Mendelova univerzita v Brně
Provozně ekonomická fakulta

Webový dashboard a REST rozhraní pro účely HR oddělení firmy SDE

Bakalářská práce

Vedoucí práce:
Ing. Jiří Balej

Karel Šimon

Brno 2016

Poděkování

Rád bych tímto poděkoval Radimu Poláškovi, za jeho věcné rady ke kódu API rozhraní Gazelle; Kamilu Vermiřovskému a Lence Novákové, za připomínky ke vzhledu a fungování systému; Ing. Jiřímu Balejovi za vedení této bakalářské práce.

Čestné prohlášení

Prohlašuji, že jsem tuto práci: **Webový dashboard a REST rozhraní pro účely HR oddělení firmy SDE**
vypracoval samostatně a veškeré použité prameny a informace jsou uvedeny v seznamu použité literatury. Souhlasím, aby moje práce byla zveřejněna v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů, a v souladu s platnou *Směrnicí o zveřejňování vysokoškolských závěrečných prací.*
Jsem si vědom, že se na moji práci vztahuje zákon č. 121/2000 Sb., autorský zákon, a že Mendelova univerzita v Brně má právo na uzavření licenční smlouvy a užití této práce jako školního díla podle § 60 odst. 1 Autorského zákona.
Dále se zavazuji, že před sepsáním licenční smlouvy o využití díla jinou osobou (subjektem) si vyžádám písemné stanovisko univerzity o tom, že předmětná licenční smlouva není v rozporu s oprávněnými zájmy univerzity, a zavazuji se uhradit případný příspěvek na úhradu nákladů spojených se vznikem díla, a to až do jejich skutečné výše.

Brno 26. prosince 2016 .....

Abstract

Šimon, K. Web dashboard and REST interface for HR department of company SDE Brno. Bachelor thesis. Brno: Mendel University, 2016.

This thesis deals with the design and implementation of Gazelle system which downloads, filters and displays data about job applicants.

Keywords

Information system, hackerrank.com, jQuery, Haml, Javascript, Ruby, framework Sinatra

Abstrakt

Šimon, K. Webový dashboard a REST rozhraní pro účely HR oddělení firmy SDE. Bakalářská práce. Brno: Mendelova univerzita v Brně, 2016.

Tato práce se zabývá návrhem a implementací systému Gazelle, který stahuje, filtruje a zobrazuje data o uchazečích o zaměstnání.

Klíčová slova

Informační systém, hackerrank.com, jQuery, Haml, Javascript, Ruby, framework Sinatra

Obsah

1 Úvod

1.1 Představení problému

Projekt Gazelle vznikl na popud personálního oddělení, které musí zpracovávat spoustu informací uchazečů o zaměstnání. Tyto informace jsou rozmístěny po různých webových aplikacích, jako jsou např. Google Docs, které umožňují současnou práci více osob najednou, kdy dokument je uložen na veřejném serveru (Jonathan Strickland, 2008); Hackerrank.com, který slouží pro online testování potenciálních uchazečů o zaměstnání (About Us, 2015); Teamio.com, které slouží pro zobrazení základního přehledu o uchazeči (S námi to zvládnete, 2016).

Z tohoto důvodu bylo vedením společnosti Software Development Europe s.r.o. (dále již jen SDE) rozhodnuto o vývoji webové stránky, která bude shromažďovat část dat na jednom místě pro rychlejší vyhledávání a reakci personálního oddělení na uchazeče o zaměstnání.

1.2 Cíl práce

Cílem práce je vytvořit rozhraní, které zpřehlední a urychlí práci se zájemci o zaměstnání ve firmě SDE. V první řadě je nutné prozkoumat již existující systémy a zjistit, zda se nedají přizpůsobit, aby splňovaly požadavky firmy. V případě, že nebude možné využít žádný existující systém, bude nutné ho navrhnout a implementovat. Výsledkem tak bude samostatný informační systém, ve kterém budou uloženy informace o uchazečích o zaměstnání a který bude umět komunikovat se stávající online testovací platformou hackerrank.com.

1.3 Přehled programovacích jazyků

Pro vytvoření RESTové rozhraní je použit jazyk Ruby s frameworkem Sinatra, který je vhodný pro programování backend systémů (Hal Fulton, 2003).

Databáze využívá technologii PostgreSQL (Bruce Momjian, 2003). Jazyk Ruby a PostgreSQL budou využity pro vytvoření backendu.

Samotná stránka se skládá z jazyka Haml, který je vhodný pro tvorbu dokumentů a aplikací. Rozdíl oproti jazyku HTML je ten, že Haml používá lehce rozdílnou strukturu HTML značek a kód napsaný v Hamlu je kratší (Haml About, 2015).

Pro funkcionalitu stránky je použit skriptovací jazyk (Nicholas Zakas, 2009) Javascript s knihovnou jQuery.

1.4 Stručný přehled jednotlivých kapitol

Ve druhé kapitole se budu věnovat stávajícím metodikám při výběru vhodných uchazečů o zaměstnání. Popíši jednotlivé části přijímacího pohovoru. V kapitole č. 3 popíši agilní metodiky vývoje software a zaměřím se na popis scrum metodiky. V kapitole

č. 4 popíši požadavky na systém Gazelle. V kapitole č. 5 se zaměřím na popis vybraných programovacích jazyků, které jsem využil při tvorbě systému. V kapitole č. 6 lze nalézt strukturu databáze a data-flow diagram systému. V kapitole č. 7 popíši vlastní implementaci systému. V kapitole č. 8 popíši funkce systému. V kapitole č. 9 vyhodnotím požadavky na systém vůči naprogramovanému systému. Kapitola č. 10 obsahuje závěr.

2 Stávající řešení

2.1 Popis procesu při náboru

Uchazeč, který se rozhodne, že se bude ucházet o zaměstnání, musí nejdříve zaslat email s průvodním dopisem, ve kterém objasní, proč by on byl nejlepší volbou pro danou volnou pozici. Tento průvodní dopis by neměl být příliš dlouhý a měl by vystihovat podstatu, proč se člověk hlásí na danou pozici.

Po přijmutí emailu ze strany personálního oddělení je tento požadavek nejprve zpracován ručně personalistkou. Zde se rozhodne, jestli je uchazeč v prvním kole vhodný na pozici (splňuje kritéria). Pokud nesplňuje požadavky, nebo jiným způsobem nevyhovuje, je odmítnut. Pokud postoupil do dalšího kola, uchazeč je upozorněn, že mu přijde druhý email, ve kterém nalezne odkaz na webovou testovací platformu. Tento test má platnost 7 dnů. Zde nastupuje na řadu právě zmiňovaná platforma HackerRank (funkcionalitě této platformy se budu věnovat v následující kapitole). Zde se zohledňuje, o jakou pozici uchazeč žádá a pošle se mu příslušný vygenerovaný test, který se vztahuje pro jeho pozici. Po vyplnění testu bude informován o výsledku online.

Poté, co kandidát tento test napíše, manager nemůže v této platformě jednoduše dohledat informaci, že kandidát již test napsal. Všechny testy, které jsou vytvořeny, se musí zdlouhavě projít a následně zjistit, který kandidát má již napsaný test, ale zároveň tento test nesmí být opravený. Jakmile manažer tento test opraví, je uchazeči sdělen výsledek s tím, že buď se rovnou domluví na termínu telefonického pohovoru, nebo mu je sdělena špatná zpráva s tím, že není vhodným kandidátem. Všechny tyto informace o pohovoru a výsledku testů jsou uloženy v separátních službách, např. Google Docs, Hackerrank.com nebo Teamio.com.

Další kolo se skládá z telefonního pohovoru, kde se zaměstnanec náborového oddělení spojí s uchazečem a ptá se na předem připravené otázky ohledně jeho znalostí, jeho osobě, oblíbených a neoblíbených programovacích jazycích a zároveň ho informuje o projektech, na kterých firma spolupracuje a co by bylo jeho náplní práce. Poté následuje krátký rozhovor v anglickém jazyce. Znalost anglického jazyka je v této firmě nezbytná, jelikož ve většině projektů se spolupracuje s americkou firmou a tudíž každý den se minimálně na meetingu komunikuje v angličtině. V mnoha případech se však musí komunikovat anglicky i mimo čas vyhrazený pro meeting.

Pokud uchazeč projde, je s ním následně domluven termín pohovoru. Na tomto pohovoru se poprvé setká se svým možným manažerem, který se stará o tým, do kterého může být přidělen. Po tomto pohovoru je buď uchazeč přijat a domluví se datum nástupu do zaměstnání a nebo odmítnut.

Data o náboru zaměstnance jsou rozmístěny na mnoha službách a hledání všech těchto informací je velmi zdlouhavé, a proto se vedení společnosti SDE rozhodlo, že větší část těchto informací se bude uchovávat pouze na jednom místě, aby se personálnímu oddělení usnadnila práce a zrychlil se tento proces hledání. Proto vznikl projekt Gazelle, který má tyto informace shromažďovat na jednom místě

a měl by umožňovat jejich procházení, upravování, hledání a řazení a tím usnadnit a zrychlit celé výběrové řízení.

2.2 Popis platformy hackerrank.com

Když uchazeč projde prvním sítem přijímacího řízení a postoupí do druhého kola, je mu odeslán email, ve kterém se uchazeči pošle odkaz na online test, který musí do týdne vyplnit.

Po kliknutí na odkaz v emailu se uchazeč přesměruje na uvítací stránku, kde si může přečíst veškeré informace jako například čas na dokončení testu, strukturu testu. Na této stránce také zadá základní osobní údaje, jako jsou například: jméno, příjmení, atd. Pak se mu zobrazí již první otázky. Test končí, pokud uchazeči dojde vymezený čas a nebo po kliknutí na tlačítko ukončení testu. Poté je test dokončen a následuje uložení do databáze a nastavení příslušných interních statusů, jimž se budu věnovat později.

Vytváření nových testů provádí manažer, který si může vytvořit libovolné množství testů. Do nově vytvořeného testu se mohou vložit nové otázky, které mají formu otevřené nebo uzavřené otázky. Ke každé otázce se může vložit doprovodný text, ve kterém se vysvětlí uchazeči, co má se zadaným úkolem dělat. Také se může přidat text s výsledky pro osobu, která bude test opravovat. Je zde také možnost přidat otázky, které jsou již přípravené od společnosti hackerrank. Tyto otázky jsou samozřejmě v anglickém jazyce, což by pro případné uchazeče neměl být problém. Test je i možné rozdělit na několik sekcí a každé sekci se může přiřadit jiný časový limit. Po dokončení vkládání všech otázek se test uloží. Po tomto kroku je již možné zvát nové kandidáty na vytvořený test.

Poté, co uchazeč odešle napsaný test, se mu nastaví několik rozdílných statusů, které se uchovávají v databázi hackerranku. Pomocí těchto statusů se pak dále rozlišuje stav, ve kterém se uchazeč nachází. Nejzákladnější status se nazývá user_status, ten se přiřadí uchazeči hned, jak je pozván na test. Výchozí hodnota je -1. Tím systém říká, že kandidát byl pouze pozván k vypracování testu, ovšem test nebyl zatím napsán. Poté, co je test napsán, se user_status změní na hodnotu 7. Toto číslo značí, že kandidát již test napsal a je proto možné test opravit. Také se mu přiřadí další 2 id statusy, ID a ats_status. Pokud je ats_status přiřazen, tak user_status se již dále nemění a zůstává na hodnotě 7. Pomocí ats_statusu lze zjistit, jestli již byl test opravený a zda uchazeč uspěl nebo naopak (Atlassian, 2012). Přehled všech ats statusů lze nalézt v tabulce č. 1.

Tabulka 1: seznam ats statusů (Atlassian, 2012)

0	ATS State Not Set
1	Test Completed - Evaluation Reqd.
2	Test Completed - Qualified
3	Test Completed - Failed
4	Phone Interview - I
5	Phone Interview - II
6	Phone Interview - III
7	Offer Sent
8	Offer Negotiation
9	Offer Accepted
10	Offer Declined
11	On Hold
12	Phone Interview - Cleared
13	Phone Interview - Failed
14	Technical Interview - Cleared
15	Technical Interview - Failed
16	HR Interview - Cleared
17	HR Interview - Failed

Pro získání informací o uchazečích se využívá RESTové rozhraní a HTTP metody GET, POST, PUT, DELETE (Andrew Havens, 2012). Server odpovídá na všechny požadavky v JSON formátu, který je vhodný pro komunikaci mezi počítačovými platformami (Introducing JSON, 2016). Lze využít například metody getTests, getTestCandidates, getTestCandidatesById a několik dalších, které umožňují základní operace nad daty (HackerRank V2 API, 2015).

Nevýhoda tohoto řešení spočívá v omezené funkcionalitě, například chybějící funkce delete, která by smazala kandidáta z databáze hackerranku. Proto je nutné, pokud například osoba z personálního oddělení udělá chybu při zadávání, nebo pokud již uchazeč nemá zájem o danou pozici, smazat uchazeče z databáze Gazelle a následně i z databáze hackerranku. Dalším a největším problémem je rychlost stahování všech informací. Pokud již firma má přibližně 300 uložených uchazečů, trvá stahování všech informací asi 1-2 minuty. Pokud jich však má ještě více trvá stahování ještě déle. Problém není v samotné rychlosti vyřizování požadavků, ale v logice získávání informací. Nejdříve se totiž musí zavolat odkaz pro stažení všech testů. Tato data se musí rozparsovat a zjistit ID ke každému testu. Toto ID se poté vloží do další URL adresy a stáhnou se všichni uchazeči.

Pokud však budeme chtít stáhnout i komentáře k testu, musí se navíc zkontrolovat, jestli uchazeč má už vygenerované ID. Pokud má, volá se další odkaz pro každého uchazeče zvlášť, který se skládá z ID testu a ID uchazeče. Místo několika dotazů tedy musíme provést řádově stovky dotazů, pokud databáze čítá stovky uchazečů, což při pár měsících používání je reálné. A toto dotazování trvá neúměr-

ně dlouho. Bohužel hackerrank nemá v datech, která se získají z jejich api, žádné časové razítko. Proto nelze zjistit, jestli se nějaká data změnila a proto pro zachování aktuálních dat se musí provádět velmi často synchronizace databáze Gazelle a hackerranku.

2.3 Existující systémy

Předtím, než se začalo přemýšlet o vývoji úplně nového systému, se samozřejmě vedení firmy SDE zajímalo o konkurenční webové systémy. Úplně nejdůležitějším kritériem pro nový systém bylo propojení s testovací platformou www.hackerrank.com. Aplikaci hackerrank bohužel nelze provozovat zcela samostatně, jelikož nelze k uchazečům ukládat některé důležité informace, například výsledky pohovorů, datum a další, které jsou důležité z hlediska testování uchazečů. Bohužel, když jsem procházel internet v průběhu února roku 2016 a hledal projekty a školní práce, které by se věnovaly problému jak získávat data z platformy hackerrank, nebyl jsem v hledání úspěšný. Proto jsem v rámci trainee programu firmy SDE vytvořil první verzi informačního systému Gazelle.

3 Agilní metodika vývoje softwaru

Tato metodika vývoje softwaru popisuje procesy a různé postupy, které se používají pro vývoj software. Jejich cílem je zaručit, aby vývoj softwaru proběhl správně, rychle a kvalitně. Při této metodice se software dodává zákazníkovi v intervalech. V této metodice jsou 4 základní pravidla agilních metod sepsána v manifestu agilního vývoje, které nalezneme v tabulce č. 2.

Tabulka 2: 4 základní pravidla agilních metod (Beck, Kent, et al., 2001)

1.	Jednotlivci a interakce před procesy a nástroji
2.	Fungující software před vyčerpávající dokumentací
3.	Spolupráce se zákazníkem před vyjednáváním o smlouvě
4.	Reagování na změny před dodržováním plánu

Agilní metodiky se dále dělí na 12 principů, které jsou sepsány v tabulce č. 3

Tabulka 3: 12 principů agilních metod (Beck, Kent, et al., 2001)

1.	Naší nejvyšší prioritou je vyhovět zákazníkovi časným a průběžným dodáváním hodnotného softwaru.
2.	Vítáme změny v požadavcích, a to i v pozdějších fázích vývoje. Agilní procesy podporují změny vedoucí ke zvýšení konkurenceschopnosti zákazníka.
3.	Dodáváme fungující software v intervalech týdnů až měsíců, s preferencí kratší periody.
4.	Budujeme projekty kolem motivovaných jednotlivců. Vytváříme jim prostředí, podporujeme jejich potřeby a důvěřujeme, že odvedou dobrou práci.
5.	Nejúčinnějším a nejefektnějším způsobem sdělování informací vývojovému týmu z vnějšku i uvnitř něj je osobní konverzace.
6.	Hlavním měřítkem pokroku je fungující software.
7.	Agilní procesy podporují udržitelný rozvoj. Sponzoři, vývojáři i uživatelé by měli být schopni udržet stálé tempo trvale.
8.	Agilitu zvyšuje neustálá pozornost věnovaná technické výjimečnosti a dobrému designu.
9.	Jednoduchost – umění maximalizovat množství nevykonané práce – je klíčová.
10.	Nejlepší architektury, požadavky a návrhy vzejdou ze samo-organizujících se týmů.
11.	Tým se pravidelně zamýšlí nad tím, jak se stát efektivnějším a následně koriguje a přizpůsobuje své chování a zvyklosti.
12.	Lidé z byznysu a vývoje musí spolupracovat denně po celou dobu projektu.

Z těchto principů můžeme vyčíst, že agilní metody jsou určeny pro lepší komunikaci mezi zákazníkem a vývojáři, a tím i k lepšímu vývoji software. Agilní metodiky se hodí k programování projektů, které nemají jasný cíl a cíle se mohou v průběhu životního cyklu vývoje měnit. Nevýhodou je, že pokud není znám konečný cíl, může se stát, že část, kterou programátor naprogramuje, bude nutné celou změnit, protože se změnil cíl a proto stávající řešení již nevyhovuje, což stojí čas a peníze. Vývoj se proto provádí po menších částech. Každá část se programuje do použitelného stavu, která se následně předvede zákazníkovi. Pokud zákazník není s funkčností nebo uživatelským prostředím spokojen, je kód vrácen programátorům s upřesněním zadání a ti následně dopracují danou část ke spokojenosti zákazníka. Je velmi důležité, aby byl vývojový tým, vlastník projektu nebo alespoň zástupce vlastníka neustále ve spojení. V praxi se totiž dost často stává, že zákazník nespecifikuje svoje požadavky v dostatečné míře a v tomto případě dochází k nedorozuměním. Programátor naprogramuje nějaký kód, o kterém si myslí, že vyhovuje požadavkům. Po kontrole ze strany zákazníka se zjistí, že kód nepracuje podle zákazníkova přání a je potřeba jej přepracovat. To ovšem stojí nemalé úsilí ze strany programátorů a hlavně čas. Takto ztracený čas se však mohl využít na vytváření nové funkcionality. Pokud se tato nedorozumění stávají často, dochází k prodražování projektu a následné možné nelibosti majitele projektu k dalšímu vývoji.

3.1 Scrum

Jedná se o asi nejznámější agilní metodiku k vývoji software. Ve scrumu se rozlišují role, které jsou uvedeny v tabulce č. 4:

Tabulka 4: Základní role scrum metodiky

1.	Product owner
2.	Scrum master
3.	Tým

Product owner

Je osoba, která by měla mít přehled o tom, jak by systém měl vypadat a fungovat. Je zodpovědná za přiřazování priorit a komunikaci mezi jednotlivými účastníky. Je důležité, aby product owner měl vždy představu, jak by jednotlivé části systému měly spolu spolupracovat a popis této funkčnosti musí poté sdělovat vývojovému týmu, aby nedošlo k omylům při vytváření jednotlivých částí systému. Dále je také zodpovědný za přiřazování priorit při vývoji systému. To znamená, že rozhoduje o tom, která část se bude vytvářet v aktuálním nebo následujícím sprintu. Může také odkládat jednotlivé požadavky na další sprinty, pokud se například objeví nějaký problém nebo funkcionalita, která je pro vývoj systému důležitější. Product owner může také psát user stories, kterým se budu věnovat později. Má také na starosti vytvoření, seřazení a údržbu product backlogu.

Scrum master

Jedná se o osobu, která je mezičlánkem mezi týmem a rušivými elementy, které mohou na tým působit zvenčí. Scrum master by měl dohlížet na dodržování principů scrum metodiky. Může však tyto principy i měnit, pokud je to nutné. Měl by řešit problémy, které v týmu mohou nastat a tak udržovat tým produktivní, aby se nemusel zaobírat technickými, mezilidskými nebo jinými problémy. Nemělo by se stát, aby byl scrum master zároveň členem vývojového týmu. V tomto případě může totiž dojít k tomu, že scrum master nerozezná rušivý externí vliv, který může mít na tým negativní dopad a tím pádem dojde ke snížení produktivity celého týmu. Zmíněný product owner však scrum masterem může být.

Tým

Jedná se o skupinu programátorů, kteří pracují na jednotlivých úkolech. Tým je zodpovědný za včasné dodání jednotlivých úkolů. Člen týmu by neměl být zároveň scrum masterem, jak je zmíněno výše.

3.2 User stories

U každého vyvíjeného systému, který používá scrum metodiku by se měl vytvořit product backlog, což je seznam různých funkcionalit (story), chyb a vylepšení, které by měl systém umět. Jako příklad můžeme použít následující story přímo z projektu Gazelle:

User story:

"Jako manažer chci v přehledu vidět uchazeče, kteří už vyplnili test, ale ještě není opraven a mít možnost skočit do testu v HackerRanku, abych jej mohl opravit. Jako náborářka je chci také vidět, abych mohla manažera případně upomenout."

Akceptační kritérium:

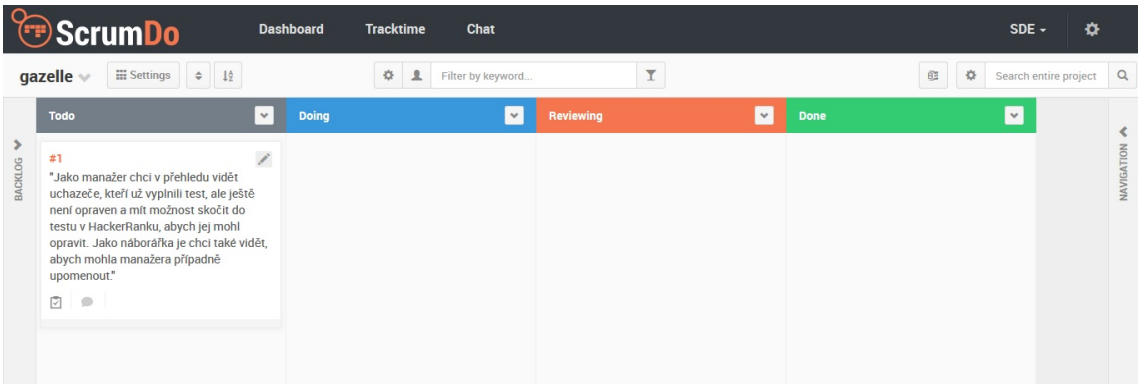
"V přehledu vidím dobu uplynulou od vyplnění testu s tím, že je ideálně barevně odstupňovaná, když je to více než 2 dny. Po kliknutí na uchazeče nebo nějaké tlačítko se otevře test v aplikaci HackerRank. Přehled lze třídit podle testu a také podle doby od vyplnění testu."

Počet story pointů: 6

Na tomto story můžeme názorně vidět, co by chtěl product owner, aby systém uměl. Jednoduše a názorně popisuje vlastnost, která bude implementována. Dále je zde uvedeno akceptační kritérium, ve kterém je napsáno, co daný vypracovaný celek musí umět, aby story bylo přijato. Následně je zde uveden počet story pointů, který značí obtížnost celého story a také značí, za jak dlouho může product owner očekávat dokončení tohoto story. Náročnost jednotlivých story se značí pomocí již zmíněných story pointů. Čím je story těžší, tím více má bodů. Hodnocení náročnosti by měli provádět vždy minimálně 2 vývojáři. Většinou se však na určení náročnosti podílí celý tým, kdy dochází nejdříve k tajnému hlasování a poté, co všichni odhlasují, se

hlasy zobrazí a vybere se hodnota, která má nejvíce hlasů. V případě, kdy se systém teprve začíná vyvíjet, je velmi časté, že obtížnost story nebude správně určena (tj. bude přiřazeno moc nebo naopak málo bodů). To však není na škodu, protože tým se teprve poznává a zjišťuje výkonové možnosti. Většinou se určení hodnoty srovná po dvou až třech sprintech, kdy tým zjistí, kolik toho zvládne za jeden sprint vytvořit.

Backlog je seřazený podle priorit. Programátor si může vzít jakékoliv story ve sprintu, o kterém ví, že by ho mohl splnit. Měl by však postupovat od vrcholu backlogu, kde se nachází nejdůležitější story. Na tvorbu backlogu a co nejsnazšímu přístupu k němu existují specializované stránky. Tyto stránky slouží k uchování a přehlednému zobrazení všech podrobností ke stories. V detailu jednotlivých stories můžeme například nalézt, kdo story vytvořil, kdo na něm pracuje a jaká je náročnost. Z vlastní zkušenosti mohu zmínit http://www.scrumdo.com/ a http://www.pivotaltracker.com.



Obrázek 1: www.scrumdo.com

Ve scrumdo (obrázek č.1) je celá stránka rozdělena na sloupce, kde každý znázorňuje rozpracovanost story. V prvním se nachází nerozpracované story, které si jakýkoliv vývojář může vzít a začít na nich pracovat. Poté se story přesune do sloupce doing, které značí, že na něm byla zahájena práce. Potom, co práci dokončí, se změní status na reviewing. V této fázi se kód testuje, jestli neobsahuje chyby a jestli má požadované vlastnosti. Pokud vše vyhovuje, přesune se story do sloupce done. To značí, že je story hotové, zkontrolované a vyhovuje požadavkům osoby, která story vytvořila. Pokud vypracovaný kód nevyhovuje, je navrácen autorovi a ten jej musí dopracovat. Poté se znovu opakuje celý proces s testováním a ověřováním splnění požadavků.

4 Popis požadavků na systém Gazelle

Než se začal systém Gazelle vyvíjet, došlo nejprve k sepsání požadované funkcionality, kterou product owner požadoval. Tyto požadavky jsme nejdříve zkonzultovali a v pár případech i upravili, jelikož například požadavek se nemohl realizovat z důvodu špatné podpory ze strany hackerranku či z důvodu, že api hackerranku nemělo funkci implementovanou.

Seznam všech požadavků je sepsán zde:

User Story 1: jako manažer i náborářka chci mít jednoduchý přehled všech uchazečů pozvaných na test, abych měl představu o náborovém řízení.

Akceptační kritérium: v přehledu vidím údaje (jméno, email, typ testu, počet dní od pozvání vs počet zbývajících dní do vypršení pozvánky, kolikrát už byl v minulosti pozván, typ testu), jak byly staženy přes API z HackerRanku. Vidím pouze uchazeče, kteří ještě nemají vyplněný a ohodnocený test (viz stavy v Hackerranku). Výstup jde třídit podle testu anebo podle doby od pozvání na test. Pokud někdo psal více testů, řešíme pro teď jen ten poslední.

Story Point: 12

User Story 2: Jako manažer chci v přehledu vidět uchazeče, kteří už vyplnili test, ale ještě není opraven a mít možnost skočit do testu v HackerRanku, abych jej mohl opravit. Jako náborářka je chci také vidět, abych mohla manažera případně upomenout.

Akceptační kritérium: v přehledu vidím dobu uplynulou od vyplnění testu s tím, že je ideálně barevně odstupňovaná, když je to více než 2 dny. Po kliknutí na uchazeče nebo nějaké tlačítko se otevře test v aplikaci HackerRank. Přehled lze třídit podle testu a také podle doby od vyplnění testu.

Story Point: 6

User Story 3: Jako náborářka chci mít možnost zrušit pozvánku na test u některého z kandidátů, například když už nemá zájem pokračovat ve výběrovém řízení.

Akceptační kritérium: Pokud uchazeč nemá zájem pokračovat nebo nemáme zájem my, kliknu na patřičné tlačítko/ikonu a po potvrzení je uchazeč z přehledu smazán. Zároveň je i v HackerRanku zrušeno pozvání na test, resp. je smazán i z HackerRanku.

Story Point: 6

User Story 4: Jako náborářka chci mít možnost pozvat nového uchazeče na patřičný HackerRank test.

Akceptační kritérium: Zadám jméno a email a zvolím test, v HackerRank se přes API patřičný uchazeč vytvoří a vygenerují se přihlašovací údaje pro test a 7 dní na jeho vyplnění. Následně se odešle předdefinovaný email s přihlašovacími údaji, dobou trvání testu a podobně.

Story Point: 10

User Story 5: Jako náborářka chci mít možnost znovu pozvat uchazeče na patřičný HackerRank test, například když mu vypršela pozvánka nebo jej chci pozvat na jiný test.

Akceptační kritérium: Zvolím patřičného uchazeče, kterému chci poslat novou pozvánku, následně zvolím test a on obdrží email stejně jako u prvního pozvání.

Story Point: 6

User Story 6: Jako náborářka chci vidět u opraveného testu, zda byl úspěšný, abych podle toho mohla reagovat.(ats stav qualified) nebo (stav failed).

Story Point: 4

User Story 7: Jako náborářka či manažer chci u opraveného testu mít možnost skočit na jeho výsledky v hackerranku.

Story Point: 4

User Story 8: Jako manažer i náborářka se chci mít možnost přihlásit k Dashboard aplikaci, aby se tam nedostal nikdo nepovolaný.

Story Point: 6

User Story 9: Jako náborářka chci mít možnost uchazeče označit jako vyřešeného, aby už nebyl v přehledu, například když už je vyřazen z výběrového řízení.

Akceptační kritérium: Pro teď třeba ATS stav 17. Takové kandidáty pak dashboard už nezobrazuje, leda bych dal filtrování na tento stav.

Story Point: 5

User Story 10: Jako náborářka chci mít možnost pozastavit výběrové řízení u kandidáta. HackerRank stav ”on hold” třeba použít.

Story Point: 5

User Story 11: Jako náborářka chci mít možnost zobrazit detailní komentář hodnotitele k testu, abych dle něj mohla prioritizovat kandidáty.

Akceptační kritérium: Nový sloupeček, obsah bude např. "QA 3, Algo 2. Možná švindloval". Jelikož asi nejde stáhnout celkové komentáře k testu, můžeme se domluvit, že to bude v komentáři k první otázce, pokud ten stáhnout jde. HackerRank podporuje uvést více komentářů k jedné otázce, poté by se vzal ten poslední k ní.

Story Point: 3

User Story 12: Jako náborářka nebo manažer chci být před provedením nějaké nevratné akce varován, že jsem déle neprovedl obnovení stránky a tedy informace mohou být zastaralé. Např. když byl poslední refresh před více než 30 minutami.

Story Point: 6

User Story 13: Jako náborářka i manažer chci mít možnost ke každému uchazeči doplnit poznámku 1 a poznámku 2.

Akceptační kritérium: Až 300 znaků každá z nich, klidně zobrazit jen začátek a celou až po najetí či kliknutí. Jedna například bude, kdo ho doporučil a druhá jiné poznámky z životopisu, ale použití se může měnit. Editace inline, ať je to uživatelsky efektivní, třeba po dvojitém kliku nebo přes ikonku, která se zobrazí při najetí myší.

Story Point: 5

User Story 14: Při procesu pozvání uchazeče na test chci mít možnost volitelně přidat rovnou obsah poznámky 1 a poznámky 2.

Story Point: 3

User Story 15: Jako náborářka chci mít možnost doplnit ke každéhu uchazeči termín prescreeningu či ústního pohovoru (samostatné sloupečky).

Akceptační kritérium: Například tlačítkem "pozvat na prescreening" a tlačítkem "pozvat na pohovor", kdy následně zadám datum. Datum pak lze inline opravovat (např. po dvojkliku nebo přes ikonku zobrazenou při najetí myší). Tím se zároveň změní patřičně i stav kandidáta ("prescreening resp. ústní pohovor). Tyto stavy není už třeba mapovat na hackerrank, ty jsou dobré jen pro psaní testu.

Story Point: 5

User Story 16: Jako náborářka chci mít možnost změnit stav na ”po prescreeningu”, ”po pohovoru”, ”zamítnout”, ”najatý”, ”šel jinam” (navíc k už existujícím stavům píše test, qualified, failed, pozastavený a vyřešený) není třeba mapovat na hackerrank stavy, ty jsou dobré jen pro fázi psaní testu. Samozřejmě dle stavu chci mít možnost filtrovat. Také by bylo dobré se domluvit, jaký filtr bude aplikován defaultně, ať se nezobrazují všichni kandidáti

Story Point: 6

User Story 17: Jako náborářka chci mít možnost upravit šablonu upomínkového emailu (stejný pro všechny).

Akceptační kritérium: Někde na dashboardu je odkaz, který otevře popup s editací šablony pro upomínkový email a umožní upravit jeho znění. Jde o textový email bez obrázků či formátování a s makry pro přihlašovací údaje a podobně (viz výše).

Story Point: 8

User Story 18: Jako náborářka chci vidět, kolikrát už jsem poslala kandidátovi upomínku k testu, kdy poprvé a kdy naposled.

Story Point: 6

---

5 Použité technologie

5.1 Haml

Haml je značkovací jazyk, který je primárně určen pro kombinaci s jazykem Ruby a má za úkol jednoduše popsat jazyk HTML bez použití ”inline code”. Pod zkratkou ”inline code” si můžeme představit kód, který je napsán přímo do těla např. funkce, místo toho aby byl napsán v separátní metodě. Jazyk Haml nemá párové tagy. Vše se píše pouze pomocí jedné značky a atributů, které píšeme do složených závorek přímo za tag.

```
%html
  %head
    %title Titulek stránky
  %body
    %h1 Paragraf č. 1
    %div{:id => "container"}
      %p text paragrafu
```

Jak lze vidět na tomto příkladu, Haml nepotřebuje ukončovací tagy, čímž nemusíme psát tolik kódu a kód samotný je velmi přehledný. Je však nutné dodržovat správné odsazení jednotlivých tagů. Tj. když například chceme zanořit p tag do divu, tak p tag musí být na novém řádku a odsazení od začátku řádku musí být větší, než má předchozí tag. Je nutné dodržovat stejné odsazování napříč všemi stránkami.

Základními principy jazyka Haml jsou:

Tabulka 5: Základní principy jazyka Haml (Haml About, 2015)

1.	Markup Should be Beautiful
2.	Markup Should be DRY
3.	Markup Should be Well-Indented
4.	HTML Structure Should be Clear

5.2 Javascript

Javascript je interpretovaný programovací jazyk, který lze používat jak na straně klienta, tak i na straně serveru, kdy se používá Node.JS. Javascript se velmi často používá pro zajištění funkčnosti webových stránek.

5.3 jQuery

jQuery je javascriptová knihovna, která rozšiřuje funkce javascriptu. jQuery poskytuje funkce na manipulaci s HTML DOM (Document Object Model), jako například animace, skrývání prvků, ajaxový asynchronní požadavek a mnoho dalších.

V projektu Gazelle je velká část manipulace s html prvky prováděna právě díky knihovně jQuery. Také jsem použil právě jQuery ajax, u kterého je díky této knihovně velmi přehledný a intuitivní zápis:

```
$.ajax({
        method: 'GET',
        url: '/get_user',
        data: {
            idUser: 1
        },
        success: function(content, textStatus, jqXHR){
            console.log("Receive: " + content);
        },
        error: function(jqXHR, textStatus, errorThrown){
            console.log('Error: ' + textStatus + errorThrown);
        }
})
```

V následující metodě je několik parametrů, které jsou nutné k úspěšnému provedení požadavku (jQuery.ajax(), 2016) :

method - zde se určuje o jaký typ požadavku jde (GET, PUT, POST,...)

url - zde se uvádí, na jakou URL má být požadavek odeslán

data - tento parametr je nepovinný a lze ho použít pokud chceme k požadavku přibalit nějaká data. Tato data můžeme například nejdříve převést do formátu JSON a teprve poté odeslat.

success - kód v této funkci se provede, jen když bude požadavek úspěšný. Je důležité zmínit, že ajaxové volání je asynchroní. To znamená, že skript nebude čekat na výsledek ajaxového požadavku a bude pokračovat dále ve vykonávání kódu, který následuje za ajax požadavkem. Proto pokud potřebujeme pracovat s daty, které nám server pomocí požadavku odešle, je nutné vložit funkci, která bude pracovat s daty, právě do funkce success.

error - kód v této funkci se provede jen, když nebude požadavek úspěšný. Z parametrů v této funkci se můžeme dozvědět, proč byl požadavek neúspěšný.

5.4 Ruby

Ruby je dynamický open source objektový programovací jazyk. Ruby je známé tím, že jeho kód je velmi čitelný a čtivý. Lze to ukázat na následujícím případě:

```
3.times { puts "Hello World!" }
```

Tento příkaz lze doslovně přeložit následovně: třikrát vypiš Hello World!. Ruby je základní programovací jazyk, na kterém staví framework sinatra, který popisuji dále.

5.5 Framework Sinatra

Sinatra je doménově specifický framework, pomocí něhož můžeme velmi lehce definovat takzvané routy. Routa je http metoda, s URL - matching pattern. To znamená, že si můžeme nadefinovat routu s libovolným názvem, která vykoná nějakou funkcionalitu po zavolání (Getting Started, 2016). Ukázat si to můžeme na jednoduchém příkladu, kdy po zavolání se vypíše pozdrav Hello world!.

```
require 'sinatra'

get '/greetings' do
  puts 'Hello world!'
end
```

Volat tuto routu můžeme několika způsoby např. pomocí ajaxu, nebo pomocí URL, kterou zadáme do prohlížeče:

```
http://www.nameOfDomain.cz/greetings
```

Získání dat z požadavku na straně serveru je velmi jednoduché a provádí se pomocí:

```
data = JSON.parse(request.body.read, :symbolize_names => true)
```

V tomto případě máme veškerá data uložená v proměnné data. Z této proměnné můžeme data využít následujícím způsobem:

```
idOdUser = data[:idOfUser]
```

Popřípadě je nemusíme ukládat do proměnné, ale rovnou s nimi můžeme pracovat a dosazovat do příkazů:

```
person = Person.get(data[:idOfUser])
```

Tento příkaz provede pomocí DataMapperu dotaz do databáze a nalezne všechny záznamy, které mají id stejné jako je hodnota v data[:idOfUser].

Musíme však znát přesnou strukturu dat, kterou jsme si na server poslali, tj. všechny identifikátory, struktury, pole a podle toho se k datům chovat.

V sinatře se také používají regulární výrazy v routách. To je výhodné například, když si potřebujeme předat nějaký parametr přes URL, ale předem nevíme, jak bude parametr vypadat.

```
get '/greetings/*' do |name|
  "Hello #{name}!"
end
```

5.6 DataMapper

DataMapper je objektově relační datamapper. Pod tímto názvem si můžeme představit konverzi mezi objektově orientovaným jazykem a relační databází (Why Da-

taMapper?, 2014). Velkou výhodou tohoto přístupu je, že programátor nemusí psát dotazy v konkrétním databázovém jazyce, ale píše kód v objektovém dotazovacím jazyce. Tímto je dosaženo toho, že datamapper není závislý na jednom databázovém jazyce a můžeme použít databázi podle našich požadavků. Můžeme tedy přepínat mezi SQlite, MySQL, PostgreSQL pouze pomocí jednoho příkazu a tím pádem nemusíme měnit dotazy do databáze napříč programem.

Příklad změny databázového systému z SQlite do PostgreSQL:

```
DataMapper.setup(:default, 'sqlite:///path/to/project/database.db')
```

Tento příkaz se používal, než se například rozhodlo přejít na nový databázový systém. Pro přechod se pouze změní parametry v příkazu DataMapper.setup.

```
DataMapper.setup(:default, 'postgres://user:password@hostname/database')
```

Další velkou výhodou je, že pokud si předem vytvoříme modely databáze, tak datamapper automaticky vytvoří i nové tabulky v již novém jazyce ihned po startu serveru. Proto migrace na nový databázový systém zabere mnohem méně času, než kdybychom ručně měnili databázový systém a vytvářeli tabulky. Bohužel DataMapper nedokáže přesunout i data z tabulek. Pro tuto akci je nutné použít jiný program.

Jednotlivé řádky znamenají jednotlivé kolonky, které se vytvoří v databázi. Např. řádek:

```
property :name,    String,  :required => true, :length => 50
```

značí, že se v databázi vytvoří kolonka s názvem name, která bude mít typ string o maximální velikosti 50 znaků a zároveň nesmí být NULL. Těmito a mnoha dalšími příkazy můžeme vytvořit rozsáhlé databáze.

Model databáze může mít následující strukturu:

```
class User
  include DataMapper::Resource

  property :id,      Serial
  property :name,    String,  :required => true, :length => 50
  property :surname, String,  :required => true, :length => 50

  has n, :address

end
```

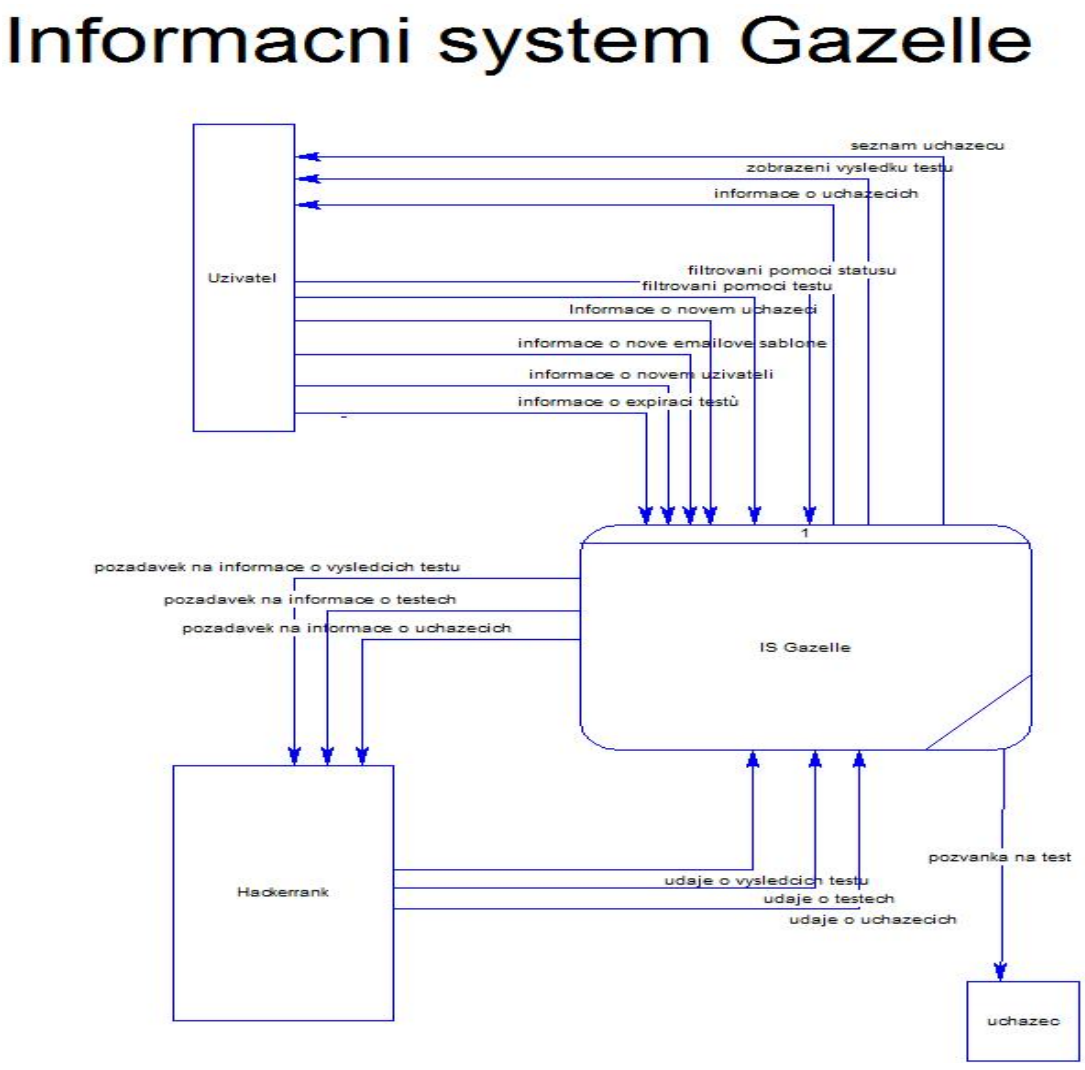
```
class Address
  include DataMapper::Resource

  property :id,     Serial
  property :street, String,  :required => true, :length => 50
  property :number, Integer, :required => true
  property :city,   String,  :required => true, :length => 50

  belongs_to :user
end
```

Na tomto příkladu můžeme vidět jednoduchou relaci mezi třídou User a Address. Příkaz has n a belongs to značí, že user může mít více adres. Pomocí příkazu Serial se v databázi vytvoří u kolonky id sekvence, která každý vložený záznam inkrementuje o 1. Pomocí příkazu default nastavíme výchozí hodnotu, pokud není žádná vložena.

6 Diagramy systému

6.1 Data-flow diagram

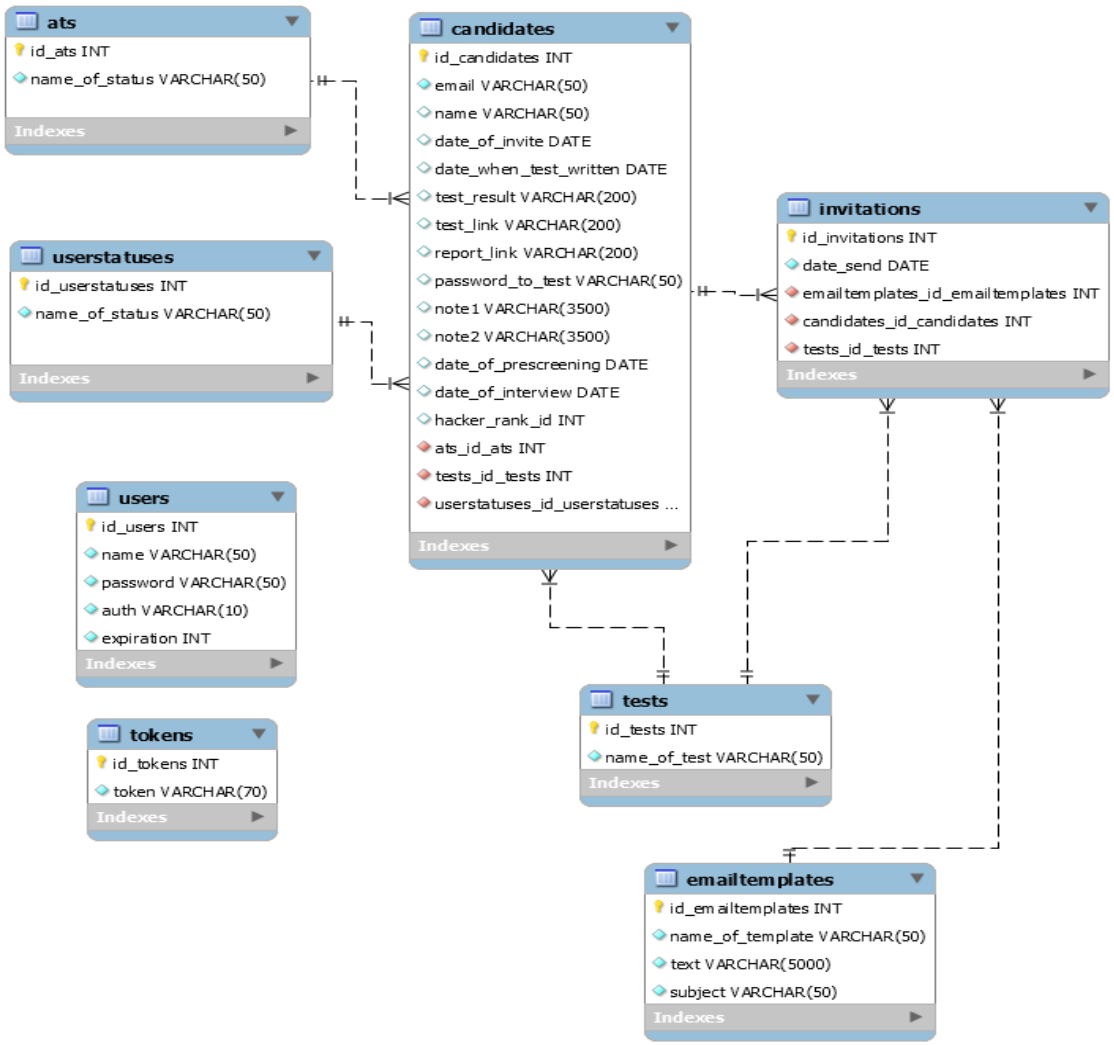
Tento Data-flow diagram znázorňuje procesy systému Gazelle



Obrázek 2: Data-flow diagram procesu informačního systému Gazelle

6.2 Entitně relační diagram databáze

Entitně relační diagram pro databázi můžeme nalézt na obrázku č. 3. Celá databáze je navržena tak, aby byla jednoduchá. Každý typ statusu má svou vlastní tabulku. Tyto statusy poté využívá tabulka candidates



Obrázek 3: Entitně relační model databáze informačního systému Gazelle

- Tabulka ats obsahuje seznam všech ats statusů.
- Tabulka candidates obsahuje záznamy pro kandidáty.
- Tabulka userstatuses obsahuje seznam všech user statusů.
- Tabulka invitations obsahuje seznam všech pozvánek, které byly uživatelům poslány ať už přes systém Gazelle nebo hackerrank.

- Tabulka tests obsahuje seznam všech testů, které manažer vytvořil v systému hackerrank.
- Tabulka emailtemplates obsahuje seznam všech šablon pro zvací/upomínací emaily.
- Tabulka tokens obsahuje seznam všech tokenů, které slouží pro přístup k datům hackerranku.
- Tabulka users obsahuje seznam všech uživatelů, kteří se mohou přihlásit do systému Gazelle.

7 Implementace systému

7.1 1. verze bez podpory backendu

Vývoj Gazelle se rozdělil do dvou částí. První část byla bez backend podpory, která umožňovala pouze zobrazit výsledky, filtrování a řazení. Pro ukládání například informací o termínu prescreeningu nebo termínu pohovoru se stále muselo chodit na stránky hackerranku nebo do Google dokumentů. Druhá část se vyvíjela již s podporou backendu. Tím se velmi usnadnil vývoj, jelikož v první části vývoje velmi často docházelo k problémům s CORSem (Cross-origin resource sharing (Enable cross-origin resource sharing, 2016)) a dobou, kterou trvalo stahování dat. K problému s CORSem docházelo z důvodu, že api server hackerranku neposílal hlavičku access allow origin. Ta je vyžadována pro bezproblémové požadavky mezi prohlížečem a serverem z jiné domény. Ovšem s nástupem backendu se již dotazy mezi prohlížečem a serverem provádí v rámci jedné domény. Tím odpadají problémy s chybějící hlavičkou, kterou servery hackerranku neposílají. V první fázi vývoje jsem tento problém vyřešil pomocí jednoduchého PHP skriptu, který přeposílal požadavky prohlížeče na server hackerranku a poté vracel odpověď.

Nevýhodou tohoto prvního řešení byla rychlost načtení. Načtení samotných testů a uchazečů bez vyhodnocení trvalo ještě přijatelných 5 sekund. Problém nastával, pokud uživatel chtěl zjistit, jakých výsledků uchazeči dosáhli. Zde již čas načítání dat trval přibližně 100 sekund, což je velmi dlouho pro normální používání. Celý proces zde pro osvětlení problému popíši:

1) Musíme stáhnout všechny testy. Testy získáme tak, že provedeme ajaxový požadavek na následující URL:

```
www.hackerrank.com/x/api/v2/tests?limit=-1&access_token={token}
```

Do této URL je nutné doplnit přístupový token, díky kterému nám server odpoví daty v JSON formátu. Testy tedy dostaneme ve struktuře, která obsahuje pole ve kterém jsou jednotlivé testy. Toto pole se musí celé projít. Z každé struktury testu je nutné získat id testu.

2) Toto id se použije pro další požadavek, který načte uchazeče pro každý test zvlášť. URL pro tento požadavek je:

```
www.hackerrank.com/x/api/v2/tests/{test_id}
/candidates/?limit=-1&access_token={token}
```

Zde dochází k prvnímu problému. Neexistuje URL, která by nám vrátila úplně všechny uchazeče, které jsme pozvali na testy. Dochází zde tudíž k posílání více požadavků než je nutné, což prodlužuje dobu načítání dat. K získání uchazečů je tedy nutné v prvním kroku vyparsovat jednotlivá id testů. Ty se použijí pro jednotlivé požadavky, které nám vrátí pro každý test uchazeče. Až do tohoto bodu trvalo načítání dat, parsování a zobrazení maximálně do 5 sekund.

3) Zde nastává zlom v rychlosti načítání požadavků. Pokud chceme načíst i jednotlivé komentáře k uchazečům, musíme projít všechny strukturu s daty uchazečů,

které jsme získali z předchozího bodu. V těchto datech musíme zjistit, jestli uchazeč má přiřazené ID a ats status od systému hackerranku. Pokud ho nemá, nemá žádný komentář. Pokud id a ats status v datech nalezneme, značí to, že uchazeč komentář mít může (samozřejmě ale mít také nemusí). To zjistíme, pokud provedeme požadavek na následující URL:

```
www.hackerrank.com/x/api/v2/tests/{test_id}
/candidates/{idUchazeče}?access_token={token}
```

Požadavek na tuto URL s uvedeným id testu a uchazeče nám vrátí podrobné informace o uchazeči. Z těchto dat již lze vyparsovat komentář k testu. S product ownerem jsme se dohodli, že jediný komentář, na který bude brán zřetel, je u první otázky a vždy poslední komentář. Bohužel je to trochu krkolomné řešení. Platforma hackerrank poskytuje speciální textové pole na konečnou poznámku, ovšem nelze ji žádným způsobem získat, jelikož ji server nepřidá k datům. Proto se komentáře píšou k první otázce, která je obsažená v přijatých datech.

V tomto kroku je tedy načítání všech dat kompletní a je možné s nimi dále pracovat. Bohužel toto načítání celkově trvá okolo 100 sekund, což je téměř nepoužitelné. Proto jsem vymyslel, že by bylo možné provádět na začátku pouze načtení uživatelů bez komentáře a komentáře poté na pozadí stahovat a průběžně pomocí JQuery zobrazovat. Bohužel tento postup má jednu velkou nevýhodu. Prohlížeč ajaxové požadavky vyřizuje postupně. Tím vznikala nepříjemnost, že pokud se načítaly komentáře a uživatel systému chtěl např. nastavit nějaký ats status přes systém, tak prohlížeč zařadil tento požadavek až na poslední místo a uchazeč stejně musel čekat, než se všechna data načetla.

Dalším nápadem bylo, že prvotní data se načtou bez komentářů a pouze pokud uživatel bude chtít, tak spustí stahovaní komentářů ručně. Bohužel zde byl zase problém, že v průběhu načítání se nedaly dělat žádné jiné ajaxové požadavky. Bylo však nutné tento problém vyřešit, protože kolegyně z personálního oddělení musely vidět i komentáře k testu, jelikož díky nim poté kontaktovaly uchazeče s tím, že buď uspěl, nebo neuspěl.

7.2 2. verze s podporou backendu

Problém s pomalým načítáním dat se podařilo částečně vyřešit s příchodem backendu, který velmi usnadnil načítání a ukládání dat. V této fázi již bylo možné ukládat různé poznámky, datum prescreeningu a pohovoru do databáze. Dále již bylo možné si zavést svoje vlastní nové statusy, které se částečně mapují na staré ats statusy. A hlavně načítání všech dat z databáze již trvá maximálně 3 sekundy. Což je oproti předchozím 100 sekundám velký rozdíl. Bylo toho docíleno tím, že již neprobíhá stahování dat mezi prohlížečem a serverem hackerranku, ale mezi serverem Gazelle a serverem hackerranku. Z tohoto důvodu může server Gazelle dodat prohlížeči všechny informace téměř okamžitě.

Stále je zde však problém, že synchronizace mezi servery trvá dlouho. Tento problém však lze minimalizovat tím, že nastavíme úlohu do CRONu, který jen zavolá určitou URl adresu a server Gazelle si již stáhne a uloží všechna data automaticky. Tím lze zajistit, že synchronizace databází proběhne například v noci, kdy se systém nepoužívá.

Při synchronizaci databází vyvstalo mnoho problémů, jak ji vůbec řešit. Prvním problémem bylo, že data, která získám z hackerranku, nemají žádné časové razítko, kdy byla data naposledy upravena. Z tohoto důvodu nelze žádným způsobem zjistit, jestli se data nějak změnila a je nutná synchronizace. Musel jsem tedy vymyslet skript, který projde veškerá data z databáze Gazelle a porovná je s daty z hackerranku. Postupným redukováním všech dat, která se musí aktualizovat, jsem došel k názoru, že aktualizace se bude týkat pouze těchto atributů: odkaz na test, odkaz na opravení testu, ats status, user status, datum kdy byl test napsán a id uživatele v databázi hackerranku. Ostatní atributy nejsou potřeba aktualizovat nebo například jsou již známé od pozvání uživatele a v průběhu celého cyklu se nijak nemění.

Menší výjimka nastává u ats statusu. Zde se provádí aktualizace pouze pokud je ats status menší než 5. Je to z důvodu, jelikož firemní statusy se částečně mapují na ats statusy. Přesněji mapují se pouze první 4 statusy:

Tabulka 6: seznam ats statusů, které se mapují na firemní statusy

0	ATS State Not Set == Test zatím nebyl napsán
1	Test Completed - Evaluation Reqd == Opravit test
2	Test Completed - Qualified == Opraveno - Qualified
3	Test Completed - Failed == Opraveno - Failed

A v případě, že uchazeč má nastavený vyšší firemní status než ats status, docházelo by k tomu, že již nastavený firemní status by se přemazal ats statusem, což by vedlo ke špatný výsledkům.

Velkým problémem je také to, když se zadá špatný email při uchazečově pozvání. Jedná se o tři důvody, kvůli kterým nelze provést změnu. První je ten, že nelze nijak změnit email přes api hackerranku. Druhým je ten, že test je vygenerovaný právě na email zadaný při pozvání. Tím pádem, pokud je email špatný a uchazeč zadá správný email, systém hackerranku nedovolí uchazeči napsat test z důvodu, že nesouhlasí email. Třetím problémem je to, že i kdyby se email měnil jen lokálně na serveru Gazelle, tak by problém vznikl hned při první synchronizaci databáze. Data se totiž porovnávají pomocí emailu uchazeče a id testu, na který je uchazeč pozván. A z toho důvodu, pokud se neshoduje email a zároveň id testu, tak skript vytvoří nového uživatele.

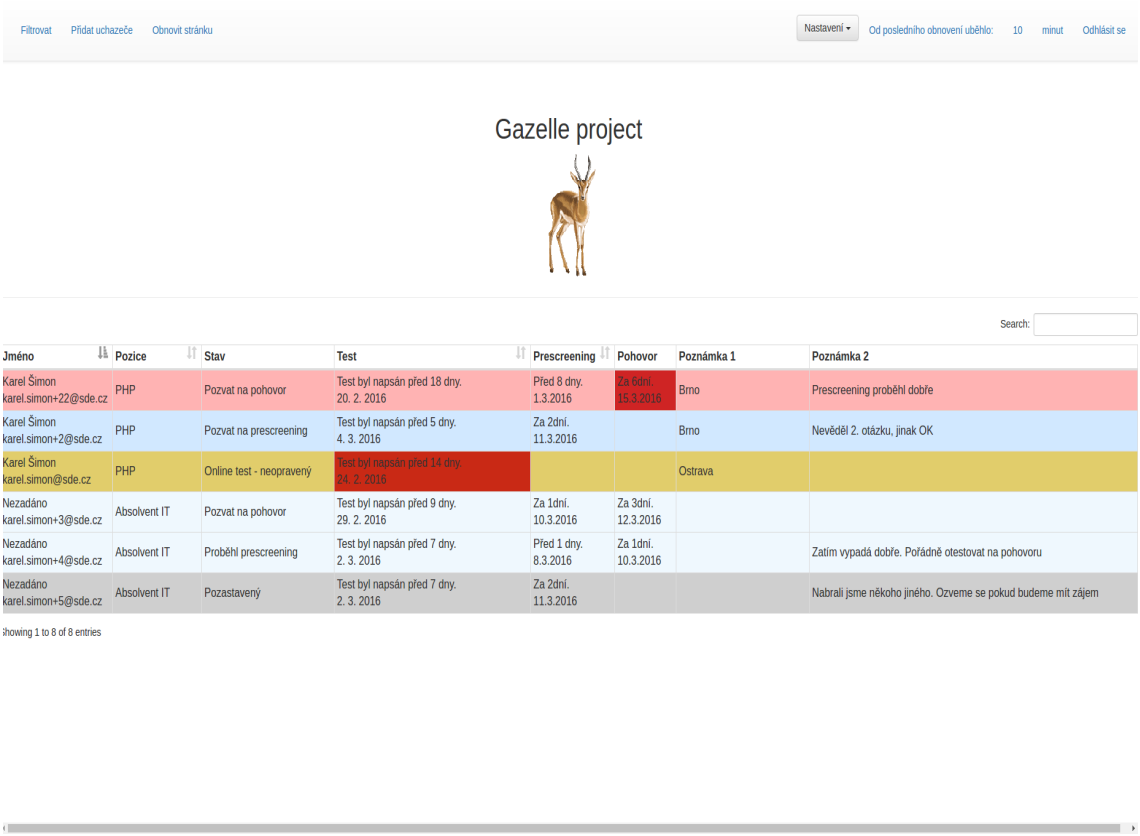
8 Popis systému

8.1 Přihlášení do systému

Systém Gazelle je provozován na vnitřních serverech firmy SDE a je dostupný pouze z intranetu. Tím je zamezeno neoprávněnému přístupu nějaké třetí osoby z internetu. Uživatel, který na stránku přijde poprvé, je ihned přesměrován na přihlašovací stránku (Obrázek č.4), kde musí zadat uživatelské jméno a heslo.

Obrázek 4: Přihlašovací stránka (zdroj: systém Gazelle)

Pokud nemá přihlašovací údaje, nemůže si je samovolně vygenerovat. Tím je zabráněno, aby kdokoliv s přístupem do intranetu si mohl prohlížet data o uchazečích. Vytvořit nové přihlašovací údaje může jen již vytvořený uživatel s administrátorskými právy. Po zadání platných přihlašovacích údajů je uživatel přesměrován na hlavní stránku (Obrázek č.5).



Obrázek 5: Hlavní strana (zdroj: systém Gazelle)

8.2 Hlavní stránka

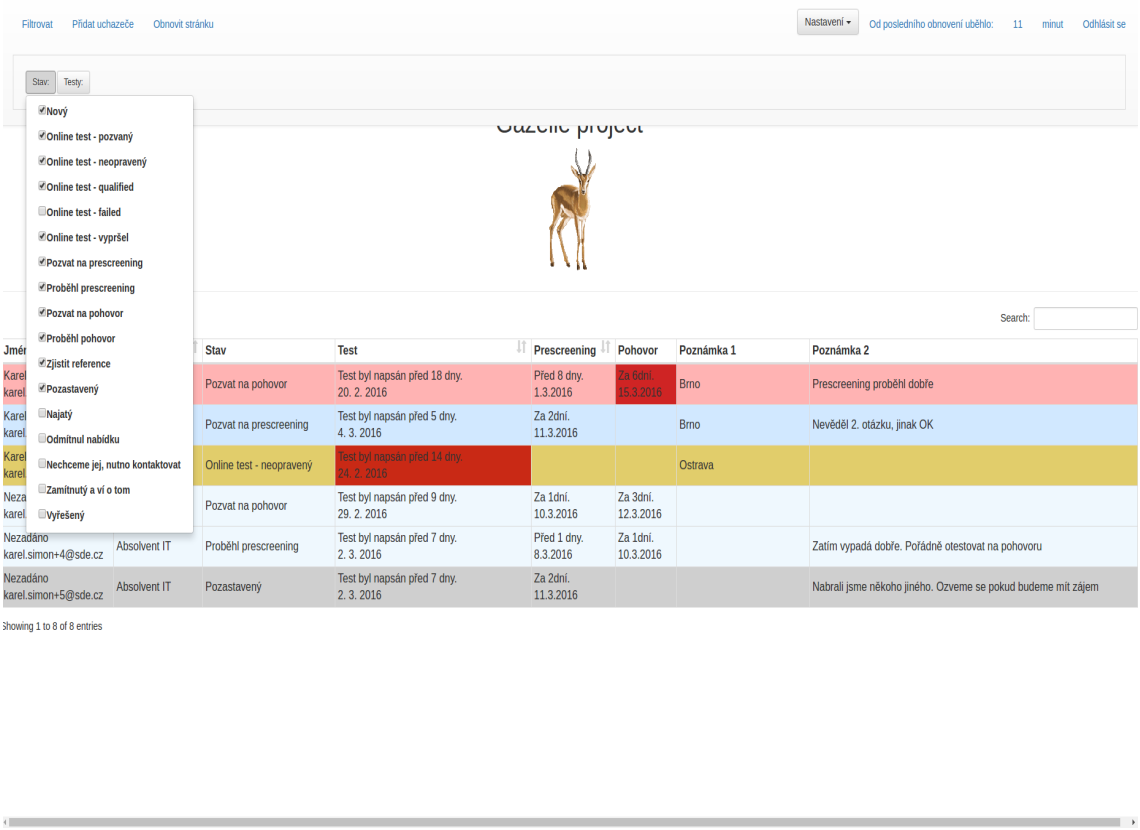
Na hlavní stránce uživatel uvidí navigační lištu, která obsahuje akce, které může provést a logo celého projektu. Dále zde uvidí tabulku se všemi kandidáty, u kterých nějakým způsobem probíhá náborové řízení.

8.3 Navigační lišta

Navigační lišta je rozdělena do několika funkčních celků - filtrování uchazečů, přidání nového uchazeče, obnovení stránky se zachováním filtrů, nastavení systému. Navigační lišta se pohybuje spolu s obrazovkou, je to z důvodu, že pokud například budeme chtít změnit filtr, když budeme procházet uchazeče, tak není nutné se vracet úplně nahoru, ale stačí jen přejet ukazatelem myši nahoru na navigaci. Také jsem implementoval funkci, která pokud uživatel použije scroll až pod úroveň nadpisů tabulky, tak tyto nadpisy se budou pohybovat spolu s navigační lištou.

8.3.1 Filtry

První tlačítko s nápisem filtrovat otevře podnabídku (Obrázek č.6), kde si lze vybrat jaké statusy a testy se zobrazí. Filtr jsem naprogramoval tak, aby si pamatoval více zaškrtnutých voleb. Tím pádem je možné zadat, že chceme vidět pouze uchazeče, kteří mají například status opraveno - failed, opraveno - qualified a zároveň jsou pozvaní/psali vybrané testy. Defaultně jsou vybrané jen některé statusy a všechny testy. Toto je z důvodu, že již není potřeba zobrazovat uchazeče, u kterých již výběrové řízení skončilo. Samozřejmě si lze zapnout filtr, který tyto uchazeče zobrazí.

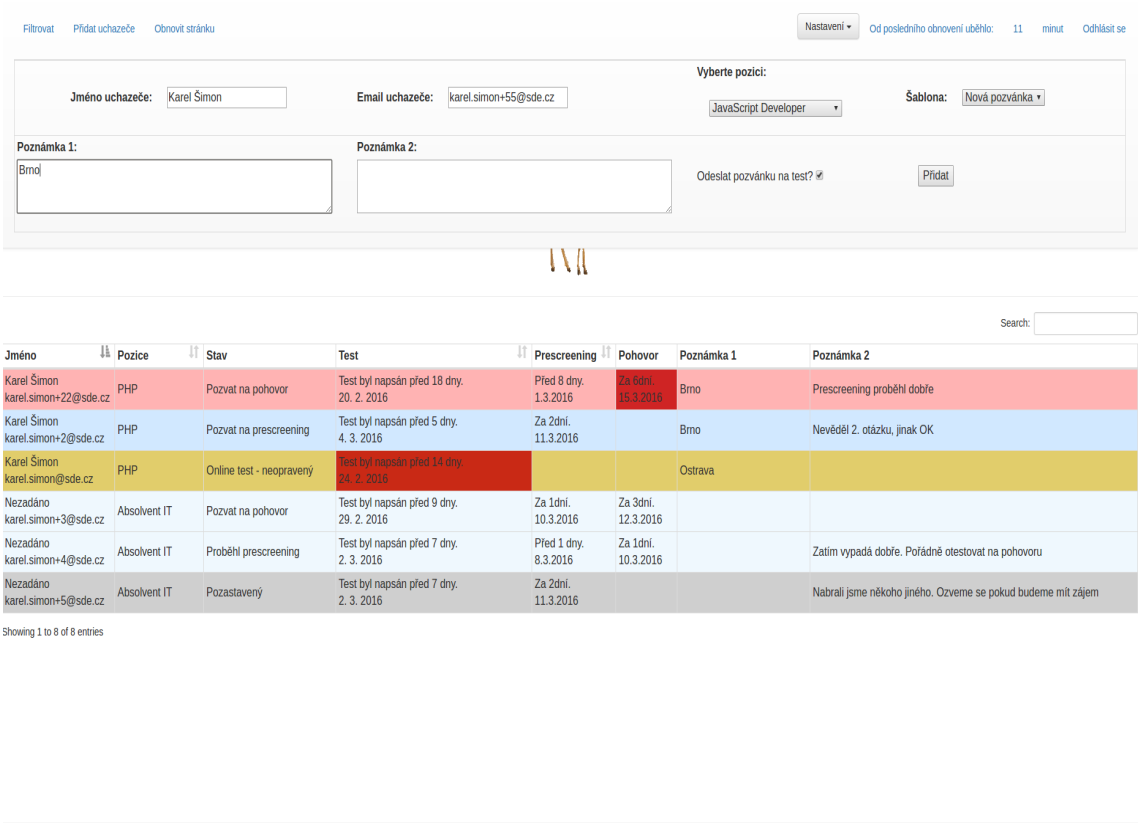


Obrázek 6: Filtrování (zdroj: systém Gazelle)

8.3.2 Přidání uchazeče

Druhé tlačítko slouží k přidání uchazeče do systému a případně i k pozvání na test (Obrázek č.7). Po kliknutí se objeví podnabídka, která obsahuje několik atributů, které lze vyplnit. Jedná se o jméno, email, test na který chceme uživatele pozvat,

check box, pomocí kterého nastavíme, jestli se má uchazeči vygenerovat test v hackerranku a zároveň vygenerovat zvací email a vyplnit poznámka 1 a poznámka 2. Pokud uživatel zaškrtne check box, že se má uchazeči vygenerovat test v hackerranku a zároveň odeslat zvací email, tak se nejprve provede požadavek na server Gazelle. Server si z dat, které uživatel poslal vybere email a id testu a provede požadavek na hackerrank. Ten test vytvoří a následně odpoví s údaji o testu. Tato data obsahují odkaz na test, email, heslo k testu a ještě několik informací. Tato data si server Gazelle převezme a s daty, která přišla od klienta, je uloží. Také se vygeneruje záznam v tabulce s pozvánkami, aby bylo možné zjistit, kdy byl uživatel pozván na test. Dále server pošle odpověď na požadavek klienta s odkazem na test a heslem k testu. Tato data se použijí pro vygenerování emailu, do kterých se vloží průvodní dotaz s informacemi o testu, odkazem na test a datem, do kdy se má test vyplnit.



Obrázek 7: Vytvoření uchazeče v systému (zdroj: systém Gazelle)

Pokud však uživatel check box na vytvoření testu nezaškrtne, nevygeneruje se žádný test a ani se nevygeneruje žádný email. Je to z důvodu, pokud se hledá například senior programátor, tak není nutné ho testovat na elementární znalosti, jenž se testují ve vytvořených testech. Rovnou se s ním postupuje do dalšího kola

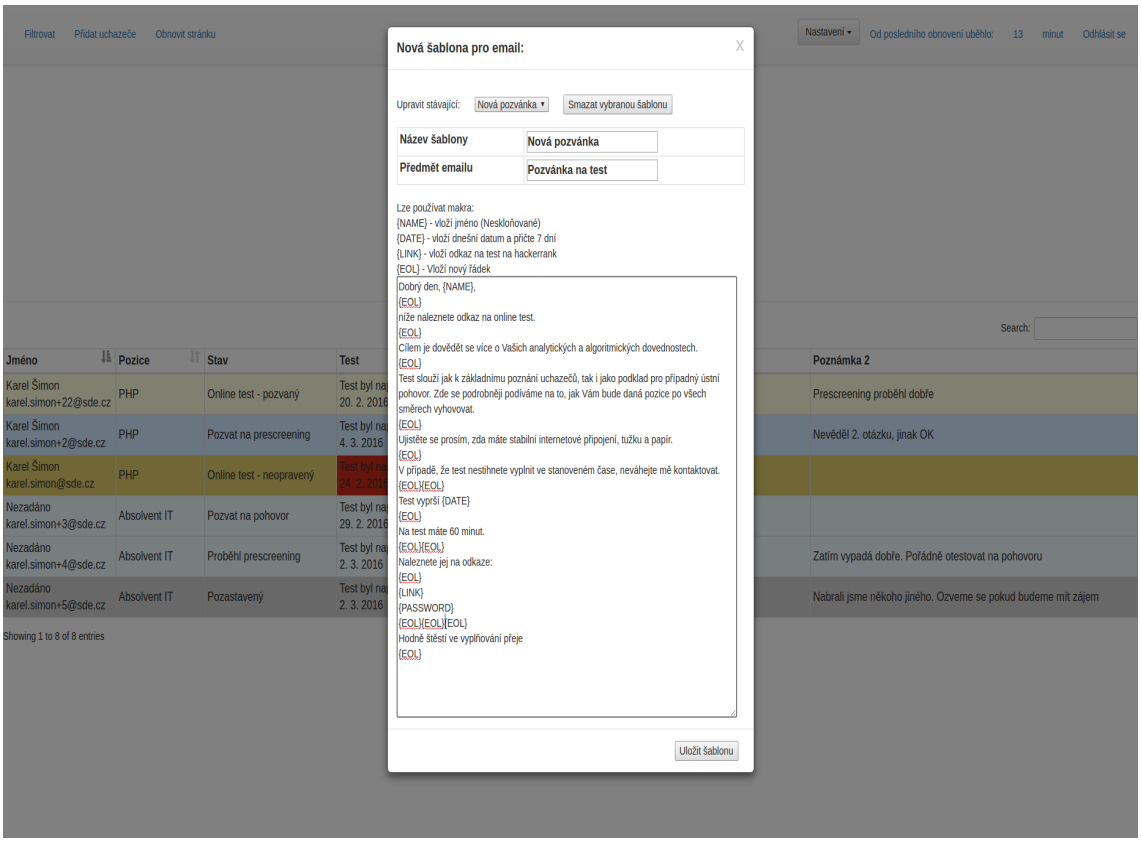
pohovorů. Poznámka 1 a poznámka 2 slouží k tomu, aby se například rovnou napsalo, že uchazeč se hlásí na pobočku firmy v Brně nebo Ostravě nebo případně, že se již jednou hlásil, ale neprošel.

8.3.3 Obnovení stránky

Třetí tlačítko obnovit slouží k rychlému načtení dat z databáze. Má stejnou funkci jako obnovení v prohlížeči, avšak s tou výhodou, že pokud si uživatel nastavil nějaké filtry, tak o ně nepřijde a data se automaticky po obnovení znovu vyfiltrují.

8.3.4 Nastavení systému

Čtvrté tlačítko slouží k nastavení základních věcí, jako například vložení nové šablony pro emailovou pozvánku, vytvoření nového uživatele, nastavení doby expirace, obnovit databázi s a bez komentářů nebo změna hesla k účtu.



Obrázek 8: Emailová šablona (zdroj: systém Gazelle)

Emailové šablony

Vložení nové šablony pro emailovou pozvánku slouží k přidání, smazání, úpravě

emailové pozvánky (Obrázek č.8). Tuto funkci jsem naprogramoval tak, aby pozvánka podporovala makra. Podporovaná makra jsou:

`{EOL}`

- Místo této značky se vloží nový řádek

`{PASSWORD}`

- Místo této značky se vloží heslo k testu. Test však nemusí mít žádné heslo, proto pokud skript žádné heslo nenajde, tak se místo této značky vloží prázdný string

`{LINK}`

- Místo této značky se vloží odkaz na test

`{DATE}`

- Místo této značky se vloží datum, které značí, do kdy může uchazeč test vyplnit

`{NAME}`

- Místo této značky se vloží jméno uchazeče

Jelikož je posílání emailu řešeno přes mailto protokol, je nutné ho naformátovat, aby email vypadal dobře. Z tohoto důvodu jsem musel přidat i zmíněné makro EOL, které přidá speciální znak nového řádku. Výhoda tohoto řešení je, že uživatel, který bude odesílat pozvánku, si může ve svém emailovém klientovi celý email zkontrolovat a případně upravit email, aby odpovídal jeho požadavkům.

Změna hesla

Tlačítko změna hesla slouží ke změně přihlašovacích údajů do systému.

Aktualizace dat v databázi

Tlačítko obnovit databázi se dělí na dvě části. První udělá to, že spustí synchronizaci databáze Gazelle s databází hackerranku bez stažení komentářů. Tato synchronizace je dobrá, pokud potřebujeme rychle stáhnout nová data například kvůli zjištění, jestli nějaký uchazeč již napsal test a je tedy nutné ho opravit. Tato synchronizace trvá přibližně 30 sekund. Je to z důvodu, že se musí projít všechny záznamy z hackerranku a z Gazelle a ty následně vůči sobě porovnat. Stejným procesem musí projít i vytváření záznamů o pozvánkách.

Druhou možností je obnovit databázi s výsledky. Tato volba je časově velmi zdlouhavá z důvodů, které jsem zmínil již dříve a měla by se provádět jen, když je to nutné. Případně, když se systémem nikdo nepracuje.

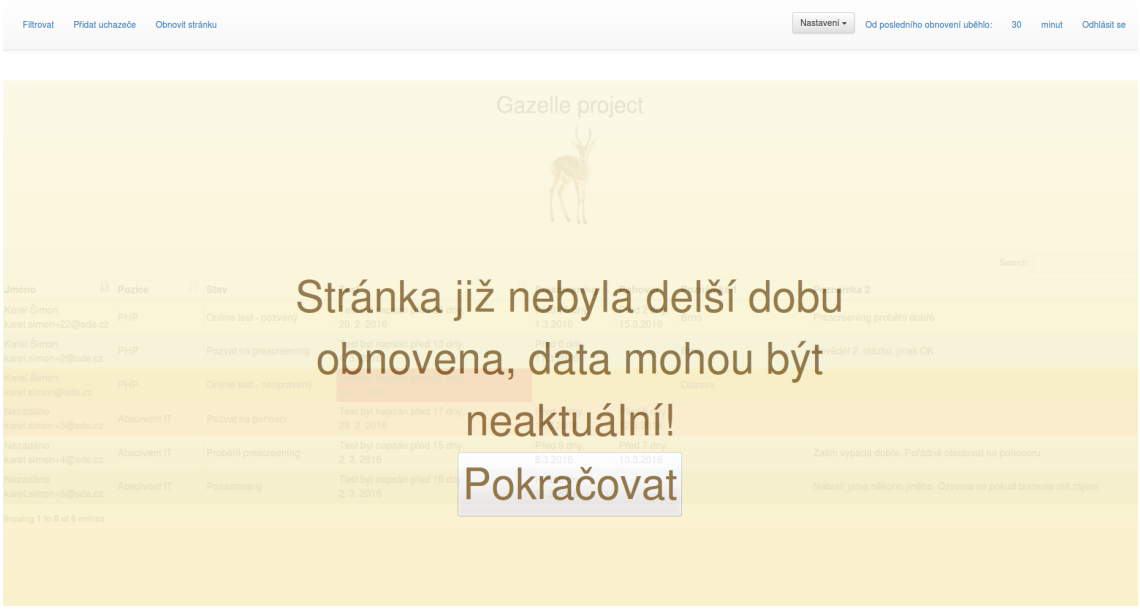
Nastavení expirace

Tlačítko nastavit délku expirace slouží k nastavení vnitřní proměnné, která určuje, za kolik dní zobrazení uchazeči změní barvu na červenou, aby se lépe vidělo, na kterého kandidáta se například zapomnělo a je nutné ho rychle vyřešit.

Přidání nového uživatele

Pomocí tlačítka přidat nového uživatele lze přidat nové uživatele systému. Tím, že se nelze jiným způsobem zaregistrovat, se zaručilo, že data uvidí pouze osoba, která

může data vidět. Přidávat nové uživatele smí pouze uživatel, který má status admin. Pokud má status user, vůbec se mu toto tlačítko nezobrazí.



Obrázek 9: Varování po 30 minutách (zdroj: systém Gazelle)

Informace o datech

Další částí navigační lišty je ukazatel, jak dlouho nedošlo k obnovení stránky a dat. Je to z důvodu, že pokud na systému pracuje více uživatelů naráz, data se velmi rychle můžou stát neaktuálními a tím si uživatelé budou přemazávat svoje data, která do systému vložili. Bohužel není nijak vyřešen problém s přepisováním dat různými uživateli. Tato funkcionalita by byla náročná na implementaci. Odpočet má také implementovanou funkci, že pokud se 30 minut neprovedlo obnovení stránky, přes celou obrazovku se zobrazí varování (Obrázek č.9), které řekne, že data mohou být zastaralá. Toto varování lze však ignorovat, zavřít a dále pokračovat v práci. Pokud se však stránka ani za dalších 30 minut neobnoví, varování se znovu zobrazí. Takto se bude zobrazovat každých 30 minut.

8.4 Tabulka s kandidáty

Tabulka má následující pole: Jméno a email, pozice, stav, test, prescreening, pohovor, poznámka 1, poznámka 2. Téměř každé pole v tabulce je klikací a lze na něm provést nějakou akci.

8.4.1 Informace o uchazeči

V poli jméno můžeme vidět jméno uchazeče a jeho email. Pokud uživatel špatně zadá jméno při pozvání uchazeče, lze ho po kliknutí na jméno změnit. Bohužel nelze

měnit email z důvodů, které jsem uvedl dříve.

8.4.2 Pozice na kterou se uchazeč hlásí

V poli pozice je vidět na jakou pozici se uchazeč hlásí.

8.4.3 Stav uchazeče

V poli stav je uveden stav ve kterém se uchazeč nachází. Uchazeč vždy začíná buď na statusu Nový, nebo Online test - pozvaný. Status Nový získá pokud uchazeči nebyl vygenerovaný online test, čímž se značí, že se rovnou přechází na další kola pohovoru. Je zde také implementována funkce, která mění barvy řádků podle toho, jaký má uživatel nastavený status. Proto uživatel systému pokud zná tyto barvy, může okamžitě rozpoznat, v jakém stavu se uchazeč nachází. Uchazeč může mít následující statusy:

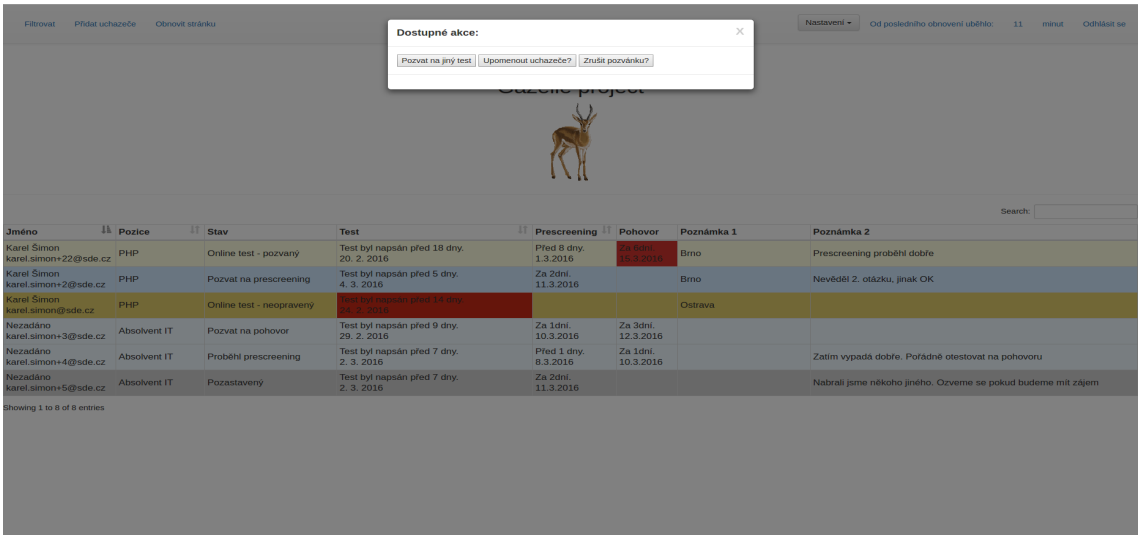
Tabulka 7: seznam firemních statusů

-1	Nový
0	Online test - pozvaný
1	Online test - neopravený
2	Online test - qualified
3	Online test - failed
4	Online test - vypršel
5	Pozvat na prescreening
6	Proběhl prescreening
7	Pozvat na pohovor
8	Proběhl pohovor
9	Zjistit reference
10	Pozastavený
11	Najatý
12	Odmítnul nabídku
13	Nechceme jej, nutno kontaktovat
14	Zamítnutý a ví o tom
17	Vyřešený

Samozřejmě lze filtrovat uchazeče na všechny tyto statusy. Po kliknutí na toto pole se otevře seznam se všemi stavy a lze ho libovolně měnit.

8.4.4 Informace o testu

V poli test je možné vidět, kdy byl uchazeč pozván na test, kdy test napsal a kolikrát byl upomenut. Je zde implementovaná funkce na zvýraznění celého pole, pokud uživatel nenapsal test do 7 dnů. Tím lze velmi rychle zjistit, že uživatel asi zapomněl na test a je možné mu hned odeslat upomínku.



Obrázek 10: Dostupné akce pro uživatele (zdroj: systém Gazelle)

Po kliknutí na pole se otevře modální okno (Obrázek č.10), které nabízí další funkčnost jako například: upomenout uchazeče, kdy se vygeneruje nový email s upomínkovou šablonou, do které se pro jistotu opět vloží odkaz na test, datum do kterého má uchazeč test vypracovat. Další funkcí je pozvat na nový test, tímto se vygeneruje nový uchazeč, nový test a nová pozvánka. Lze zde také nalézt odkaz na test, který uchazeč vyplnil. Po kliknutí na toto tlačítko se otevře v prohlížeči okno, do kterého se načte napsaný test v hackerranku. Každý status má svou vlastní funkcionalitu. Je to z důvodu, že například u pozvaného uchazeče, který ještě nepsal test, nelze otevřít napsaný test, jelikož žádný nemá.

8.4.5 Datum prescreeningu

V poli prescreening se nastavuje datum prescreeeningu. V případě, že uchazeč napíše test a uživatel systému nenastaví datum prescreeningu, tak podle nastavené hodnoty délky expirace se po této hodnotě políčko obarví do červené barvy, aby se tento uživatel zvýraznil. Pokud se datum nastaví, červená barva zmizí. Lze také datum z tohoto pole smazat, pokud například uchazeč odmítl nabídku.

8.4.6 Datum pohovoru

V poli pohovor lze nastavit datum pohovoru. Funguje stejně jako pole prescreening. Tj. je zde funkce na zvýraznění, pokud již prescreening proběhl a není nastaveno datum pohovoru. Také se zde zvýrazňuje, pokud pohovor již proběhl a uživatel má stále status pozvat na pohovor.

8.4.7 Poznámky

Pole poznámka 1 a poznámka 2 mají stejný význam a slouží pouze k tomu, aby si uživatel mohl zapisovat vlastní zápisky k uchazeči.

9 Vyhodnocení požadavků na systém Gazelle

Všechny požadavky jsem se snažil naprogramovat tak, jak byly zadány, což se u většiny požadavků podařilo. Ovšem u některých toto nedovolovala funkčnost platformy hackerrank a proto byly po konzultaci s produkt ownerem upraveny tak, aby se daly realizovat.

Naprogramované user story 1, 2, 6, 9, 10, 11, 13, 14, 15, 16, 18 lze vidět na obrázku č. 5.

Story č. 3, 5, 7 lze vidět na obrázku č. 10.

Story č. 12 lze vidět na obrázku č. 9

Story č. 4, 14 lze vidět na obrázku č. 7

Story č. 8 lze vidět na obrázku č. 4

Story č. 17 lze vidět na obrázku č. 8

10 Závěr

V rámci této bakalářské práce jsem vytvořil systém Gazelle pro podporu výběru nových uchazečů o zaměstnání, který sjednotil část dosud používaných technologií. I nadále však zůstává povinnost používat platformu hackerrank, která ale poskytuje minimální funkčnost přes API rozhraní a také špatně implementuje funkce pro vyřízení požadavků na získání dat. Naštěstí se již stále více přemýšlí o nahrazení platformy hackerrank vlastním řešením, čímž by se ušetřila spousta peněz za poplatky hackerranku.

Nový modul, který by nahradil hackerrank, by samozřejmě bylo možné implementovat do stávajícího systému Gazelle. Je nutné však zmínit, že tento systém byl původně brán jako "proof of concept". A jelikož systém Gazelle svůj účel splnil, rozhodlo se o vývoji nové verze V2, která je již funkční a nahradila původní verzi. Proto by vývoj tohoto modulu pro mou verzi systému byl zbytečný, jelikož nová verze používá jiné technologie a propojování staré a nové verze by mohlo přinést mnoho problémů.

Pokud bych zhodnotil vývoj systému po ekonomické stránce, s největší pravděpodobností by se v krátkodobém období nevyplatil. Je nutné brát na vědomí, že jsem na systému pracoval v rámci trainee programu společnosti SDE a vývojem systému jsem získával nové zkušenosti. Proto mi spousta věcí trvala mnohem déle než obvykle a zkušenému programátorovi by vývoj zabral mnohem méně času. Myslím si však, že desítky až stovky hodin, které jsem při vývoji strávil hledáním a učením se nových věcí, rozhodně nebyla zbytečná investice.

V době psaní této bakalářské práce byl již systém přibližně asi čtvrt roku v provozu, a proto jsem mohl opravit chyby, které zamezovaly z různých důvodů ve funkčnosti systému. V březnu roku 2016 jsem kontaktoval kolegyni z personálního oddělení SDE s žádostí, zda by mi mohla sdělit míru její spokojenosti se systémem a případné připomínky. Její zpětná vazba obsahovala následující výhody a nevýhody systému Gazelle z pohledu osoby, která ho denně používá. **Výhody:**

-Systém byl určen k tomu, aby se snížila frekvence návštěv serveru hackerrank, což se podařilo, jelikož v současné době se již jejich systém skoro nemusí navštěvovat.

-Je ihned vidět, jakého výsledku uchazeč dosáhl.

-Je ihned vidět, kdy byl uchazeč pozván na test a kdy test expiruje.

-Velkou výhodou je, že pozvánky na test se mohou otevírat v poštovním klientu a posílat z firemního emailu, čímž se velmi zredukoval problém, že pozvánka zapadne ve spamové složce, což se velmi často dříve stávalo.

-Je ihned vidět, pokud se uchazeč hlásí podruhé. Systém toto dokáže rozpoznat.

-Vše je na jednom místě

Nevýhody:

-Chyby v systému:

-Občas se neupdatovala data mezi databázemi.

-Občas se nevygeneroval dobře zvací email s odkazem testu.

Dále se mi dostalo věcné zpětné vazby k funkčnosti systému a díky tomu jsem mohl případně upravit funkčnost jednotlivých funkcí pro lepší použitelnost systému. Dalším věcným přínosem pro mou budoucí kariéru byla možnost vyzkoušet si vývoj systému od úplného začátku a komunikace s product ownerem a uživateli, kteří systém používají.

11 Reference

About Us. HackerRank. [online]. 2015 [cit. 17.1.2016]. Dostupné z: https://www.hackerrank.com/aboutus.

Candidate Status in API Results - HackerRank for Work User Documentation - Confluence. Atlassian. [tabulka]. 18.11.2015 [cit. 20.12.2015]. Dostupné z: https://hackerrank.atlassian.net/wiki/display/HFWUD/Candidate+Status+in+API+Results.

Enable cross-origin resource sharing. Enable cross-origin resource sharing. [online]. 15.1.2016 [cit. 18.1.2016]. Dostupné z: http://enable-cors.org/.

Fulton, Hal *Ruby – kompendium znalostí pro začátečníky i profesionály*. Brno, Zoner Press, 2003, s. 768, ISBN 978-80-7413-018-2.

Getting Started. Sinatra. [online]. 9.3.2016 [cit. 17.3.2016]. Dostupné z: http://www.sinatrarb.com/intro.html.

HackerRank V2 API. I/O Docs. [online]. 2015 [cit. 17.1.2016]. Dostupné z: http://apidocs.hackerrank.com/hr4wv2.

Haml About. Haml. [online]. 2015 [cit. 17.1.2016]. Dostupné z: http://haml.info/about.html.

Havens, Andrew *Beginners guide to creating a REST API*. Andrew Havens Ruby Developer. [online]. 13.9.2008 [cit. 17.1.2016]. Dostupné z: http://www.andrewhavens.com/posts/20/beginners-guide-to-creating-a-rest-api/.

Introducing JSON. JSON. [online]. 2016 [cit. 17.1.2016]. Dostupné z: http://www.json.org/.

jQuery.ajax(). jQuery write less, do more. [online]. 17.3.2016 [cit. 17.3.2016]. Dostupné z: http://api.jquery.com/jQuery.ajax/.

Manifest Agilního vývoje software, 2016. *Manifest Agilního vývoje software*. agilemanifesto. [online]. 2016 [cit. 8.2.2016]. Dostupné z: http://agilemanifesto.org/iso/cs/

Manifest Agilního vývoje software, 2016. *Manifest Agilního vývoje software*. agilemanifesto. [online]. 2016 [cit. 8.2.2016]. Dostupné z: http://agilemanifesto.org/iso/cs/principles.html

Momjian, Bruce *PostgreSQL - Praktický průvodce*. Brno, Computer Press, 2003, s. 402, ISBN 80-7226-954-2.

S námi to zvládnete. Teamio. [online]. 2016 [cit. 17.1.2016]. Dostupné z: http://www.teamio.com/cz/.

Strickland, Jonathan *Introduction to How Google Docs Works - How Google Docs Works | HowStuffWorks*. Howstuffworks Tech. [online]. 2.6.2008 [cit. 17.1.2016]. Dostupné z: http://computer.howstuffworks.com/internet/basics/google-docs.htm.

Why DataMapper?. DataMapper. [online]. 17.3.2014 [cit. 17.3.2016]. Dostupné z: http://datamapper.org/why.html.

Zakas, Nicholas *JavaScript pro webové vývojáře*. Brno, Computer Press, 2009, s. 832, ISBN 978-80-251-2509-0.

Seznam obrázků

Seznam tabulek

Přílohy

A Zdrojový kód

Zdrojový kód se nachází na přiloženém CD.