, vilket innebär att den kan köras från så väl datorer med alla typer av operativsystem till mobiltelefoner och surfplattor. Till skillnad från icke webbaserade applikationer krävs ingen installation eller uppdateringar på klientdatorer då själva programvaran är placerad på en eller flera servrar [10].

Det unika med en SPA jämfört med andra webbaserade applikationer är, som namnet antyder, att de består av endast en sida. Denna sida innehåller all Javascript, CSS och HTML-kod som krävs för att navigera och interagera med applikationen. Praktiskt sett innebär detta att de flesta beräkningar och uppdateringar av vyer utförs direkt från klientdatorn istället för att anropa servern. En SPA utnyttjar alltså mer av klientdatorns kraft än serverns vilket resulterar i mindre belastning på såväl nätverket som servern [10].

3.4 MEAN

MEAN är kombinerad lösning för webbapplikationer som använder sig av Javascript för både front-end och back-end [11]. Namnet är en akronym av MongoDB, ExpressJS, AngularJS och Node.js, som alla fyra baseras på Javascript. Att använda samma programmeringsspråk i många delar av arkitekturen underlättar utvecklingsprocessen. En person som inte har varit inblandad i en viss del kan fortfarande förstå syntaxen och göra modifieringar vid behov. Detta projekt använder alla delar förutom Express. Nedan följer en beskrivning av vad de olika teknikerna som använts fyller för funktion.

3.4.1 MongoDB

Till skillnad från de mest populära databaserna, Oracle och MySQL, som är så kallade relationsdatabaser är MongoDB en dokumentorienterad databas [12]. Skillnaden mellan en relationsorienterad och en dokumentorienterad databas är att den sistnämnda är friare när det gäller lagring av data. En dokumentorienterad databas som MongoDB kontrollerar inte om referenserna är korrekt angivna eller av rätt typ. MongoDB använder sig av JSON (Javascript Object Notation) liknande objekt för lagring av data.

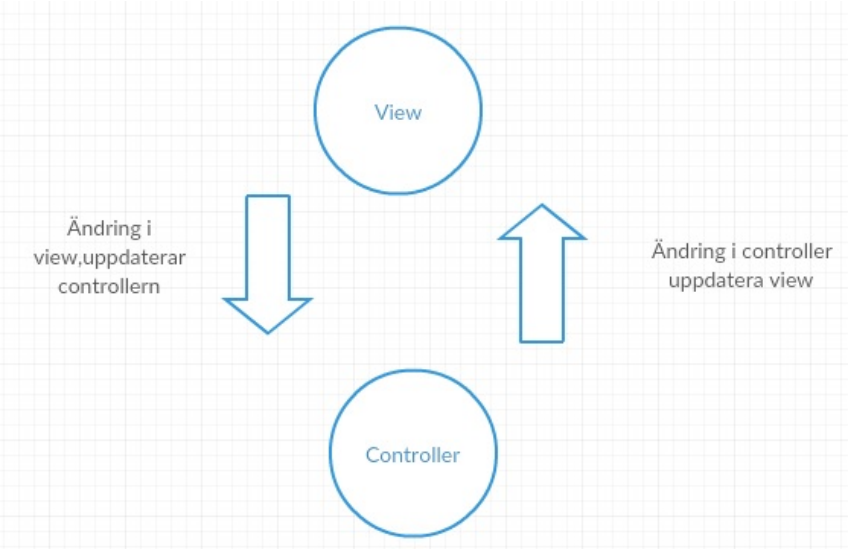
3.4.2 AngularJS

AngularJS, eller bara Angular, är ett Javascript-ramverk som är till stor hjälp för utveckling av SPA-applikationer. Angular använder sig av så kallade *directives*, direktiv, som är bundna till HTML-element och berättar för kompilatorn hur just det elementet ska bete sig. Detta är ett sätt att utöka HTMLs syntax med funktioner som normalt sett, utan Angular, hade krävt egenskapade Javascript-funktioner. Ett antal standarddirektiv finns redan fördefinierade, men det är även möjligt att skapa egna direktiv med hjälp av Javascript-liknande kod [13].

Angular implementerar även MVC-mönstret (beskrivet i avsnitt 3.2). Datamodellen som Angular använder består, förutom av datan som för tillfället används, av regler som beskriver hur data lagras. Vyn är i praktiken användargränssnittet och består därför av HTML och CSS. Varje vy har en egen HTML- och CSS-fil för att Angular dynamiskt ska

kunna byta ut vyer i en SPA. Angular implementerar controller-delen som en Javascript-fil vars uppgift är att synliggöra variabler och funktioner för vyn. Varje vy har därför en controller kopplad till sig.

HTML används vid webbutveckling och är anpassat för att skapa statiska webbsidor, men inte interaktiva applikationer. Om HTML kompletteras med Angular (eller Javascript) kan dynamiska webbsidor, eller en så kallad SPA, skapas. Angular använder sig av två-vägs data-bindning, vilket innebär att även data i HTML-vyn kan bindas till modellen. Detta betyder att samtidigt som data i HTML-vyn ändras, så ändras också data i controllern (se Figur 3.1).



Figur 3.1: Med två-vägs-databindning kommer en användare som ändrar data från vyn även uppdatera controllern eftersom data i vyn är bunden till variabler i controllern. På samma sätt orsakar en ändring av data i controllern en uppdatering av vyn.

3.4.3 Node.js

Node.js, eller bara Node, är ett ramverk byggt för skalbara serverapplikationer. Node körs endast på en tråd och kan därför hantera tiotusentals anslutningar samtidigt utan att behöva använda sig av dyra processorberäkningar [14]. Node väntar aldrig på att ett anrop ska returnera data utan fortsätter direkt till nästa funktion. Svar från tidigare anrop kommer senare med hjälp av callback-funktioner som utförs under exekvering. Eftersom Node endast körs på en tråd och använder sig av asynkrona anrop är deadlock något som inte kan inträffa på en Node-server [14].

3.5 Övriga ramverk och moduler

I följande avsnitt beskrivs de protokoll och moduler som ger ytterligare funktionalitet till tidigare nämnd teknik.

3.5.1 Sockets

Protokollet HTTP erbjuder primärt halv-duplex kommunikation över internet. För att möjliggöra full-duplex kommunikation utvecklades sockets, som efter en handskakning med mottagaren kan skicka data till dess motpart utan att först skicka en förfrågan. Detta kan ske parallellt, det vill säga båda parterna kan skicka data till varandra samtidigt, och på så vis uppnås realtidskommunikation [15].

3.5.2 Primus

Primus är en modul till Node, och tillhandahåller ett enhetligt API för realtids server-klientkommunikation som är helt oberoende av vilket realtidsramverk som används i grunden [16]. De olika realtidsramverken för Node ger principiellt samma funktionalitet, nämligen realtidskommunikation mellan server och klient. Däremot har dessa ramverk olika tankar om hur realtidskommunikation ska gå till och har därför skilda implementationer.

Primus fungerar som ett abstraktionslager för dessa realtidsramverk, vilket betyder att Primus API kan användas utan att tänka på vilket ramverk som egentligen används. I praktiken möjliggör Primus ett byte från exempelvis *Websockets* till *Socket.IO* genom att ändra på ett fåtal kodrader i servern, trots att Websockets och Socket.IO i grunden har olika API.

Primus använder sig av så kallade *sparks* som utgör länken mellan servern och en klient. Vidare används Primus rooms, ett tillägg som utökar Primus med möjligheten att skapa sparks som kan delas mellan flera klienter [17]. Dessa rum ger även servern möjlighet att skriva till samtliga klienter som befinner sig i samma rum.

3.5.3 Async

Async är en modul till Node som möjliggör parallella funktionsanrop, vilket exempelvis är användbart när flera oberoende databasanrop behöver utföras [18]. Async ger även möjlighet att kontrollera i vilket flöde som funktioner exekveras.

3.5.4 Redactor

Redactor är en proprietär Javascript-baserad textredigerare för webben, utvecklad av Imperavi [19]. Redigeraren formaterar texten som HTML, vilket gör att det du ser och skriver i redigeraren visas på samma sätt som i en webbläsare. Redactor kräver en licens som Software Skills sedan tidigare införskaffat och tillhandahöll projektet.

3.5.5 UI-Router

UI-Router är en modul som till Angular som används för att dynamiskt byta ut delar av en HTML-fil, vilket möjliggör skapandet av en SPA [20]. Navigationen i en applikation byggd med UI-Router kan ses som övergångar mellan olika tillstånd i applikationen där enbart de element som ändras behöver laddas om.

4. Förstudie

Innan den faktiska implementeringen kunde påbörjas studerades Software Skills nuvarande system, primärt hur de använde sig av tredjepartsprogrammen Trello och Google Drive och vilken data som sparades där. Förstudien innefattande även en undersökning av hur en applikation som främjar samverkan mellan användare i allmänhet utformas, samt hur detta senare skulle appliceras i det ATS som skulle komma att utvecklas. Utöver detta undersöktes redan befintliga webbapplikationer som åstadkommer just detta - de framtida konkurrenterna.

4.1 Software Skills nuvarande system

För att bättre förstå för vilken funktionalitet som skulle uppnås och vilka särskilda behov Software Skills hade undersökte gruppen hur deras nuvarande webbapplikation användes. Detta gjordes med hjälp av intervjuer av personalen på Software Skills där gruppen kom fram till följande slutsats:

Software Skills använder idag sin webbapplikation för att lägga ut jobbannonser som kandidater sedan söker till. Kodtester för den specifika annonsen skickas ut till sökande kandidater för att ge rekryterare en uppfattning om deras kompetens, detta utförs direkt på sidan. När en jobbannons skapas på deras webbapplikation skapas samtidigt ett Trello-board, som sedan fylls på med Trello-Cards allt eftersom kandidater söker. Med hjälp av Trello får företaget en överskådlig vy över de sökande för ett specifikt jobb, och kan flytta kandidater mellan olika steg i anställningsprocessen, exempelvis *ny*, *intervjuad*, *anställd*, etcetera. All data och statistik sparas i kalkylark på Google Drive, så som hurvida en person slutligen blev anställd på det aktuella företaget, om personen sökte självmant eller blev headhuntad av rekryterare på Software Skills.

4.2 Software Skills behov

Efter samtal med Software Skills vd och en granskning av det tidigare systemet stod det klart att företagets största behov var en ATS-lösning som ersatte de externa tjänsterna

Trello och Google Drive. Det var av hög vikt att strukturen efterliknade Trellos då denna gav en samlad bild av rekryteringsfasen, vilket var en huvudsaklig anledning till att Trello ursprungligen användes. Det var också viktigt att man på ett enkelt och smidigt sätt kunde flytta kandidater mellan olika rekryteringssteg, något i stil med Trellos dra-släpp funktion. Då företagets tidigare system utnyttjade Google Drive som ett slags grafiskt gränssnitt av deras databas fanns ett behov av en ny lösning som kunde visualisera databasen på deras egna plattform. Även data som tidigare enbart sparats ner på Google Drive skall istället sparas ner i Software Skills egna databas.

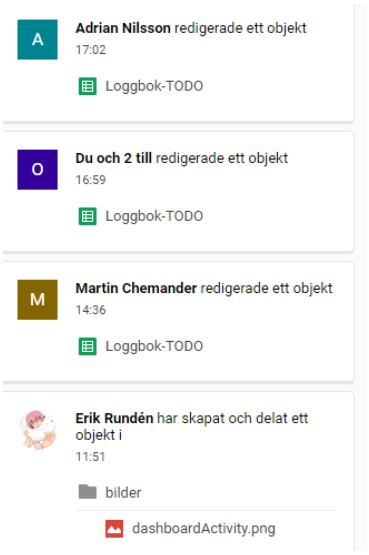
4.3 Webbapplikationer som främjar samverkan

Instuderingen av vilken funktionalitet som är central i webbapplikationer som gagnar samverkan mellan dess användare viktades mestadels till den forskning som bedrivits inom detta område. Även webbapplikationer som åstadkommer just detta i allmänhet, och inte specifikt för ett ATS studerades. De flesta studier visade emellertid ungefär samma slutsats. Exempelvis menade Michael Koch och Tom Gross på att följande stöd bör ges i en webbapplikation med flera användare [7]:

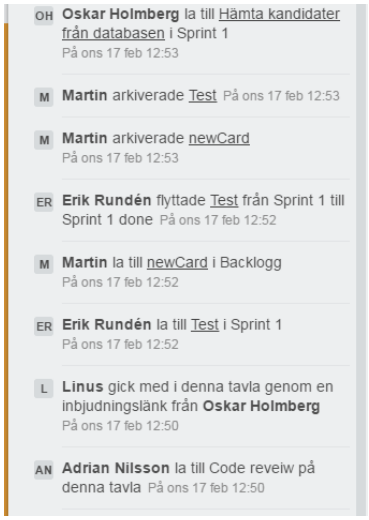
- **Stöd för kännedom** En central del i samverkande applikationer är att man skall ges möjlighet att få kännedom om vad som sker, v ad som hände, när det hände och hur. Användare skall inte isoleras från varandra utan snarare sluta sig samman.
- **Stöd för kommunikation** Samtidigt som stöd för kännedom ger ett indirekt sätt att kommunicera finns det också ett behov av direkt kommunikation. Exempel på detta är e-post, chattsystem och videolösningar.
- **Stöd för att koordinera aktiviteter** Även här hjälper stöd för kännedom att koordinera aktiviteter, men det finns även mer explicita tillvägagångssätt. Exempel på detta är arbetsflödeshantering och och specifika kommunikationslösningar.
- **Stöd för arbetslag** Exempel på lösningar är specifika arbetslagsrum.
- **Stöd för community** Möjlighet att sprida kunskap och nyheter inom företaget.

Beslutet att inrikta oss mot små och medelstora företag och tidsrymden av detta projektet gjorde att vi valde främst fokusera på de förstnämnda tre stöden: stöd för kännedom, kommunikation samt att koordinera aktiviteter.

För att hitta mer explicita exempel på lösningar för att ge stöd till dessa studerades webbapplikationer där användare jobbar tillsammans. Exempel på applikationer där sådan funktionalitet används är dokumentarkiverings- och arbetsflödeshanteringsprogram. Gruppen själva använde för detta Google Drive och Trello. Likheter som uppmärksammades var användandet av en aktivitetslogg för att visa händelser som tidigare skett (se Figur 4.1).



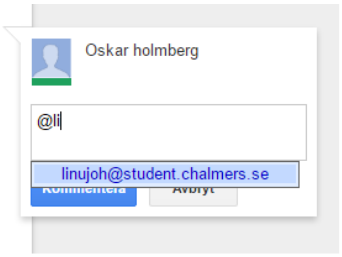
(a) Google Drives aktivitetslogg.



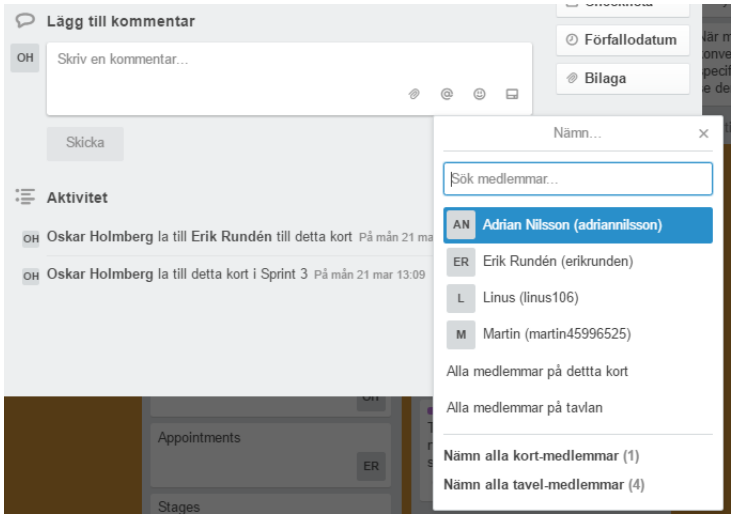
(b) Trellos aktivitetslogg.

Figur 4.1: Jämförelse mellan aktivitetsloggar i andra webbapplikationer. Notera att Google Drive kan klumpa ihop flera ändringar i samma fil till ett inlägg, medan Trello alltid redovisar individuella bidrag.

Utöver aktivitetsloggar ges det möjlighet att kommentera ett specifikt Trello Card, eller en markerad text i Google Drives fall. Detta kan likställas med att kommentera ett jobb eller en jobbansökan i ett ATS.



(a) Google Drives kommentarfunktion.



(b) Trellos kommentarfunktion.

Figur 4.2: Jämförelse mellan kommentarer i andra webbapplikationer.

I kommentarerna för Google Drive (se Figur 4.2a) och Trello (se Figur 4.2b) ges det möjlighet att omnämna andra användare som är involverade i projektet. I Google Drives fall kom det automatiskt upp en meny med alternativ då @ skrevs, och i Trellos fall får

användaren manuellt klicka på symbolen @ i Figur 4.2b för att nämna andra användare. Denna funktionalitet likställdes med att nämna rekryterare i ett ATS.

4.4 Undersökning av befintliga plattformar

För instudering av vilken funktionalitet som finns bland framtida konkurrenter studerades ett antal befintliga e-rekryteringsapplikationer. Vi kollade på sidor som Uptrail och Team Tailor och fann flertalet gemensamma funktioner. Den vanligaste återkommande funktionaliteten hos dessa sidor listas nedan.

- **Sökfunktion** Funktion som gör att rekryterare på ett enkelt och smidigt sätt kan hitta en specifik kandidat eller ett specifikt jobb.
- **Filtreringsfunktion** Funktion som gör det möjligt att sortera ut exempelvis kandidater eller jobb beroende på förutbestämda kriterier.
- **Meddelandefunktion** Funktion som möjliggör realtidskonversationer mellan rekryterare och kandidater direkt i applikationen.
- **Information om kandidat** En vy som presenterar en mer detaljerad information om en specifik kandidat, där exempelvis kandidatens namn, färdigheter och status i rekryteringsprocessen presenteras.

Vi fann dessa funktioner relevanta för rekryteringsprocessen och beslutade därför att vårt ATS skulle implementeras med liknande funktionalitet.

5. Arbetsmetod

Webbapplikationen utvecklades tillsammans med Software Skills och större delen av designen skapades av två masterstudenter från Interaktionsdesign. Detta innebar ett mycket tätt samarbete med såväl företaget som designutvecklarna. Alla frågor rörande applikationen som dök upp under projektets gång togs upp och diskuterades på planerade veckomöten, där både designutvecklarna och Software Skills vd närvarande. På dessa möten närvarade även företagets tekniska utvecklare, en person som stöttat och hjälpt till med de tekniska frågorna som uppkom.

5.1 Framtagande av kravspecifikation

Trots att ATS-system ofta är av komplex natur fanns det väldigt lite studier på hur man hanterar kandidater genom hela rekryteringsprocessen. Med hjälp av intervjuer med personer från Software Skills upprättades en kravspecifikation som var grunden till den backlogg som användes under projektets gång. Frågor som besvarades var bland annat hur de använde nuvarande externa tjänster, men också vilken funktionalitet som skulle eftersträvas i den nya produkten för att kunna hantera en anställningsprocess så smidigt som möjligt. Kravspecifikationen kompletterades allt eftersom masterstudenternas design blev allt mer komplett, vilket passade en agil arbetsmetod. Utöver problembeskrivningen från Software Skills utökades kravspecifikationen med de aspekter och funktionalitet som tillkom från förstudien (se avsnitt 4).

5.2 Scrum

Under projektets gång applicerade gruppen Scrum-ramverket [21] som är en vanlig arbetsmetod i mjukvarurelaterade projekt. I Scrum delas ett projekt upp i form av backlogg, aktiviteter och sprintar. Vanligtvis hålls inom Scrum korta dagliga möten, men eftersom gruppen inte hade möjlighet till detta användes istället en modifierad version med veckovisa möten.

Backloggen listade alla nödvändiga delar och funktioner, med tillhörande prioritet och

uppskattad tidsomfattning, som krävdes för att uppfylla de krav som ställdes på slutprodukten. Backloggen var dynamisk, vilket betyder att nya funktioner kunde tillkomma eller tas bort under projektets gång.

Aktiviteterna bestod av ett möte på Software Skills en gång i veckan, samt två till tre workshops per vecka där gruppen tillsammans kom fram till en tid som passade.

Sprintar startades i samråd med personalen på Software Skills där specifika element från den tidigare skapade backloggen valdes ut. Detta bestämde vad gruppen skulle utföra de nästkommande två veckorna. Efter en vecka under pågående sprint evaluerades nuvarande status, vid behov lades nya mål till eller i värsta fall togs vissa bort. I slutet på en sprint utvärderades resultatet, vilket även lade grunden till nästkommande sprint som började direkt efter.

5.3 Versionshantering

Eftersom gruppens arbete var uppdelat i sprintar som ledde till att det blev många ändringar av koden under korta perioder var det viktigt att ha någon typ av versionshantering för att kunna se vad som ändrades, och även lätt kunna återgå till gamla versioner av koden vid behov. Det system som användes var Git [22] och detta tillåter projekt att delas in i olika förgreningar vilket betydde att gruppen kunde jobba vid sidan av Software Skills ursprungliga kod. När företaget uppdaterade deras webbapplikation kunde vi sammanfoga ändringarna till förgreningen och därmed fortsätta arbetet på en mer uppdaterad bas.

6. Implementation

Detta avsnitt förklarar hur teknologierna beskrivna i den tekniska bakgrunden (kapitel 3) använts för att realisera det specificerade problemet. Se resultatkapitlet (7) för en beskrivning av det resulterande systemet inSource från ett användarperspektiv.

6.1 Modellering

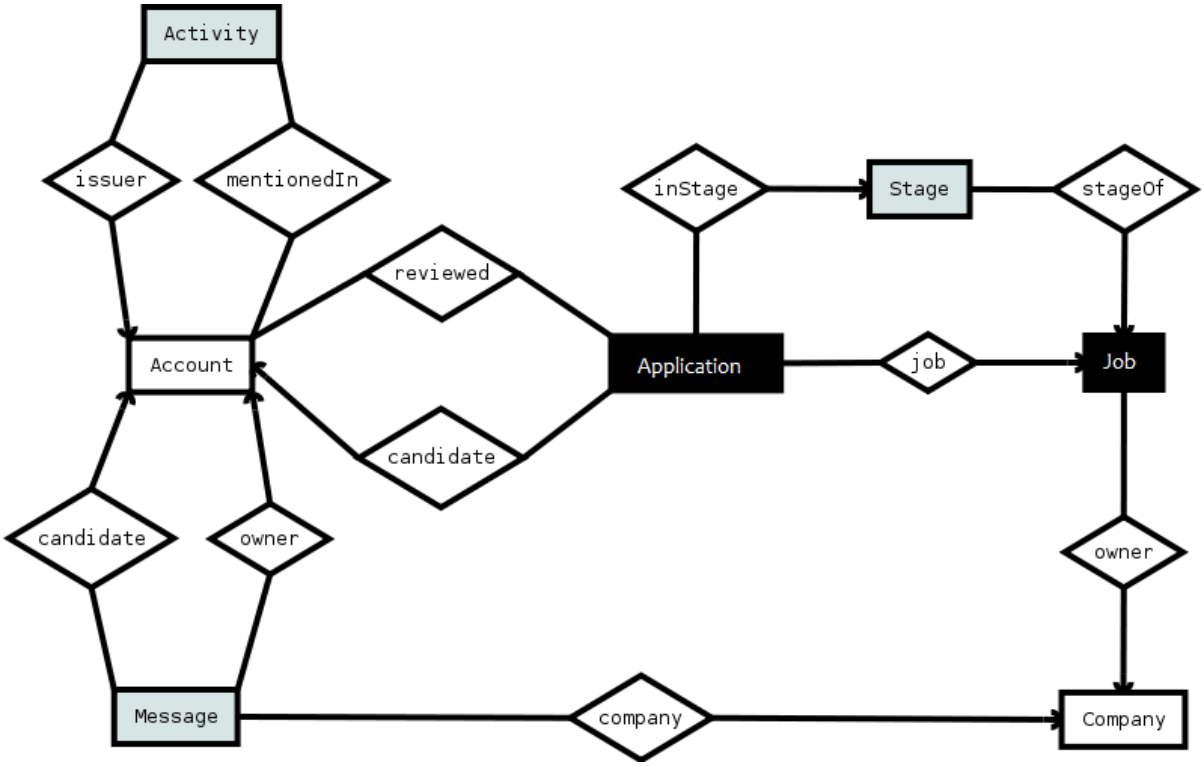
Efter att en kravspecifikation samt backlogg skapats framställdes även ett Entity-Relationship-diagram, vilket är ett diagram över hur databasen slutligen ska se ut. Detta gjordes för att få en bättre inblick över databasen före implementering. En MVC-modell togs även fram för att få en överblick bild över ATS-systemet, denna återfinns i appendix A.1.

6.1.1 Databasen

MongoDB är som nämnts i avsnitt 3.4.1 en dokument-orienterad databas. Data som sparas ner i JSON-format kan i princip innehålla vad som helst. MongoDB säkerställer inte om referenserna är korrekt angivna, om de överhuvudtaget existerar, eller om de är av korrekt typ. Detta innebar ett stort ansvar på oss som programmerare när vi modellerade databasen och krävde stor vikt vid felsökning och verifikation av dess korrekthet.

De delar av databasen som redan var implementerade vid projektets början var konton, jobb, jobbansökningar samt företag. Dessa har dock kontinuerligt vidareutvecklats under projektets gång.

Information som tidigare sparats på Google Drive och Trello skulle nu sparas i Software Skills egna databas. Den ökade komplexiteten i det nya rekryteringssystemet bidrog till att även de redan befintliga delarna i systemet behövde vidareutvecklas. Även Software Skills kodtester var en del av den redan befintliga databasen, dessa har dock inte vidrörts inom ramen av detta projekt. Det som däremot inte tidigare fanns överhuvudtaget i databasen var meddelandefunktionalitet, aktivitetslogg, omnämnanden, omdömen samt rekryteringssteg.

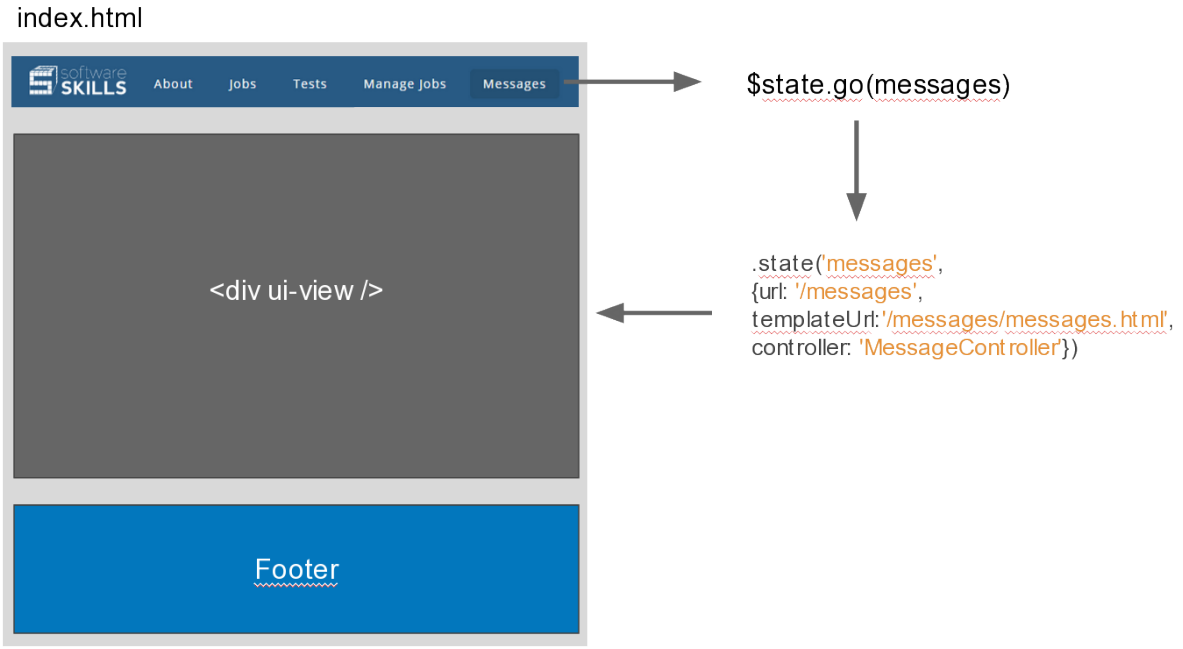


Figur 6.1: Simplifierat ER-diagram som beskriver strukturen av databasen. Gråa entiteter representerar det vi själva lagt till. Svarta entiteter är sådana som redan fanns från projektets början, men som vi behövt modifiera.

I Figur 6.1 visas ett ER-diagram över den databas som kom att implementeras. För enkelhetens skull har de olika attribut som varje entitet innehåller utelämnats och även de testrelaterade entiteter som tidigare fanns.

6.1.2 SPA-arkitektur med hjälp av UI-Router till Angular

Software Skills hemsida var sedan tidigare uppbyggd som en SPA genom användning av UI-Router. Eftersom inSource integreras i hemsidan har vi behövt följa denna uppbyggnad. Principiellt fungerar UI-Router så att vi på ett ställe i huvuddokumentet (index.html) har ett enskilt element med attributen *ui-view* dit UI-Router dynamiskt kan injicera HTML från ett av flera fördefinierade tillstånd. För varje ny vy som skapas definieras ett nytt tillstånd, likt den i Figur 6.2, som blir känd för UI-Router. Varje tillstånd definieras av dess namn, vilken URL den använder, vart HTML-filen som utgör vyn finns, vart controllern som förknippas med vyn finns samt en eventuell *resolve*-funktion som hämtar vy-specifik data från databasen.



Figur 6.2: Figuren beskriver skeendet då en användare klickar på *Messages* i menyn. Istället för att ladda en helt ny HTML-fil som uppdaterar hela sidan säger vi med funktionen *go(stateName)* att UI-Router ska byta tillstånd. Eftersom vi på ett annat ställe i koden har definierat tillståndet *messages* kan UI-Router injicera innehållet i den HTML-fil som associeras med tillståndet i det elementet i index-dokumentet med attributet ui-view. Effekten blir att enbart de element som ändras laddas om när vi byter tillstånd.

6.2 Rekryteringssteg

För att underlätta kategoriseringen av ansökande kandidater använder sig applikationen av så kallade rekryteringssteg, vilket är en beskrivning av vart i rekryteringsfasen som kandidaten befinner sig. Vid skapandet av ett nytt jobb lägger applikationen till tre statiska rekryteringssteg som ingår i alla jobb: *ny*, *avfärdad* och *erbjuden*. Utöver dessa steg finns det möjlighet att utöka med ytterligare rekryteringssteg för att anpassa jobbet med avseende på vilken rekryteringsprocess som skall tillämpas på det specifika jobbet. När en ny ansökan kommer in till ett jobb placeras denna ansökan under kategori ny i rekryteringsstegen. Det är även i denna kategori alla ansökningar som inte blivit behandlade av någon rekryterare återfinns. Allt eftersom rekryteringsprocessen fortskrider och ansökningarna behandlas kan de sökande flyttas mellan de olika rekryteringsstegen. En mer ingående beskrivning om hur skapandet av ett jobb går till i applikationen återfinns i appendix A.2

I Tabell 6.1 ges en överblick hur ett rekryteringssteg är implementerat i databasen. För att hålla reda på i vilken ordning i rekryteringsfasen som rekryteingssteg skall vara placerad i sparas en position för varje rekryteringssteg.

Tabell 6.1: Representation av ett rekryteringssteg i databasen.

Stage
_id: ObjectID()
type: String
name: String
job: ObjectID()
position: int
updatedAt: Date()
createdAt: Date()

6.3 Realtidskommunikation

Kommunikationsaspekten som kom att genomsyra hela utvecklingen av webbapplikationen gjorde det nästintill obligatoriskt att kunna erbjuda realtidskommunikation. Med hjälp av abstraktionslagret Primus implementerades detta, med Websockets som underliggande protokoll. Realtidskommunikation är främst användbart för den meddelandefunktion som kom att implementeras, men gav också möjlighet att exempelvis se i realtid när en jobbannons får nya sökande.

När en kandidatanvändare loggar in tilldelas denne genom Primus Rooms ett personligt rum dit andra användare kan skicka data i realtid. Rekryterare går även med i ett gemensamt rum för företaget. Syftet med rummen är primärt att skapa en yta mot vilken användare i andra rum kan skicka meddelanden.

6.4 Aktivitetslogg

I samråd med Software Skills utvecklades en applikation med en platt struktur, det vill säga att samtliga rekryterare har samma behörighet på sidan och kan således utföra alla ändringar i rekryteringsprocessen. Detta medförde dock att alla händelser måste loggas. Parallellt med att ändringar utförs i applikationen sparas därför också en logg ner som visar vem som utförde aktiviteten, vilken operation som utförts och en referens till det aktuella jobbet eller ansökan som relaterade till aktiviteten. Vissa ändringar, så som att flytta en kandidat mellan rekryteringssteg, krävde emellertid ytterligare attribut, Tabell 6.2 beskriver dock den övergripande strukturen.

Tabell 6.2: Intern representation av ett aktivitetsloggs-element i databasen

Activity
_id: ObjectID()
type: String
operation: String
application: ObjectID() I de fall en applikation är involverad
job: ObjectID() I de fall ett jobb är involverat
issuer: ObjectID() Utfärdaren av händelsen
date: Date()

För att inSource skulle hålla sig till idén om MVC som beskrevs i 3.2 innehöll inte datamodellen någon information eller logik för hur en aktivitet från loggen presenteras för användaren. Det som sparades ner var vilken typ av operation som utfördes, vem som utförde den, när det skett och det eventuella jobbet eller jobbansökan som var relaterat till händelsen. Detta ger möjlighet att presentera aktivitetsloggen på olika sätt. Exempelvis kan den presenteras på olika språk och visa aktiviteter på olika sätt beroende på om betraktaren är en rekryterare eller kandidat. I en vy där det redan är känt vilket jobb eller ansökan som händelsen relaterar till vore det exempelvis redundant att visa denna information igen på sidans aktivitetslogg.

6.5 Omdömen

Efter att lösningar av olika typer av betygssystem diskuterats implementerades omdömen där en rekryterare kan bedöma en kandidats färdigheter, ambitioner, samarbetsvilja samt dennes ledaregenskaper. Olika rekryterares omdömen summeras sedan ihop med hjälp av Javascript och bildar ett genomsnittsvärde som kan användas för att filtrera kandidater på.

Tabell 6.3: Representation av ett omdöme i databasen.

Review
_id: ObjectID()
type: String
application: ObjectID()
reviewer: ObjectID()
skills: int
motivation: int
teamPlayer: int
leader: int
note: String
date: Date()

I Figur 6.3 ses representationen av ett omdöme i databasen. I de fall då en rekryterare inte fått kännedom om någon av de fyra egenskaper som ska betygsättas, kan denne välja att utelämna fältet och det sparas då ner som en nolla i databasen, vilket indikerar att man ännu inte vet detta.

6.6 Mentions

För att uppfylla de kommunikationskriterier dagens rekryteringssystem kräver implementerades ett sätt att omnämna andra rekryterare i noteringar angående specifika jobbansökningar. I samband med att en notering sparas ner i databasen sparas också en referens till de användarkonton som nämndes i noteringen.

Tabell 6.4: Representation av en notering i databasen. Attributet *mention* är en lista med referenser till de konton som blev omnämnda i noteringen.

Activity
_id: ObjectID()
type: String
operation: 'note'
application: ObjectID()
note: String
mention: [ObjectID()]
date: Date()

6.7 Meddelandesystem

Samtidigt som inloggning sker hämtar servern alla meddelanden som antingen är skickade till användaren, eller skickade av användaren själv, och gör dessa synliga för controllern på klientsidan.

Sökfunktionen är baserad på Javascript-funktionen indexOf, som jämför en textsträng mot innehållet i en annan textsträng. För att sökningen inte ska vara skiftlägeskänslig konverteras först både sökordet och strängarna som sökordet matchas på till gemener.

Enligt MVC, beskrivet i avsnitt 3.2, ska datamodellen inte innehålla logik som är till för att formatera data som senare presenteras för användaren, detta är controllerns uppgift. En skriftlig konversation kan betraktas som ett sätt att gruppera meddelanden på, och tillför därmed ingen data som inte kan byggas in direkt i datamodellen. För att hålla oss till MVC-mönstret representeras därför inte konversationer internt i databasen (se Tabell 6.5). Istället sorteras meddelanden in i konversationer först i controllern på klientsidan.

Tabell 6.5: Intern representation av ett meddelande i databasen. Attributet *content* utgör själva meddelandet och består av en textsträng formaterad som HTML av Redactor. Utifrån attributen *company* och *candidate* kan meddelanden sorteras in i konversationer mellan företag och kandidater.

Message
_id: ObjectID()
type: String
candidate: ObjectID()
company: ObjectID()
owner: ObjectID()
content: String
read: boolean
createdAt: Date()

För att skriva meddelanden används textredigeraren Redactor. Redactor formaterar texten med hjälp av HTML, vilket innebär att innehållet i ett meddelandes *content*-attribut (se Tabell 6.5) kan visas direkt i webbläsaren som formaterad text genom Angulars direktiv *ng-bind-html*.

6.8 Async

På grund av Nodes icke-blockerande natur kan parallella anrop till databasen ske med hjälp av node-modulen Async. Direkt när en funktion på servern skickar iväg ett I/O-anrop blir den inaktiv medan den väntar på svar. Detta betyder att servern kan fortsätta exekvera nästkommande funktion. Exempelvis kan en förfrågan skickas till databasen om att hämta ett företags jobb och därefter, innan svaret kommit, skicka en förfrågan om att hämta företagets konton då dessa var oberoende av varandra. Ett fall då detta inte går att applicera är när kandidaterna till de jobb som ett företag har utlagda ska hämtas från databasen. Eftersom det finns flera företag i databasen måste då alla jobb utlagda av vårt företag hämtas, och först därefter kan ansökningarna relaterade till jobben hämtas.

7. Resultat

Målet var att utveckla och integrera en prototyp av ett ATS anpassat efter Software Skills behov som skulle göra dem oberoende av Trello och Google Drive, fast ändå ge liknande funktionalitet. I dagsläget är inte Trellos dra-släpp funktion implementerad. Däremot är den bakomliggande funktionaliteten implementerad som Trellos dra-släpp funktion använts till, fast med hjälp av kryssrutor och rullgardinsmenyer. Vi har skapat ett kommunikations-verktyg som ger en direkt förbindelse mellan rekryterare och kandidater i rekryteringsprocessen. Google Drive har ersatts med en detaljerad kandidatvy som är specifik för ett jobb. Data tillhörande kandidaten går att hitta på denna vy även efter att positionen har blivit tillsatt.

Följande kapitel beskriver ur ett användarperspektiv de funktioner i inSource som vi implementerat. Samtliga figurer i kapitlet visar alltså hur inSource körs direkt i webbläsaren, och inte statiska bilder på en tänkt design.

7.1 Meddelandesystem

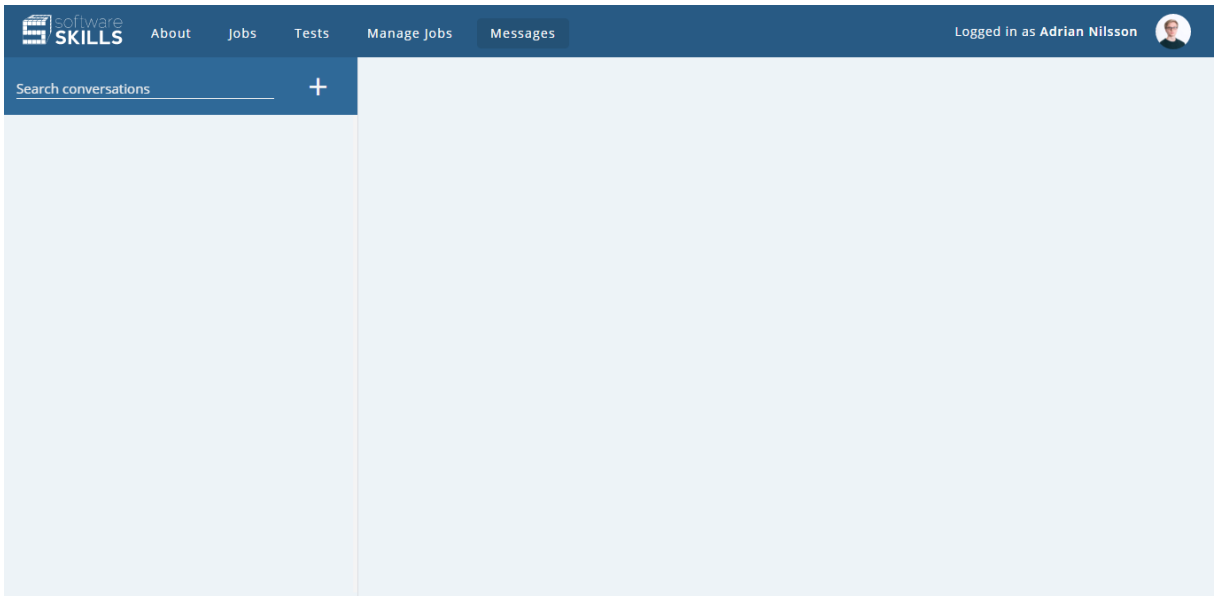
Det meddelandesystem som skapats möjliggör realtidskommunikation mellan enskilda kandidater och rekryterare. Konversationerna är sorterade i kronologisk ordning med den senast aktiva konversationen överst.

Sökrutan ger möjlighet att med fri text söka efter specifika konversationer. Sökningen är inte skiftlägeskänslig och sökordet matchar på motpartens namn samt på innehållet i alla meddelanden. En sökning på exempelvis *Emil* ger alla konversationer med personer som har Emil i sitt namn (alltså skulle både Emil Andersson och Anders Emilsson visas) samt de konversationer där texten Emil förekommer i ett meddelande.

7.1.1 Ett detaljerat exempel

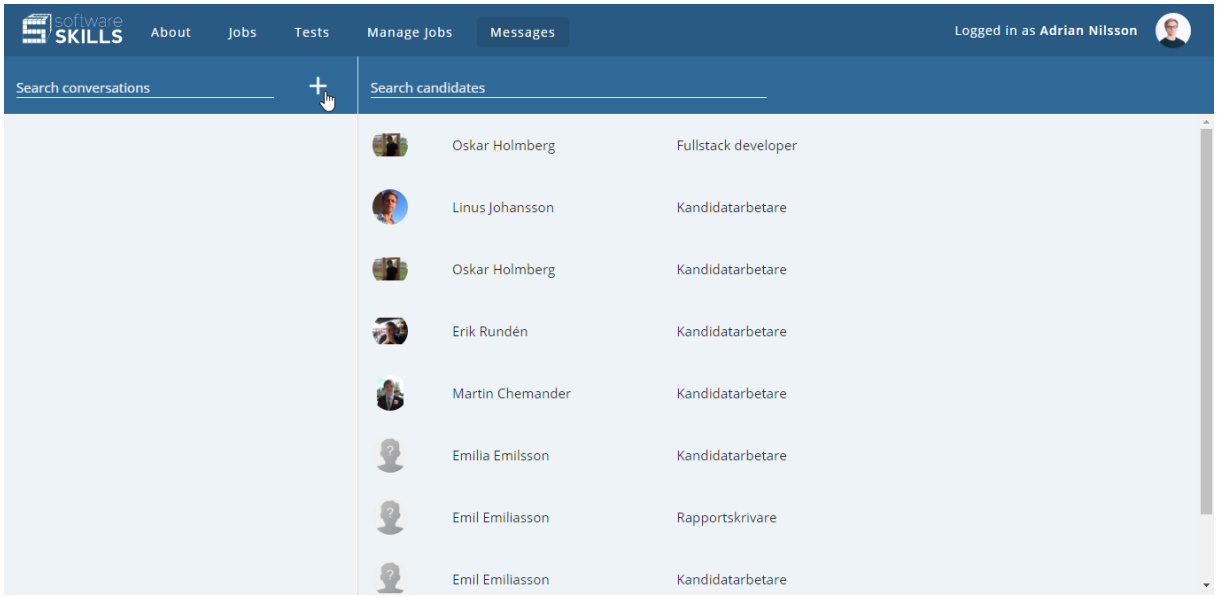
Vi betraktar i detta avsnitt ett exempel som demonstrerar meddelandesystemets funktionalitet. Ponera att en kandidat Emilia Emilsson har ansökt till jobbet *Kadidatarbetare*, utlagd av företaget *Kandidatarbete*. En rekryterare på företaget tror att Emilia kan vara

en lämplig kandidat för jobbet. Kommunikation med Emilia kan då ske direkt genom webbapplikationens meddelandesystem. Eftersom rekryteraren är ny på företaget har denne inga konversationer sedan tidigare, och ser därför vyn i Figur 7.1 när hen går till Messages från menyn.



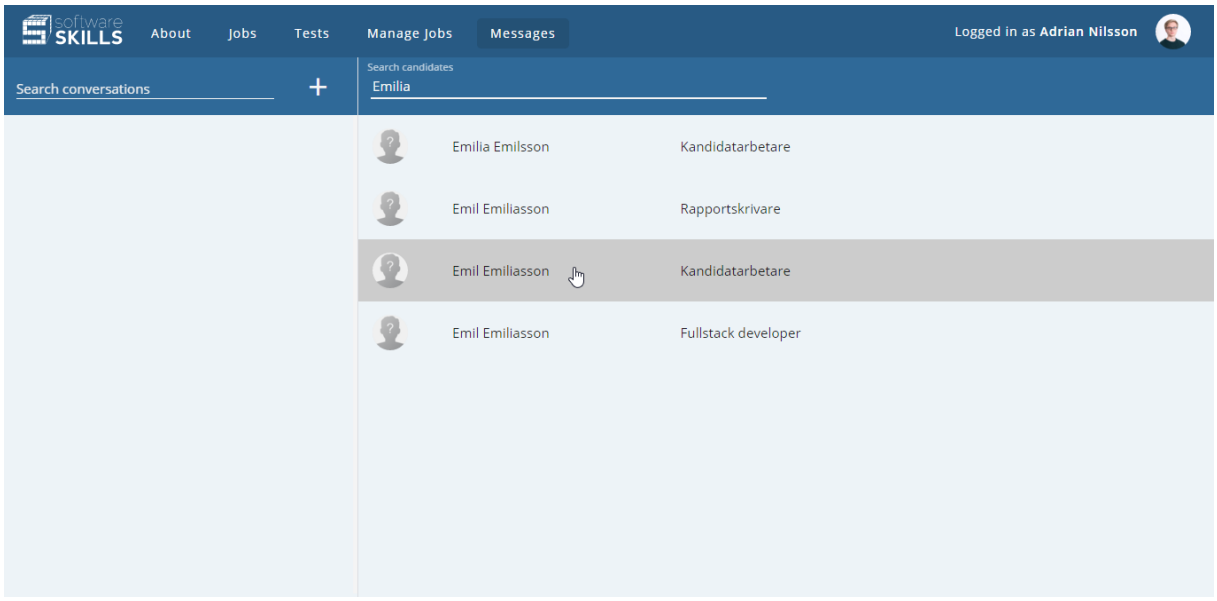
Figur 7.1: Figuren visar meddelandevyn så som den ser ut när det inte finns några konversationer, vilket typiskt motsvarar första gången en användare öppnar vyn.

Genom pluset till höger som sökrutan kommer rekryteraren vidare till en vy där denne kan söka bland och starta konversationer med kandidater (se Figur 7.2). En kandidatanvändare som klickar på pluset ser istället en lista med företag som personen i fråga kan starta en konversation med.



Figur 7.2: Efter att rekryteraren har klickat på pluset för att starta en ny konversation visas en lista med alla kandidater som svarat på någon av företagets jobbannonser. En kandidat som svarat på två annonser utlagda av samma företag, så som Emil Emiliasson i figuren, listas därför två gånger, en gång för varje ansökning.

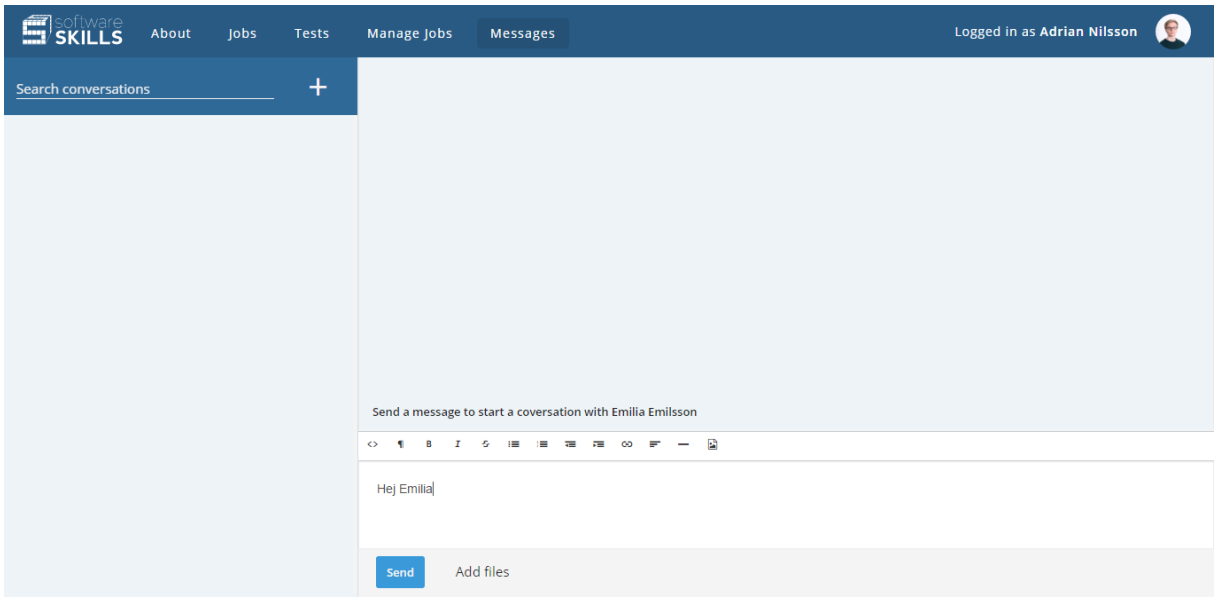
Från vyn i Figur 7.2 kan rekryteraren göra en fritextsökning för att hitta rätt kandidat. Kandidaterna filtreras då i realtid, alltså samtidigt som rekryteraren skriver.



Figur 7.3: En sökning på *Emilia* ger alla personer med Emilia någonstans i sitt namn, i figuren *Emilia Emilsson* och *Emil Emiliasson*. Notera gråmarkeringen på kandidaten som muspekaren hovrar över.

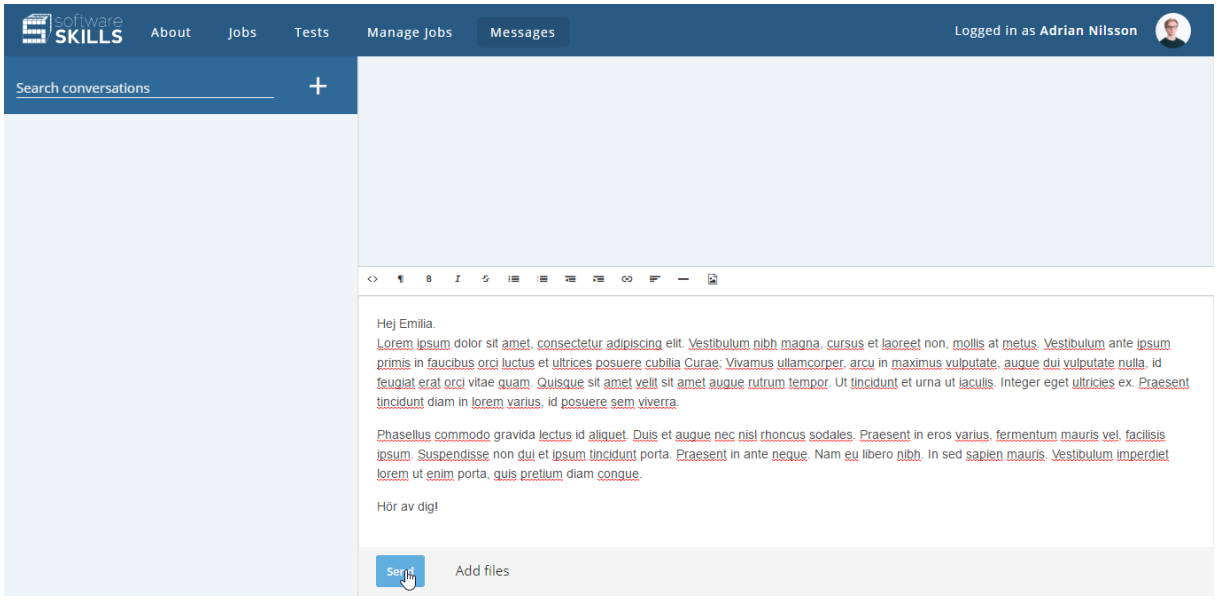
Efter att rekryteraren klickat på Emilia i vyn i Figur 7.3 presenteras textredigeraren Redactor där själva meddelandet skrivs. Rekryteraren uppmanas samtidigt skicka ett

första meddelande som initierar konversationen. Syftet med uppmaningen är det tydligt ska framgå vem meddelandet skickas till. Vyn illustreras i Figur 7.4.

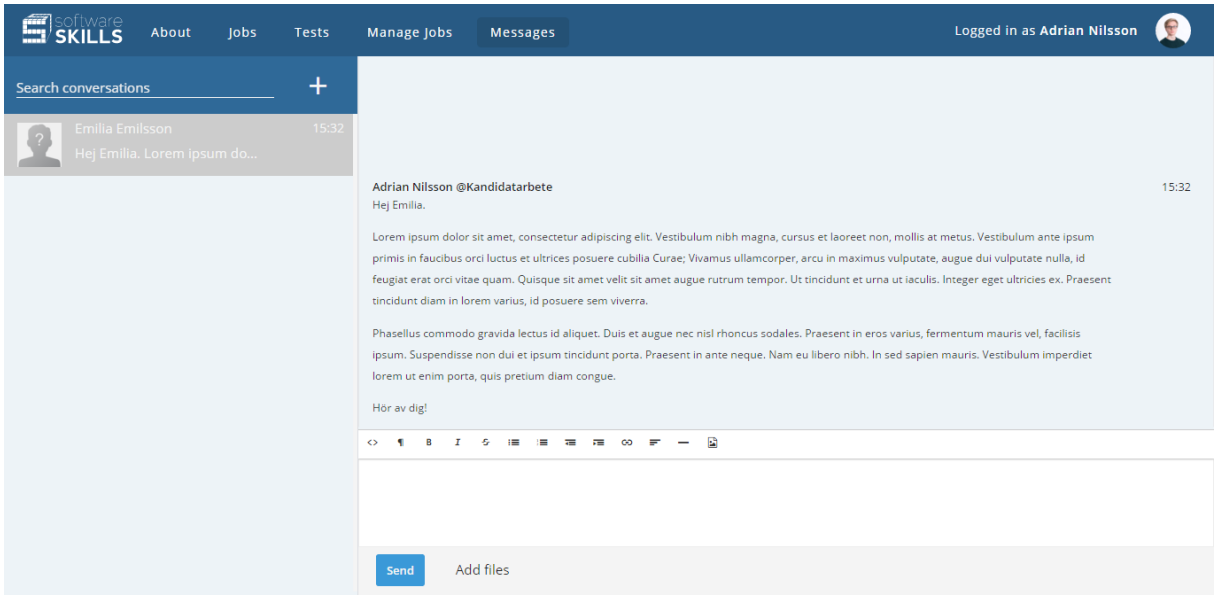


Figur 7.4: Figuren visar vyn för att starta en ny konversation, i detta fall med Emilia Emilsson.

Vid längre textstycken växer textredigeraren på höjden (se Figur 7.5) tills dess att den når menyn på toppen av sidan då en rullningslist istället introduceras inuti redigeraren. Figur 7.6 visar hur vyn ändras direkt efter att meddelandet skickats. Dessutom skapas en ny konversation i sidomenyn om meddelandet var det första i konversationen, jämför Figur 7.5 och Figur 7.6.

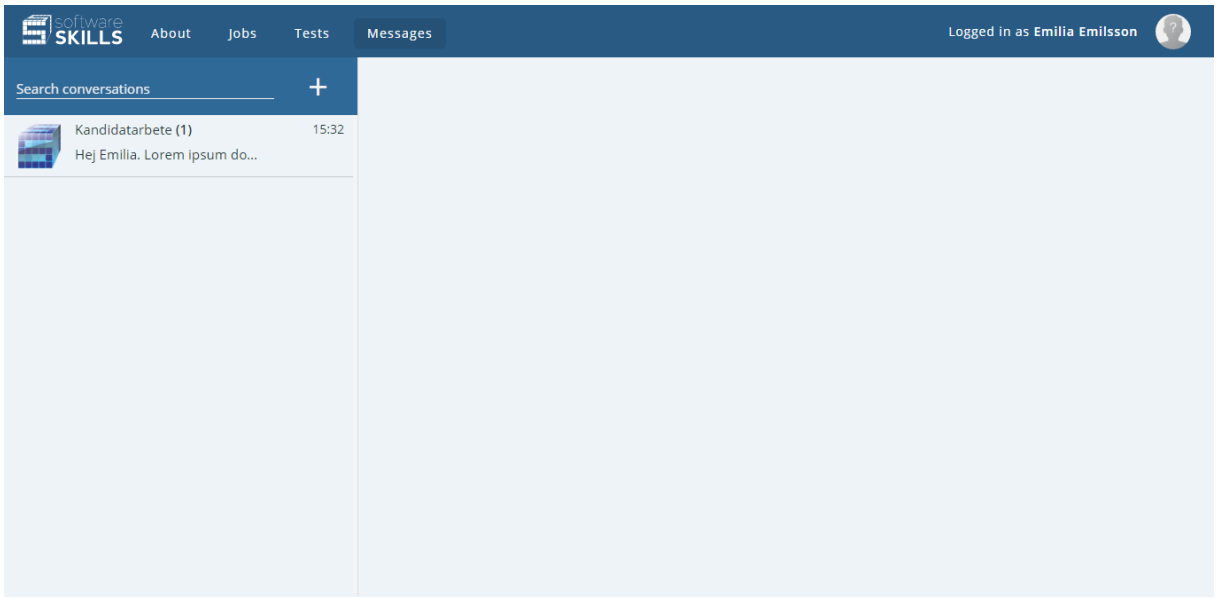


Figur 7.5: Figuren visar hur textredigeraren växer på höjden vid komposition av ett längre meddelande. Textredigeraren döljer då innehåll som överlappar p.g.a. den extra höjden. Efter meddelandet skickats återgår textredigeraren till sin normala höjd.



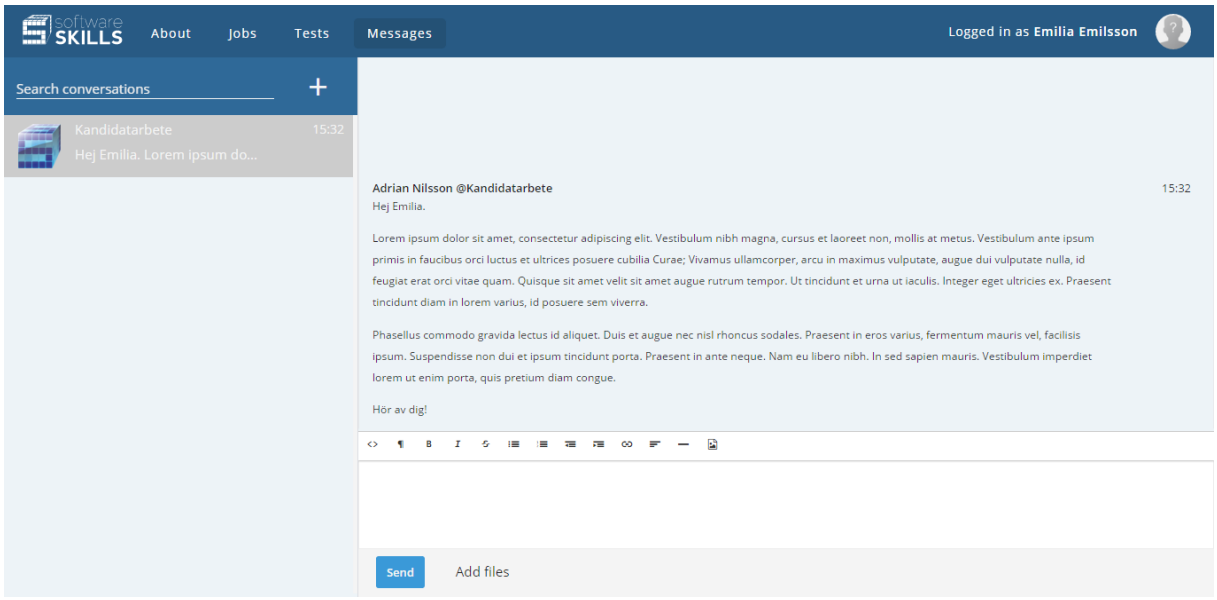
Figur 7.6: Rekryteraren har genom att skicka meddelandet från figur 7.5 startat en konversation med kandidaten Emilia Emilsson. Notera att konversationen är tillgänglig i sidomenyn till vänster.

Om Emilia inte är inloggad när rekryteraren skickar meddelandet kommer ett mejl som meddelar att hen har ett oläst meddelande automatiskt att skickas till den angivna e-postadressen i Emilias profil. När Emilia sedan går till meddelandevyn kommer hen istället för den tomma vyn i Figur 7.1 se att en konversation med företaget *Kandidatarbete* redan har skapats (se Figur 7.7).



Figur 7.7: Visar en kandidat som har en konversation med företaget *Kandidatarbete*. Konversationen har ett oläst meddelade, vilket indikeras av den fetstilta siffran bredvid företagsnamnet. Även tiden då det senaste meddelandet skickades visas i konversationskortets högra hörn

I Figur 7.8 har Emilia valt att visa konversationen med företaget *Kandidatarbete*. I vyn ges information om vem på företaget som skickade meddelandet, när det skickades samt själva meddelandet. Härifrån kan Emilia sedan skriva ett svar och föra konversationen vidare.

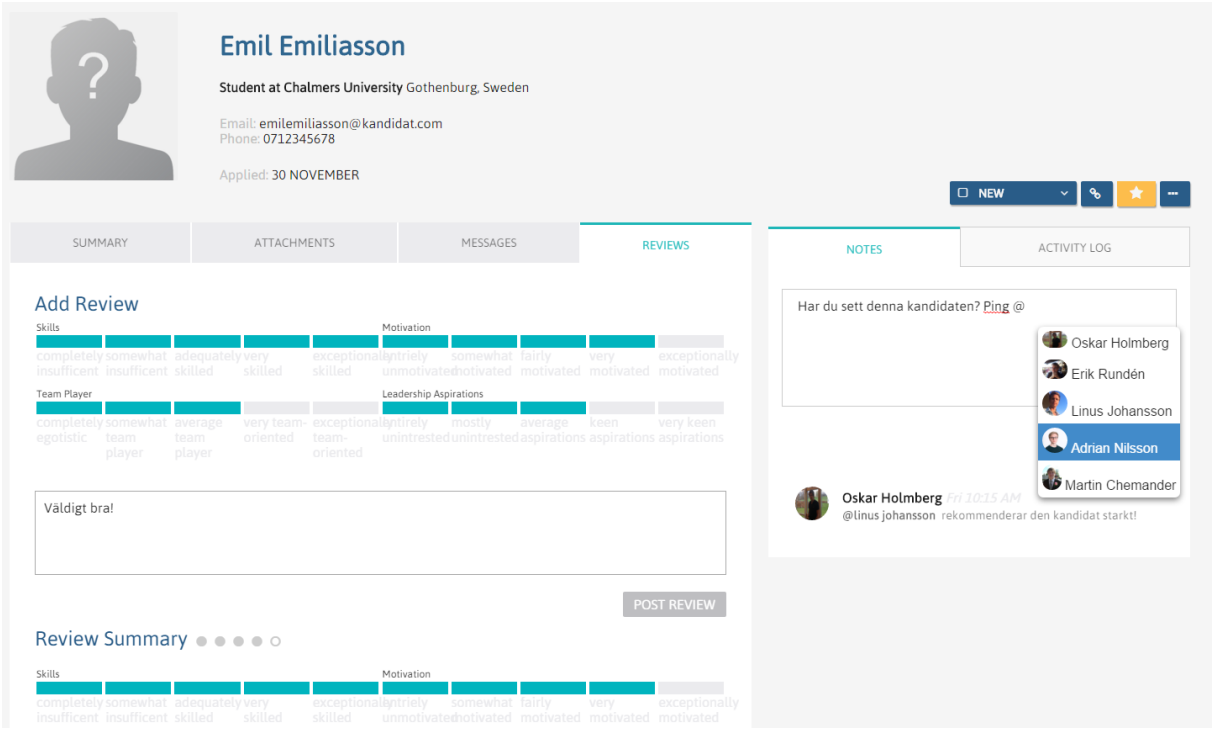


Figur 7.8: Vyn kandidaten från Figur 7.7 kommer till då denne öppnar konversationen med företaget Kandidatarbete. Vid öppnande av konversationen markerades alla olästa meddelanden som lästa. Notera att den aktuella konversationen är gråmarkerad.

Meddelandesystemet är den enda delen av applikationen som designutvecklarna inte har skapat en design för. Istället har vi varit helt ansvariga för hela meddelandesystemet. Alla figurer i detta avsnitt är tagna på en 15,6-tums bildskärm med upplösningen 1366x768 pixlar.

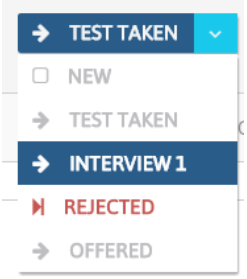
7.2 Detaljerad kandidatvy

Under denna vy presenteras detaljerad information angående en kandidat som sökt till ett specifikt jobb. Här ges möjligheten att bokmärka särskilt lovande kandidater och flytta den sökande kandidaten mellan olika rekryteringssteg. Vyn innehåller även sociala funktioner som att nämna någon från företaget i en notering angående kandidatens ansökan, visa meddelandehistorik, skicka nya meddelanden samt ge den sökande ett omdöme. Alla ändringar loggas och sparas ner i databasen, och visas i vyn under fliken *Activity Log*.



Figur 7.9: Vy som ger detaljerad information om en kandidats jobbansökan. Här ges rekryterare möjlighet att betygssätta en kandidat, skriva anteckningar, flytta en kandidat mellan rekryteringssteg samt skicka meddelanden.

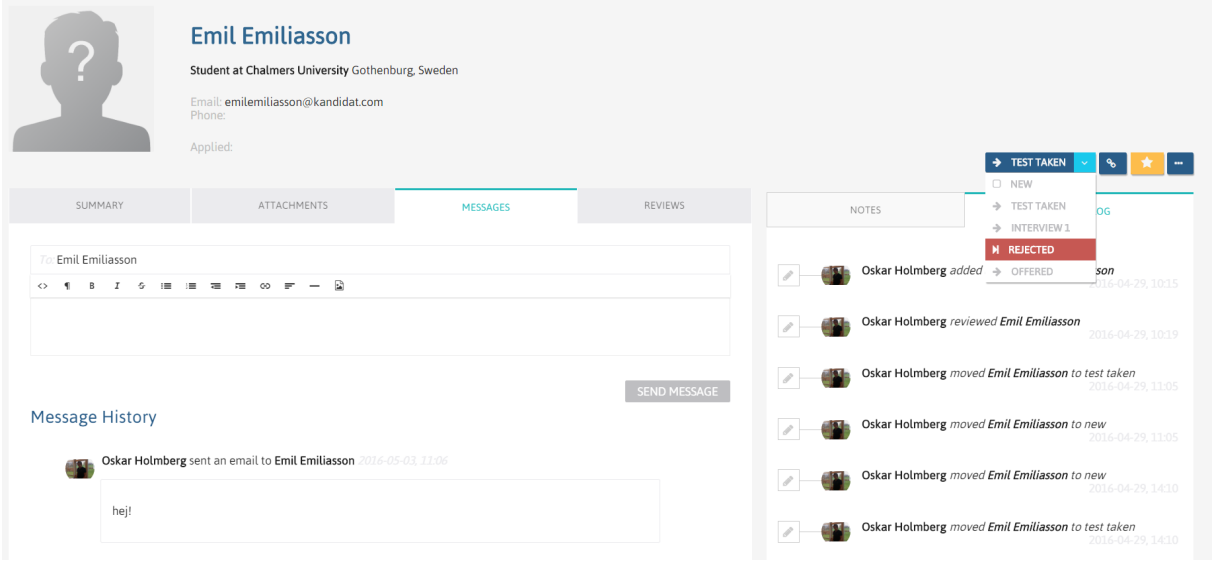
I Figur 7.9 ses den detaljerade kandidatvyn som helhet. Fliken *Summary* innebär kort en sammanfattning över den information en kandidat själv angett vid ansökningstillfället. Då det inte ingick i projektbeskrivningen att ändra ansökningsprocessen för en kandidat utelämnades utveckling av denna flik. För att kunna flytta en kandidat mellan olika rekryteringssteg används en flervalsmeny (se Figur 7.10). Under den aktiva fliken *Notes* visas samtliga rekryterares anteckningar för den specifika kandidaten.



Figur 7.10: Flervalsmeny för att flytta en kandidat från ett rekryteringssteg till ett annat. I figuren är kandidaten för närvarande i rekryteringssteg *Test Taken*.

Figur 7.11 visar hur den detaljerade kandidatvyn ser ut när flikarna *Acitivty Log* och *Messages* är aktiva. Under fliken *Messages* kan rekryterare se företagets meddelandehistorik med en kandidat och även välja att skicka nya meddelanden. Aktivitetsloggen visar alla

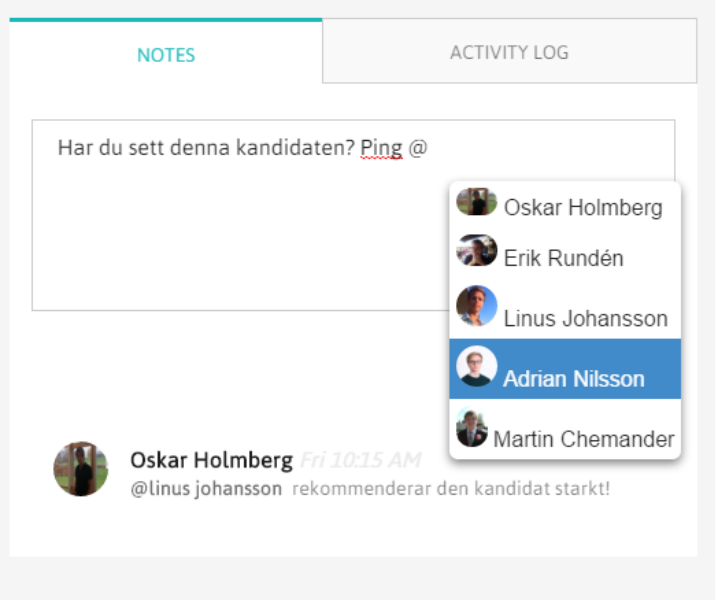
händelser som är relaterade till kandidatens, i detta fall *Emil Emiliasson*, ansökan.



Figur 7.11: Vy för detaljerad info om en specifik jobbansökan. Här är flikarna *Activity Log* och *Messages* aktiva. Meddelanden som skickas under fliken Messages blir även tillgängliga i meddelandesystemet som beskrevs i avsnitt 7.1.

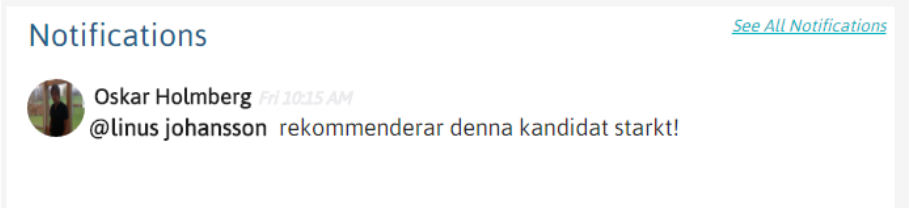
7.3 Mentions

Under fliken *Notes* i den detaljerade kandidatvyn ges möjlighet att nämna någon från företaget i en egen notering angående en kandidat (se Figur 7.12). Samtidigt som en notering sparas ner i databasen sparas också en referens till den nämnda personens konto och länkar på så vis samman personen med noteringen.



Figur 7.12: Omnämna kollegor i noteringar om kandidater. Här ser vi hur rullgardinsmenyn av rekryterare kommer upp när @ används i löpande text.

Omnämnandet i Figur 7.12 visas sedan upp under notifieringar på den omnämnda personens dashboard (se Figur 7.13).

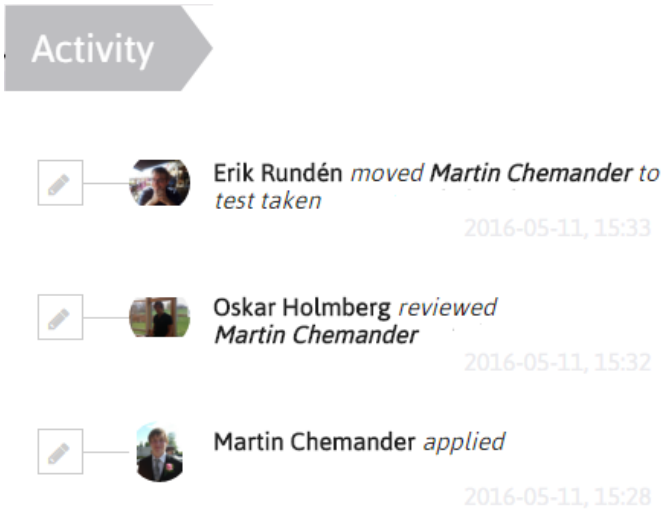


Figur 7.13: Exempel på en omnämning i en notering. Här har Oskar Holmberg nämnt Linus Johansson i sin notering.

7.4 Aktivitetslogg

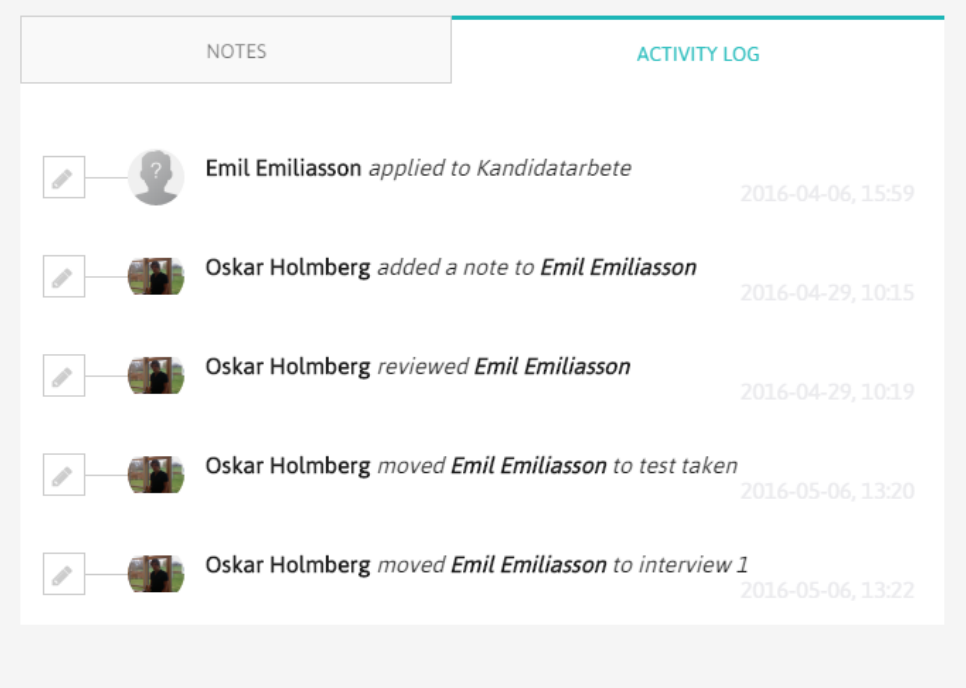
Samtidigt som ändringar utförs på applikationen sparas det också ner ett loggelement innehållandes information om vilken ändring som utförts och vem som utförde ändringen. Denna information visas sedan för rekryterare i form av en aktivitetslogg. Beroende på betraktarens roll på sidan samt vilken vy som för tillfället är aktiv visas denna data på ett av tre olika sätt. De olika vyer vi valt att visa aktivitetsloggen i är dashboard (Figur 7.14), vyn för ett specifikt jobb (Figur 7.15) och slutligen den detaljerade kandidatvyn (Figur 7.16).

Figur 7.14: Aktivitetsloggen för dashboardvyn. Här visas alla händelser som den inloggade rekryteraren är delaktig i.



Figur 7.15: Aktivitetslogg för ett specifikt jobb. Visar alla händelser som har skett för jobbet i fråga, allt från vem som först skapade jobbet till att en eventuell kandidat blivit erbjuden arbete.

Under vyn för ett specifikt jobb visas alla händelser från att jobbannonsen skapades tills dess att en kandidat blivit anställd. Exempel på jobbspecifika händelser visas i Figur 7.15.

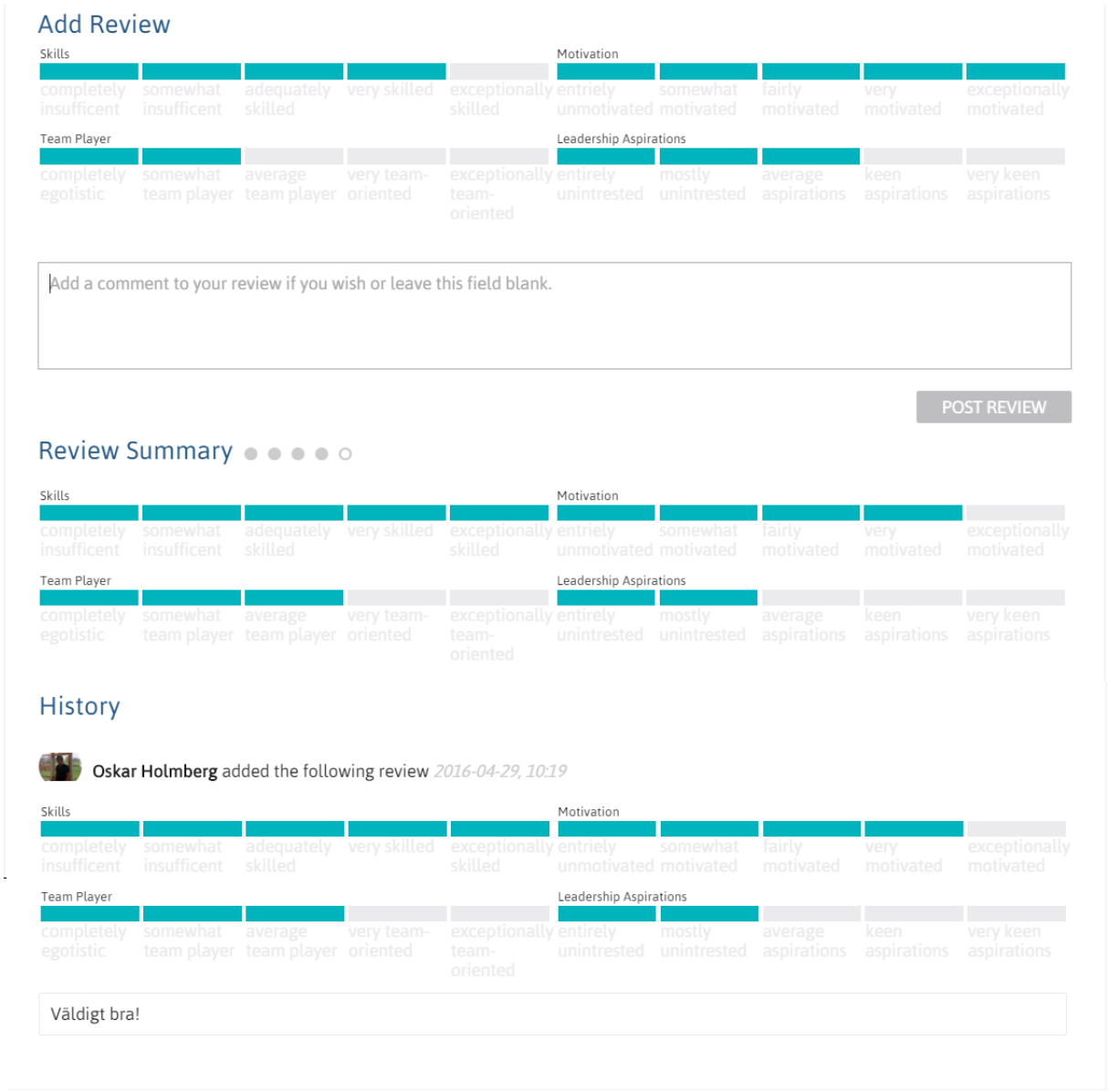


Figur 7.16: Aktivitetslogg för en specifik jobbansökan, i detta fall har kandidaten Emil Emiliasson sökt jobbet Kandidatarbete.

I Figur 7.16 visas enbart aktiviteter relaterade till Emil Emiliassons jobbansökan. Data visas enbart för rekryterare som arbetar med att tillsätta tjänsten ifråga. Kandidaten kan således inte se vare sig omdömen, noteringar eller vilket rekryteringssteg de för nuvarande befinner sig i.

7.5 Omdömen

Möjligheten att recensera en jobbansökan ges under den detaljerade kandidatvyn. I sin recension kan man betygsätta kandidatens färdigheter, ambitioner, hur pass bra lagspelare denne är samt ledarskapsförmåga, även en notering kan läggas till i en recension. Samtliga recensioner summeras ihop och presenteras, slutligen visas även historik över tidigare recensioner, för att se hur enskilda rekryterare resonerar kring en kandidat. I Figur 7.17 visas resultatet av detta.

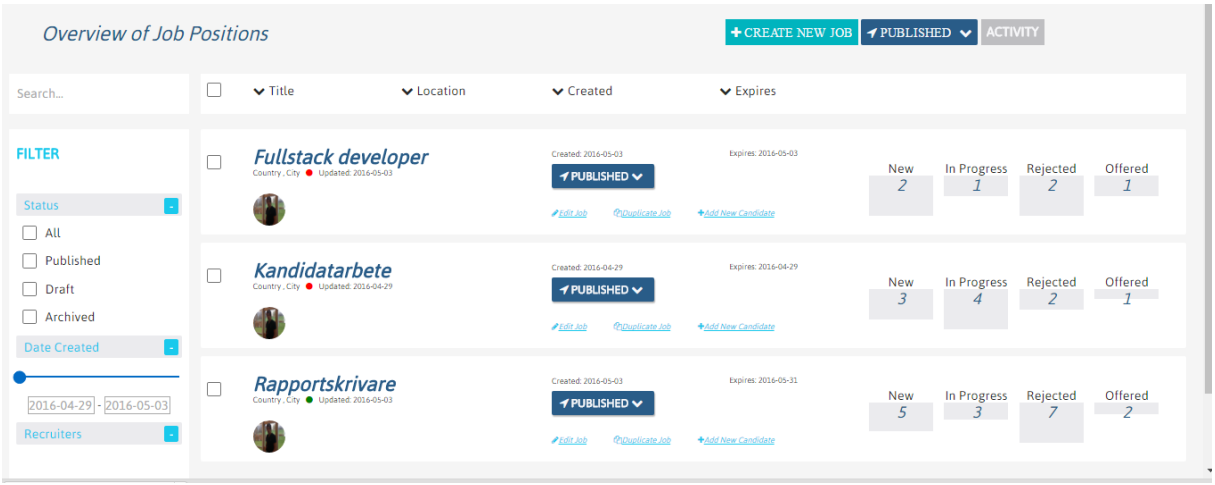


Figur 7.17: Omdömen beträffande en specifik jobbansökan. Här ges möjligheten att lägga till ett nytt omdöme, se en summering av tidigare omdömen samt se en historik över hur de enskilda omdömena såg ut.

7.6 Jobbvy

I denna vy av applikationen kan ett företag få en övergripande överblick av alla sina listade jobbannonser, såväl jobb som är aktiva för stunden som äldre arkiverade jobb (se Figur 7.18). Det visas även information om hur många kandidater som är placerade i respektive rekryteringssteg för varje jobb samt om jobbet är officiellt publicerat eller inte. De steg som inte tillhör något av de statiska stegen *new*, *rejected* eller *offered* (stegen förklaras i avsnitt 6.2) är i denna vy sammanslagna och visas under rubriken *In Progress* för respektive jobb. Annan information som tjänstens geografiska placering, när jobbannonsen skapades och

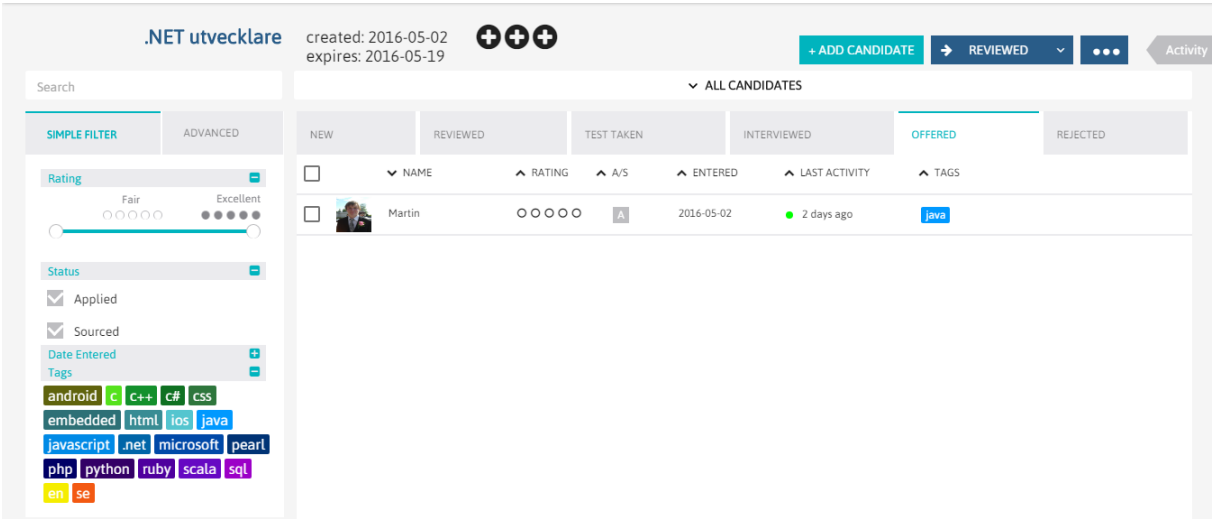
sista ansökningsdag visas också i denna vy. Det finns även möjlighet att filtrera på om jobben är officiellt publicerade eller inte samt en sökfunktion där det är möjligt söka på jobbnamn.



Figur 7.18: Denna vy visar alla listade jobb ett företag har och innehåller sök- och filtreringsfunktioner vilket underlättar när en rekryterare behöver söka efter mer specifika kriterier.

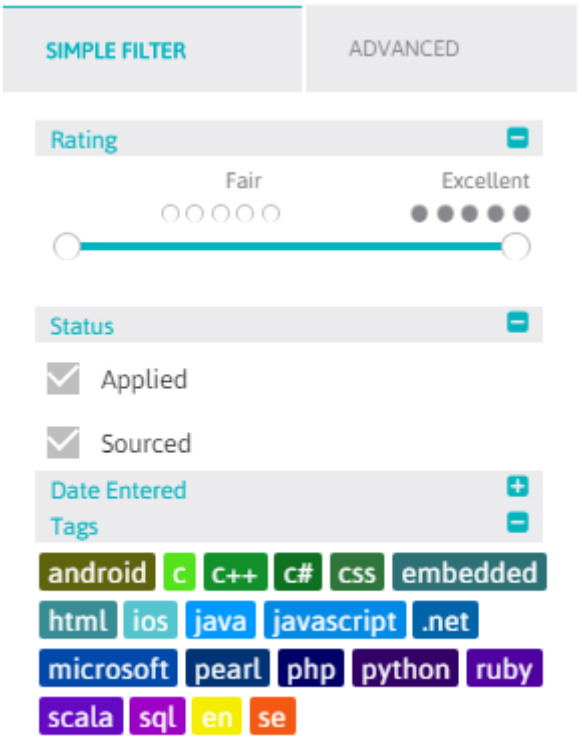
7.7 Specifik jobbvy

Ponera att vi från jobbvyn i Figur 7.18 nu vill se en detaljerad vy av ett specifikt jobb (säg .NET-utvecklare). Då kommer man till vyn som visas i Figur 7.19. Vyn skall ge en övergripande blick av ett specifikt jobb. Information om när jobbannonsen skapades samt sista ansökningsdag ges tydligt intill namnet på jobbet.



Figur 7.19: Detta är vyn för ett specifikt jobb. Här visas övergripande information om jobbet samt rekryteringsstegen.

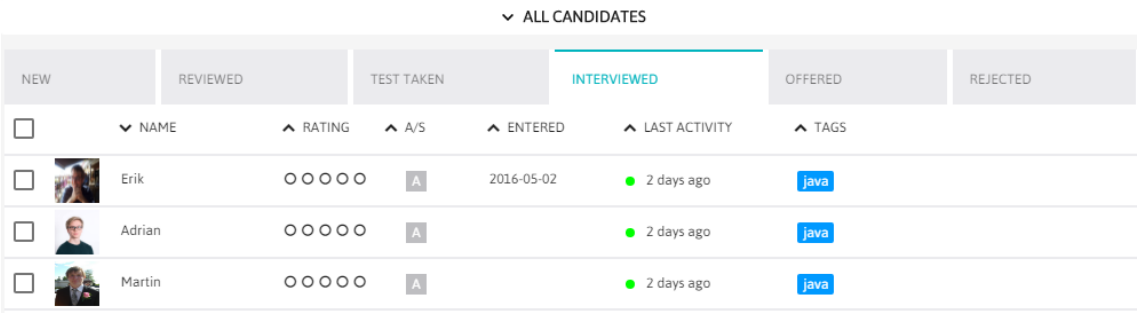
Vyn skall även ha en en filtreringsfunktion, som kan filtrera kandidater beroende på om de sökt själva (applied) eller blivit headhuntade av företaget (sourced). Filtrering ska även kunna göras efter omdömen, taggar eller färdigheter som jobbet kräver. Alla filtreringsmöjligheter visas i Figur 7.20. Dessa funktioner är dock ännu inte realiserade då den bakomliggande strukturen ej implementerades i tid.



Figur 7.20: Filtreringsflik som används till att filtrera kandidater efter färdigheter, sammanlagda omdöme (rating) och status.

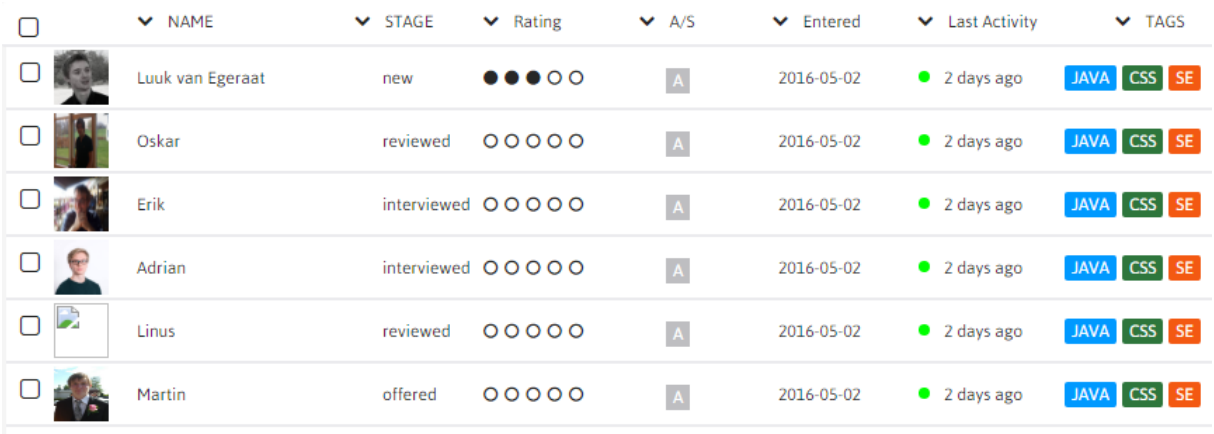
Om ett jobb har många kandidater och en rekryterare letar efter en särskild kandidat så finns en sökfunktion till hands som, likt sökfunktionen i meddelandesystemet, filtrerar bort allting som inte överensstämmer med sökordet.

Den specifika jobbvyn består huvudsakligen av ett antal flikar som representerar jobbets rekryteringssteg. Under varje flik återfinns de kandidater som befinner sig i det rekryteringssteget. I Figur 7.21 visas tre kandidater som befinner sig i steg *Interviewed*, d.v.s. steget där de precis har intervjuats. Under varje flik återfinns även information om de ingående kandidaterna. Denna information omfattar kandidatens namn, om kandidaten har ansökt själv eller blivit tillagd av rekryterare, omdöme från rekryterare, senaste aktivitet samt vilka färdigheter kandidaten har (se Figur 7.21).



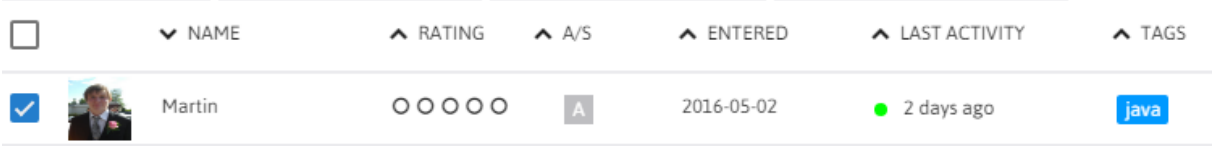
Figur 7.21: Här visas vilka kandidater som befinner sig i steget *Interviewed*, samt information om varje kandidats ansökan.

Skulle en rekryterare snabbt vilja hitta en kandidat utan att på förhand veta i vilket steg som denne befinner sig i, kan vyn ändras genom att klicka på All-candidates (se Figur 7.21). Detta ändrar vyn så att alla kandidater visas i en och samma lista (se Figur 7.22), istället för att ha dem uppdelade i rekryteringssteg under separata flikar.



Figur 7.22: Vyn som listar alla kandidater som sökt jobbet. Här visas alla sökande oavsett vilket rekryteringssteg en kandidat befinner sig i.

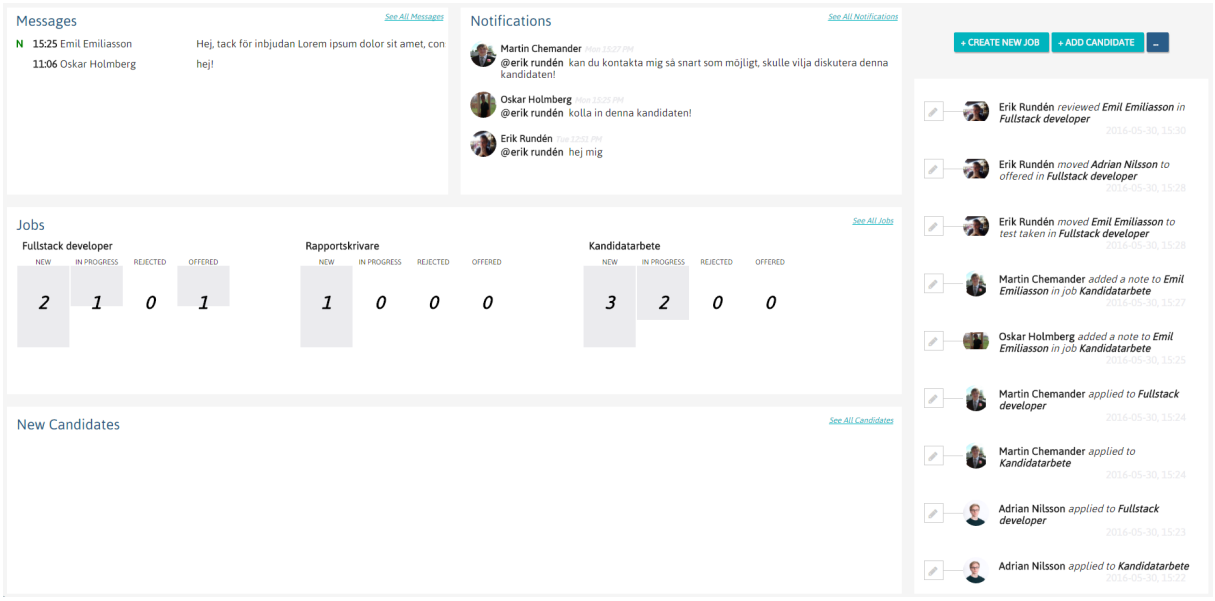
Möjlighet för rekryterare att flytta en kandidat till ett steg från ett annat görs med hjälp av flervalsmenyn i Figur 7.10. I den här jobbspecifika vyn kan en flytt mellan rekryteringssteg endast genomföras efter att en kryssruta tillhörande kandidaten som skall flyttas har fyllts i (se Figur 7.23).



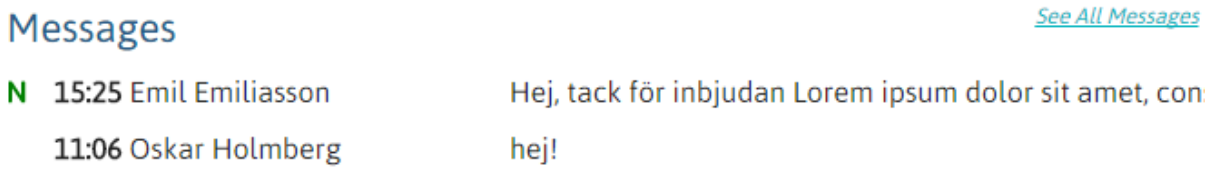
Figur 7.23: Kandidat som är redo att flyttas till annat steg. Notera att kryssrutan till vänster är ifylld.

7.8 Dashboard-vy

Dashboard är den första vyn som visas för rekryteraren vid öppnande av webbapplikationen. Vyns funktion är att ge en överskådlig blid av de senaste händelserna som berör rekryteraren (se Figur 7.24). Här samlas och sammanfattas alltså information från många andra vyer och sidan har även snabbknappar för att till exempel skapa nya jobb (förklaras i appendix A.2) och lägga till nya kandidater.

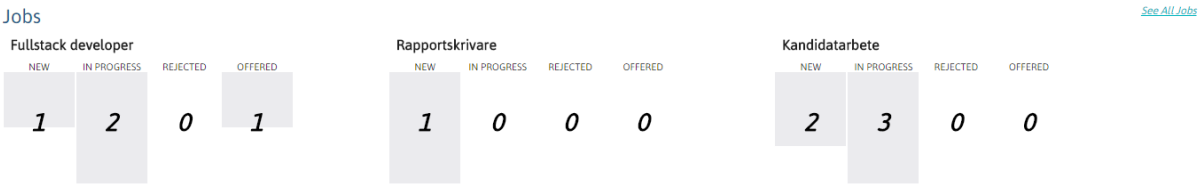


Figur 7.24: Överskådlig vy för rekryteraren där information om jobb och kandidater sammanfattas, det finns även snabbknappar i den högra överkanten för att skapa nya jobbannonser och lägga till nya kandidater.



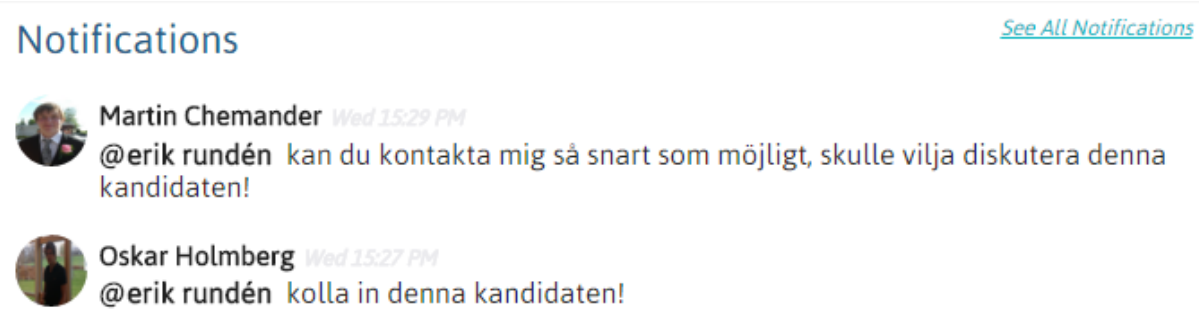
Figur 7.25: Rekryterarens två senaste meddelanden där det översta meddelandet är oläst, viket indikeras av ett *N* i vänstermarginalen. Knappen *See All Messages* leder till meddelandesystemet.

Figur 7.25 visar de senaste meddelandena som rekryteraren fått, sorterade efter datum, samt en eventuell markering längst till vänster som indikerar ett oläst meddelande.



Figur 7.26: Aktiva jobb som rekryteraren är inblandad i. Staplarna ändrar storlek beroende på antal kandidater som befinner sig i varje steg.

I Figur 7.26 visas en översikt av tre aktiva jobb där rekryteraren kan se antalet kandidater i olika steg av processen. Det finns även möjlighet att komma till en specifik vy, likt den i Figur 7.19, för respektive jobb genom att klicka på dem.



Figur 7.27: Omnämnanden från andra rekryterare.

I Figur 7.27 visas omnämnanden från andra rekryterare på företaget som är bundna till en viss kandidat. Dessa kommentarer är till för kommunikation mellan rekryterare angående specifika kandidater. Alltså har kandidaten i fråga har inte möjlighet att se kommentarer om sig själv.

8. Diskussion

Trots att nya ATS möjliggör en avancerad rekryteringsprocess har det bedrivits väldigt lite forskning inom detta område, och i synnerhet från en rekryterares perspektiv [2], [23]. På grund av bristen på forskning var det svårt att få fram ett underlag till konkret funktionalitet som gynnar samverkan för specifikt e-rekrytering. Istället fick vi förlita oss på dem allmänna förhållningssätt som kom fram under förstudien.

De förhållningssätt som förstudien påvisade var att den slutgiltiga webbapplikationen skall gagna kommunikation mellan rekryterare, ge kännedom om vilka aktiviteter som sker i webbapplikationen samt ha stöd för att koordinera dessa. Det här är kriterier som vi anser till viss mån uppfyllts, samtidigt som vi ser utrymme för innovation inom samtliga.

Transparens genomsyrar de flesta av dagens populära webbapplikationer, ofta i form av en aktivitetslogg som beskriver vem som gjort vad och när. Detta implementerades även i vår lösning. Vi menar även att en aktivitetslogg indirekt ger möjlighet att kommunicera aktiviteter och koordinera arbete mellan rekryterare.

Ett mer tydligt sätt att kommunicera och koordinera aktiviteter ges genom möjligheten att omnämna personer i sitt arbetslag i sina anteckningar. Även möjligheten att ge kandidater omdöme hjälper rekryterarna att koordinera aktiviteter och kommunicera med varandra då de visas för samtliga personer som är involverade i det specifika jobbet.

I samråd med personal på Software Skills implementerades meddelandefunktionen i form av skriftlig kommunikation mellan ett företag och en kandidat. Detta gjorde att det ännu inte finns något direkt sätt att kommunicera rekryterare emellan, eller sätta upp gruppkonversationer för arbetslag. Meddelandefunktionen har däremot fördelen att det ger ett direkt sätt för rekryterare att kommunicera med kandidater i webbapplikationen. Meddelandefunktionen underlättar rekryterarens arbetsuppgift avsevärt då kommunikation med kandidater har kommit att bli deras huvudsyssla [2].

8.1 Metoddiskussion

I detta avsnitt diskuteras projektets arbetsmetoder och hur dessa påverkat projektet.

8.1.1 Samarbete

Som nämndes i avsnitt 5, utvecklades inSource i samarbete med både företaget Software Skills och två masterstudenter från Interaktionsdesign. Detta samarbete har överlag fungerat mycket väl, men har även lett till vissa oklarheter. Många av dessa oklarheter berörde den grafiska design som utvecklades av de två masterstudenterna. Deras arbete bestod, likt detta projekt, av en iterativ process och således förändrades designen allt eftersom arbetet fortskred. Alltså har vi under största delen av projektet varit tvungna att arbeta mot temporära, halvfärdiga, designer som med jämna mellanrum förändrades, vilket i viss mån varit tidskrävande.

Den slutgiltiga designen av applikationens vyer kom väldigt sent in i projektet vilket också återspeglas i en jämförelse av det visuella resultatet av vår applikation och masterstudenternas design. Hade en slutgiltig design funnits innan projektet påbörjades hade resultatet förmodligen blivit mer exakt och således mynnat ut i en mer stilren produkt. Det bör dock nämnas att de tidiga designerna var mindre komplexa och lättare att implementera. Detta gjorde dem till lämpliga objekt för att träna upp våra kunskaper i de språk och ramverk som användes, vilket var nyttigt eftersom de flesta i gruppen saknade erfarenhet av webbutveckling.

8.1.2 Scrum

Att arbeta efter Scrum-ramverket har överlag fungerat bra även om ingen i gruppen hade någon större erfarenhet av att jobba med agila arbetssätt innan. Saker som kunnat förbättrats hade varit att sätta ett mindre antal och mer exakta mål om vad som skulle göras i varje sprint. På så vis hade fler mål i sprintarna blivit uppfyllda och inte behövts flyttas över till nästa sprint. Mer exakta mål hade också gjort målen lättare att utvärdera om de helt eller bara delvis uppfyllts.

8.2 Jämförelse av webbaserade och lokalt installerade plattformar

Att utveckla en webbaserad applikation föll sig ganska naturligt i detta projekt framför allt eftersom Software Skills tidigare plattform var webbaserad och integrationen av detta projekts prototyp skulle förenklas avsevärt. Det hade dock varit fullt möjligt att utveckla en icke webbaserad applikation, alltså en applikation som installeras lokalt via en exekverbar fil.

Då många jobbansökningar idag sker via internet, så även för Software Skills, hade en lokal applikation ändå behövt kommunicera med internet för att hämta in nya jobbansökningar

som kommer in till systemet. Dessa ansökningar skulle sedan kunnat sparas ner på lokala servrar. Fördelen med en sådan lösning hade varit att den största delen av systemet skulle kunna köras även om internetanslutningen bryts. Det enda som hade påverkats hade varit att systemet inte kunnat hämta in nya jobbansökningar.

En nackdel med lokalt installerade applikationer är att de oftast är operativsystemsberoende, vilket innebär att det måste utvecklas en version av applikationen till varje typ operativsystem som applikationen skall installeras på. En annan nackdel är att installering och uppdatering av mjukvaran måste ske på varje enskild enhet som använder applikationen. Dessa nackdelar existerar inte i webbaserade applikationer då allt detta sker på serversidan av applikationen.

För att ställa de ovannämnda för- och nackdelarna om en webbaserad eller lokalt installerad applikation i relation till projektets utgångspunkt så anser vi det webbaserade alternativet vara mer fördelaktigt. Det främsta argumentet för detta är att de inte kräver någon installation, vilket skapar ett mycket flexibelt system som på ett enkelt sätt kan användas från vilken webbläsare som helst vart man än befinner sig.

8.3 Resultatdiskussion och måluppfyllnad

Detta avsnitt är avsett diskutera applikationens funktionalitet samt i vilken utsträckning projektet uppfyllt den uppställda målsättningen.

8.3.1 Målet att ersätta Trello

Målet att ersätta Trellos funktionalitet uppfylldes till stor del. Det saknas dock en vy där ansökande kandidater är representerade som så kallade Trello-Cards och man genom att bara använda muspekaren kan dra kandidater mellan olika rekryteringssteg, en så kallad dra-släpp-funktion. Den främsta orsaken till att denna vy inte kunde implementeras var att den slutgiltiga designen färdigställdes sent. Dessutom är dra-släpp inte trivialt att implementera. På grund av dessa orsaker fanns det helt enkelt inte tid färdigställa denna vy trots att den var högt eftertraktad av Software Skills vd.

Däremot speglas Trellos funktionalitet i applikationens specifika jobbvy (se Figur 7.22). Användaren kan utföra samma saker i den specifika jobbvyn som tidigare genomfördes i Trello, fast på ett annat sätt. Skillnaden ligger främst i hur kandidater representeras. Den vy som implementerats ger en något mindre överskådlig bild av rekryteringens status (de specifika rekryteringsstegens ingående kandidater) och saknar dessutom dra-släpp-funktionalitet.

8.3.2 Målet att ersätta Google Drive

Det andra huvudsakliga målet med projektet var att ersätta Google Drive genom att all information och statistik om kandidater och jobb istället sparas ner i databasen. Med en vy där alla, både gamla och nya, kandidater ett företag har listas och görs sökbar skulle inSource också ersätta Google Drive. Dock hann vi inte implementera denna vy. Däremot finns en detaljerad vy av kandidaters enskilda jobbansökningar (se Figur 7.9), som till stor del blev fullständig. Därmed finns en del av den funktionalitet som Software Skills ville ha från Google Drive tillgänglig i den prototyp av inSource som skapats.

8.3.3 Meddelandesystemet

Att implementera ett meddelandesystem var inte ett av de övergripande målen med detta projektet. Med tanke på applikationens karaktär går det argumentera för att det hade varit smidigare att direkt mejla till dem som gjort en jobbansökan. Gruppen, tillsammans med Software Skills, ansåg emellertid att en meddelandefunktion ändå skulle vara användbar. Dels för att ansökningshanteringssystemet inte skulle behöva använda några externa tjänster överhuvudtaget, men också för att informationen sparas i Software Skills databas, vilket kan vara användbart då konflikter kan uppstå.

Ett meddelandesystem kan utformas på många olika sätt. Det skulle exempelvis kunna vara ett mer chattliknande system, likt Facebooks chattsystem, där man via mindre konversationsflikar kan skriva till kandidater. Vi ansåg dock att ett sådant meddelandesystem inte lämpade sig bra i ett ATS då man ofta vill ha en stor vy för att kunna skriva och läsa längre konversationer. Dessutom ansåg vi att ett mer chattliknande meddelandesystem skulle uppmuntra till mer alldaglig skrift och således ge ett mindre professionellt intryck.

Beslutet att använda oss av Websockets var ganska självklart då vi var tvungna för att få den funktionalitet som vi ansåg önskvärd. Detta innebar dock att diverse äldre webbläsare skulle missa denna funktionalitet. Webbsidans klientel är främst till IT-rekrytering och då utgår vi från att majoriteten inte använder sig av dessa - så mest troligt är att detta inte påverkar sidans användbarhet överhuvudtaget.

Trots att meddelandesystemet inte var ett övergripande mål ansåg vi det vara en väldigt vital del i ett ATS då kommunikation med kandidater har kommit att bliva en rekryterares huvudsyssla [2]. Därför ansåg vi det vara av stor vikt att meddelandesystemet färdigställdes, och eftersom designutvecklarna inte arbetade med meddelandesystemet skapades även bättre förutsättningar för detta vilket gjorde att meddelandesystemet slutligen nådde en alfaversion.

8.4 Framtida funktioner

Det finns många funktioner som inte implementerats i applikationen men som i ett framtidsperspektiv skulle höja applikationens värde som ett rekryteringsverktyg. Meddelandesystemet skulle exempelvis kunnat utformas på ett sätt som möjliggör individuella konversationer mellan kandidater och rekryterare. Detta hade höjt integriteten hos företagets rekryterare samtidigt som kandidaten fått ett mer personligt intryck. Konversationer mellan rekryterare skulle också vara en funktion som hade höjt applikationens användarvärde. Detta hade både främjat den sociala aspekten inom företaget samt ökat de interna kommunikationsmöjligheterna.

En bättre vy för aktivitetsloggen där äldre aktiviteter visas, samt möjligheten att filtrera på en viss person eller typ av aktivitet, kan bli ett framtida behov. I dagens implementation visas endast de senaste aktiviteterna vilket kan göra loggen svår att följa i stora arbetslag av rekryterare, eller då jobb får många sökande.

I en tid då mobilanvändandet växer kraftigt och används till mer än som enbart kommunikationsmedel hade det optimala varit att ändra om vyerna till en mobilanpassad vy då antal skärmpixlar gick under en viss bestämd gräns. Detta hanns inte med inom tidsrymden av detta projekt men är en självklar förändring i en vidareutveckling av applikationen.

8.5 Lärdomar av projektet

Projektet har varit mycket lärorikt för samtliga gruppmedlemmar, dels i det nära samarbete med både Software Skills och masterstudenterna men också i de programspråk vi kommit att lära oss. Arbetsprocessen har varit väl förankrat i hur det går till ute i arbetslivet, där vi jobbat agilt efter den kravspecifikation som lämnades när projektet påbörjades. Även den uppdelning där gruppens uppgift främst var att implementera den design som framställts av masterstudenterna reflekterar den arbetsprocess som företag bedriver.

Då Software Skills tidigare kodbas var en MEAN-stacklösning användes Javascript för att koda både front- och back-end. Detta betydde att vi inte var nämnvärt tvungna att dela upp arbetet, där några specialiserade sig i någon riktning. Detta resulterade i att vi kunde lära oss Javascript mer detaljerat istället för att lära oss olika programspråk för back-och front-end ytligt, och vi kunde på så vis utveckla en bättre produkt.

En annan erfarenhet vi tar med oss från projektet är hur snabbt webbutveckling förändras. Hela SPA arkitekturen med Angular kan var utdaterade inom loppet av några månader. Intressant nog hände just detta under projektet då Angular 2.0 gick in i en betaversion. Eftersom Angular 2.0 innebär stora förändringar i hur man arbetar med Angular kommer

troligtvis delar av vårt arbete inom kort vara föråldrad.

8.6 Säkerhetsaspekten

I projektets början var tanken att fokus skulle ligga på säkerhet, men arbetet lagt på detta blev inte så stort då Software Skills redan hade en existerande webbplattform med olika säkerhetsåtgärder. En av säkerhetsaspekterna gruppen kunde arbeta med var att begränsa vilken data som hämtades från databasen. När data hämtas finns det åtgärder som ser till att korrekt data returneras så inte obehöriga får tillgång till känslig information.

All typ av överföring som görs på webbapplikationen är krypterad eftersom protokollet HTTPS används [24]. Detta skyddar mot så kallade man-i-mitten attacker, som är en typ av attack där en person har tillgång till den data som skickas mellan två parter, och också har möjlighet att modifiera den.

8.7 Etiska dilemman

I utvecklandet av webbapplikationer som involverar personuppgifter där data lagras och behandlas finns det vissa aspekter som kräver stor försiktighet. Datalagring är till stor del begränsad av de EU-direktiv som finns uppsatta. Om ett företag som finns i ett flertal länder använder sig av inSource skulle en flytt av datalagring mellan nationer kunna bli problematisk [5]. Även användandet av information för andra ändamål än för det specifika jobbet bör ske med stor aktsamhet.

I de filer som involverade kontakt med databasen lades det särskild vikt vid att se till att varken kandidater eller administratörer som inte tillhör det relaterade företaget i fråga kunde ges möjligheten att få tillgång till känslig information.

Det finns idag en debatt angående diskriminering vid rekrytering av ny arbetskraft, något gruppen såg som en självklarhet att inte bidra till. Webbapplikationen har således igen möjlighet att filtrera ner kandidater beroende på egenskaper som inte går att styra över, exempelvis kön, etnicitet, ålder och sexuell läggning.

8.8 inSource i samhället

Användningsområdet för inSource är primärt utformat till rekrytering av arbetskraft, och har således svårt att appliceras in i andra delar av webben. Däremot skulle delar av systemet, så som meddelandefunktionen och omnämnanden, kunna användas i andra applikationer som främjar samverkan mellan användare. Applikationen är i motsats till

traditionell rekrytering helt digital, och medför därav ingen pappersförbrukning. Detta menar vi är gynnsamt för miljön. Denna digitala typ av rekrytering har även visats kunna minska tiden för en rekryteringsprocess [1] vilket är direkt gynnsamt för företag. De kan fokusera mer av sin tid på annat håll, eller hålla igång fler jobbannonser samtidigt. Även jobbsökande gagnas av detta då snabba beslut leder till snabbare anställning. De sociala aspekter som inSource främjar handlar primärt om samarbete mellan rekryterare, och fokus ligger på att sammansluta användarna snarare än att isolera dem. Meddelandefunktionen är direkt gynnsam för kommunikation med jobbsökande.

Det finns dock en annan sida av myntet i utvecklingen av en webbaserad applikation. Större delen av kommunikationen med kandidater sker över internet vilket försvårar möjligheten för rekryterare att bedöma beteende och social kompetens [25], och kandidater kan dessutom uppleva kommunikationen som opersonlig. När en kandidat söker går det inte heller att säkerställa dennes identitet, applikationen bygger därav på tillit mellan rekryterare och kandidater som kan leda till att värdefull tid går till spillo vid fall av felaktigheter. En annan nackdel är att meddelanden för tillfället lagras i klartext i databasen, något som skulle kunna leda till att användarnas integritet äventyras vid ett intrång.

En bredare problematik med e-rekryteringssystem som inSource är att det blir allt svårare för kandidater att skilja sig ur mängden. Medan företagen avnjuter ett större och bredare urval av kandidater måste kandidaterna kämpa hårdare för att visa sin konkurrenskraft och göra sig synliga för företagen. Det är inte längre invånarna på den lokala orten som konkurrerar om jobben, utan snarare hela landet eller till och med hela världen.

9. Slutsats

Rapportens syfte var att undersöka vilken funktionalitet som är central i webbapplikationer som främjar samverkan mellan dess användare. Tillsammans med projektets syfte och mål, att ersätta de av Software Skills nyttjade externa tjänster Trello och Google Drive, resulterade detta i en helhetslösning av ett ATS där kommunikation ligger i fokus. Förstudien visade att webbapplikationen bör innehålla stöd för kännedom, kommunikation samt koordination. Detta integrerades sedan in i andra centrala delar av ett ATS i form av ett meddelandesystem, möjligheten att omnämna andra rekryterare, betygsätta kandidater och en aktivitetslogg.

Software Skills har uttryckt att de är nöjda med resultatet, och ämnar även vidareutveckla den resulterande prototypen. Meddelandesystemet designades av gruppen själva utefter Software Skills krav och önskemål. Detta resulterade i att vi inte behövde vänta på en design, utan kunde påbörja utvecklingen direkt efter planeringen av projektet. Gruppen ämnade få meddelandesystemet till produktionsnivå, något vi i mångt och mycket lyckats med då meddelandesystemet integrerades i alfaversionen av Software Skills nya webbplattform.

Den design och ytterligare funktionalitet som tillkom i det absoluta slutet av projektet hann tyvärr inte implementeras helt. De tidiga skisserna som gruppen arbetat utifrån gjorde dock att stora delar av funktionaliteten i problemspecifikationen i slutändan implementerades.

Litteraturförteckning

[1] I. Lee, "The Evolution of E-Recruiting: A Content Analysis of Fortune 100 Career Web Sites," *Journal of Electronic Commerce in Organizations*, vol. 3, nr. 3, ss. 57–68, jul 2005. [Online]. URL: http://proxy.lib.chalmers.se/login?url=http://search.proquest.com/docview/236483143?accountid=10041 [Hämtad: 2016-05-16].

[2] A. B. Holm, "E-recruitment: Towards an Ubiquitous Recruitment Process and Candidate Relationship Management," *Zeitschrift für Personalforschung*, vol. 26, nr. 3, ss. 241–259, 2012. [Online]. URL: http://proxy.lib.chalmers.se/login?url=http://search.proquest.com/docview/1032552477?accountid=10041 [Hämtad: 2016-05-16].

[3] M. Armstrong och S. Taylor, *Armstrong's handbook of human resource management practice*, 13 uppl. London, United Kingdom: Kogan Page Limited, 2014, kap. 18, s. 229.

[4] A. B. Holm, "Institutional context and e-recruitment practices of Danish organizations," *Employee Relations*, vol. 36, nr. 4, ss. 432–455, 2014. [Online]. URL: http://dx.doi.org.proxy.lib.chalmers.se/10.1108/ER-07-2013-0088 [Hämtad: 2016-05-16].

[5] P. Cappeli, "Making the Most of On-Line Recruiting." *Harvard Business Review*, vol. 79, nr. 3, ss. 139–146, 2001. [Online]. URL: http://proxy.lib.chalmers.se/login?url=http://search.ebscohost.com.proxy.lib.chalmers.se/login.aspx?direct=true&db=buh&AN=4147424&site=ehost-live&scope=site [Hämtad: 2016-05-28].

[6] L. A. Guerrero och D. A. Fuller, "A pattern system for the development of collaborative applications," *Information and Software Technology*, vol. 43, nr. 7, ss. 457–467, 2001. [Online]. URL: http://www.sciencedirect.com/science/article/pii/S0950584901001549 [Hämtad: 2016-05-30].

[7] M. Koch och T. Gross, "Computer-supported cooperative work - concepts and trends," i *Proceedings of the 11th Conference of the Association Information and Management (AIM)*. Bonn: Köllen Verlag, jun. 2006, ss. 165–172. [Online].

URL: http://www.kooperationssysteme.de/docs/pubs/KochGross2006-aim-cscw.pdf [Hämtad: 2016-05-28].

[8] W3C, "HTML & CSS." [Online]. URL: https://www.w3.org/standards/webdesign/htmlcss [Hämtad: 2016-05-05].

[9] A. Freeman, *Pro AngularJS*. Berkeley, CA: Apress, 2014, kap. Putting AngularJS in Context, ss. 45–54. [Online]. URL: http://dx.doi.org/10.1007/978-1-4302-6449-1_3

[10] G. Fink och I. Flatow, *Pro Single Page Application Development: Using Backbone.js and ASP.NET*. Berkeley, CA: Apress, 2014, kap. Introducing Single Page Applications, ss. 3–13. [Online]. URL: http://dx.doi.org/10.1007/978-1-4302-6674-7_1

[11] MEAN.IO. [Online]. URL: http://mean.io/ [Hämtad: 2016-05-08].

[12] Db-engines, database popularity rankings. [Online]. URL: http://db-engines.com/en/ranking/ [Hämtad: 2016-05-08].

[13] A. Freeman, *Pro AngularJS*. Berkeley, CA: Apress, 2014. [Online]. URL: http://dx.doi.org/10.1007/978-1-4302-6449-1_3

[14] S. Pasquali, *Mastering Node.js*. Olton, Birmingham, GBR: Packt Publishing Ltd, 2013. [Online]. URL: http://site.ebrary.com.proxy.lib.chalmers.se/lib/chalmers/detail.action?docID=10813424

[15] V. Wang, F. Salim, och P. Moskovits, *The Definitive Guide to HTML5 WebSocket*. Berkeley, CA: Apress, 2013, kap. Introduction to HTML5 WebSocket, ss. 1–12. [Online]. URL: http://dx.doi.org/10.1007/978-1-4302-4741-8_1

[16] Primus – ett abstraktionslager för realtidsramverk i Node.js. [Online]. URL: https://github.com/primus/primus [Hämtad: 2016-05-01].

[17] Primus Rooms – En modul till Node.js som utökar Primus med möjligheten att skapa rum. [Online]. URL: https://github.com/cayasso/primus-rooms [Hämtad: 2016-05-13].

[18] Async.js – kraftfulla funktioner för asynkron JavaScript. [Online]. URL: https://github.com/caolan/async [Hämtad: 2016-05-04].

[19] Redactor – En textredigerare för webben. [Online]. URL: https://imperavi.com/redactor/docs/how-to-install/ [Hämtad: 2016-05-13].

[20] AngularUI Router – Ett navigationsramverk för Single Page Applications till Angular. [Online]. URL: https://github.com/angular-ui/ui-router [Hämtad: 2016-05-30].

[21] C. Skaskiw. (2013) Scrumguiden. [Online]. URL: http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-SE.pdf [Hämtad: 2016-05-12].

[22] S. Chacon, G. Cornell, J. Gennick, M. Lowman, M. Moodie, J. Pepper, F. Pohlmann, B. Renow-clarke, D. Shakeshaft, M. Wade, och T. Welsh, "Pro Git," *Control*, ss. 1–210, 2009. [Online]. URL: http://www.springerlink.com/index/10.1007/978-1-4302-1834-0 [Hämtad: 2016-05-13].

[23] D. L. Stone, D. L. Deadrick, K. M. Lukaszewski, och R. Johnson, "The Influence of Technology on the Future of Human Resource Management," *Human Resource Management Review*, vol. 25, nr. 2, ss. 216–231, jan 2015. [Online]. URL: http://www.sciencedirect.com/science/article/pii/S1053482215000030 [Hämtad: 2016-05-30].

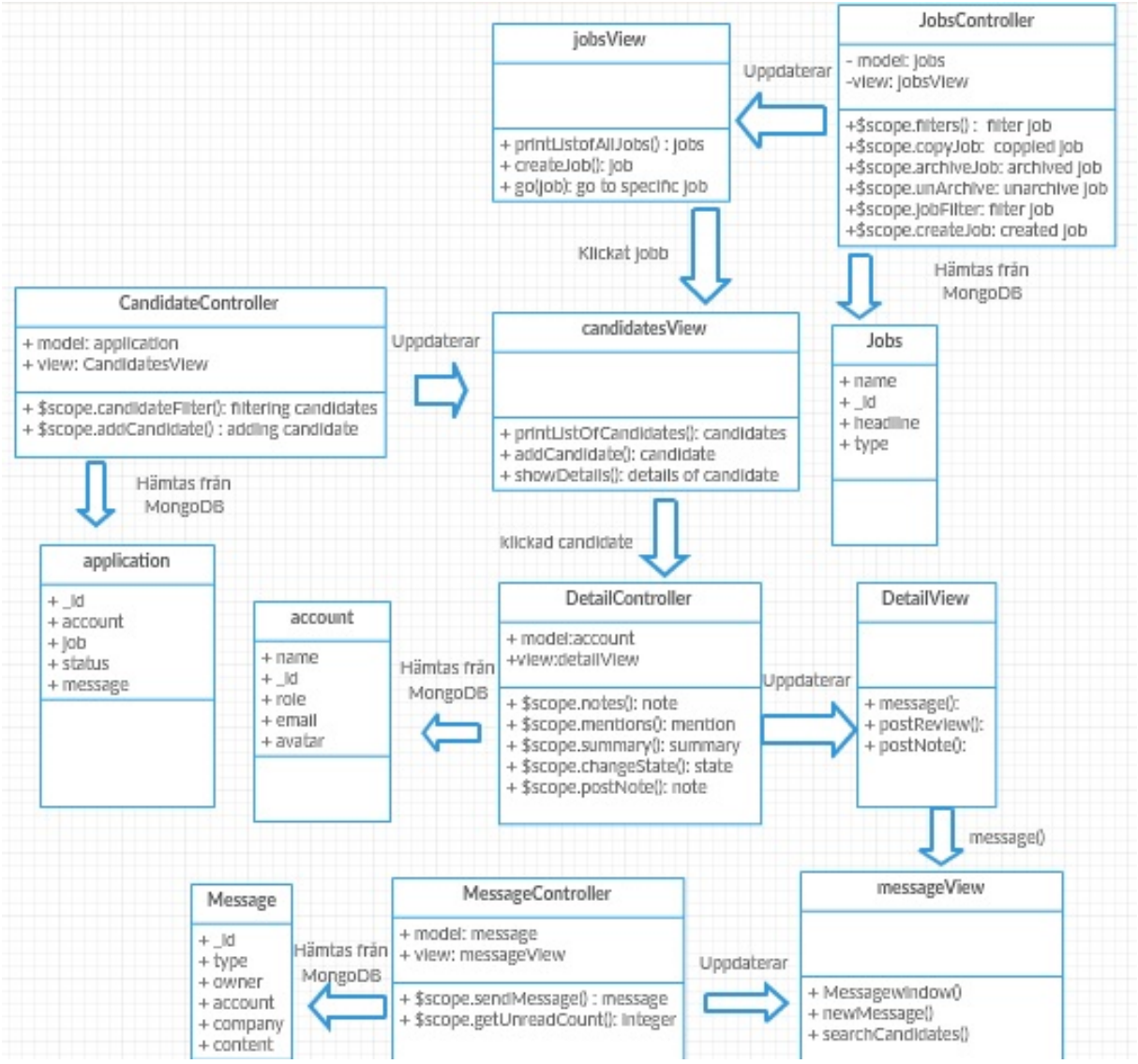
[24] IETF – HTTP Over TLS. [Online]. URL: https://tools.ietf.org/html/rfc2818 [Hämtad: 2016-05-15].

[25] E. Ettinger, C. Wilderom, och H. Ruel, "Service-Quality Criteria of Web Recruiters: A Content Analysis," i *2009 42nd Hawaii International Conference on System Sciences*. IEEE, 2009, ss. 1–10. [Online]. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4755609 [Hämtad: 2016-05-15].

A. Appendix

A.1 MVC-Modell

I figur A.1 ges en överblick av vyer, controllers, data som skapats till ATS-prototypen samt hur dessa förhåller och samverkar med varandra. Vi kan exempelvis se att JobsController hämtar data från databasen i form av Jobs, som är delvis uppbyggd av name, id, type och headline. Controllern uppdaterar sedan job-vyn som har funktioner som printListofAlljobs, createJob, go(job).

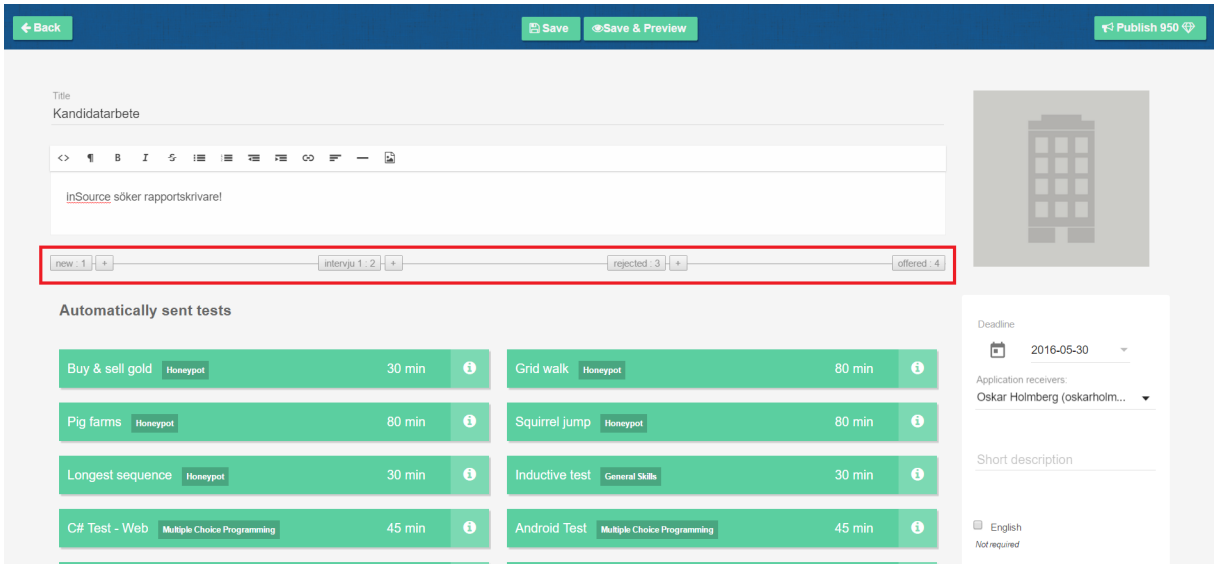


Figur A.1: MVC-modell över ATS-systemet.

A.2 Skapa jobbannonser

I Figur A.2 beskrivs skapandet av en jobbannons från en rekryterares perspektiv. I denna vy kan en rekryterare redigera all information rörande det specifika jobbet, allt från att ge en kort beskrivning av jobbet till att välja vilka kodtester som skall skickas till de personer som söker jobbet.

Funktionaliteten och vyn för att skapa en jobbannons var något som redan var implementerat i Software Skills tidigare system och omfattades således inte av detta projektet. För att inSource skulle kunna behandla kandidater på liknande sätt som Trello, det vill säga att ha kandidater placerade i olika rekryteringssteg, utökades dock denna vy. Denna utökade funktionalitet åskådliggörs inom det rödmarkerade området i Figur A.2.



Figur A.2: I denna vy kan en rekryterare skapa och redigera jobbannonser. Det som återfinns inom det markerade området är den extra funktionalitet som detta kandidatarbete implementerat i Software Skills redan befintliga jobbredigeringsvy.

Inom det markerade området i Figur A.2 kan en rekryterare utöver de statiska stegen *new*, *rejected* och *offered*, dynamiskt lägga till nya rekryteringssteg för att anpassa jobbet för dess specifika rekryteringsbehov.