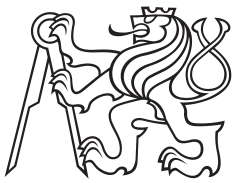
Diplomová práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Systém pro správu lidských zdrojů

Bc. Šimon Kohout

Květen 2016
Vedoucí práce: Mgr. Artur Linhart

České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra počítačů

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Bc. Šimon Kohout**

Studijní program: Otevřená informatika
Obor: Softwarové inženýrství

Název tématu: **Systém pro správu lidských zdrojů**

Pokyny pro vypracování:

Analyzujte problematiku správy lidských zdrojů v prostředí malých až středně velkých podniků. Seznamte se a porovnejte existující aplikace sloužící ke správě lidských zdrojů s důrazem na náborový proces nových zaměstnanců.
Na základě provedené analýzy proveďte a řádně zdokumentujte návrh webové aplikace, vhodně řešící problematiku správy lidských zdrojů a pokrývající proces náboru a selekce nových zaměstnanců.
Dle vytvořeného návrhu aplikace následně proveďte její implementaci a výslednou aplikaci otestujte. Součástí testování aplikace musí být testování jednotkové, integrační, zátěžové a zároveň i testování uživatelského rozhraní.
Systém implementujte jako webovou aplikaci v jazyku JAVA a k implementaci klientské části aplikace využijte javascriptový framework Ext JS.

Seznam odborné literatury:

1. BAUER, Christian, Gavin KING a Christian BAUER. Java persistence with Hibernate. Rev. ed. London: Pearson Education [distributor], 2007, xxxiv, 841 p. ISBN 9781932394887.
2. KOUBEK, Josef. Personální práce v malých a středních firmách. 4., aktualiz. a dopl. vyd.Praha: Grada, 2011, 281 s. Management (Grada). ISBN 978-80-247-3823-9.

Vedoucí: Mgr. Artur Linhart

Platnost zadání: do konce letního semestru 2016/2017

prof. Dr. Michal Pěchouček, MSc.
vedoucí katedry

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta elektrotechnická

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 17. 2. 2016

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 27. 5. 2016

.....

Abstrakt / Abstract

Cílem této diplomové práce je provést analýzu správy lidských zdrojů. Na základě výsledků této analýzy je navržena a implementována aplikace využívající JAVA EE a framework Ext JS.

The goal of this thesis is to analyse issues of human resource management. Based on the results of this analysis a application is designed and implemented using JAVA EE and Ext JS framework.

Obsah /

1 Úvod 1
2 O projektu 2
2.1 Baader Computer 2
3 Řízení lidských zdrojů 3
3.1 Personální činnosti 3
3.1.1 Vytvoření a analýza pracovních míst 3
3.1.2 Personální plánování 4
3.1.3 Získávání, výběr a následné přijímání pracovníků 4
3.1.4 Hodnocení pracovníků 4
3.1.5 Rozmísťování pracovníků a ukončování pracovního poměru 4
3.1.6 Odměňování pracovníků 4
3.1.7 Vzdělávání a rozvoj pracovníků 4
3.1.8 Pracovní vztahy 4
3.1.9 Péče o pracovníky 4
3.1.10 Personální informační systém 4
3.2 Získávání pracovníků 5
3.2.1 Kompetence 5
3.2.2 Definice pracovního místa 5
3.2.3 Tvorba pracovní nabídky 5
3.2.4 Tým pro výběr zaměstnance 6
3.2.5 Forma výběru 6
3.2.6 Výběr pracovníků 6
3.2.7 Pohovor 7
3.2.8 Závěr výběrového řízení 7
4 Alternativní aplikace 8
4.1 Mark 8
4.2 Teamio 9
5 Analýza 11
5.1 Rozsah práce 11
5.1.1 Rozdělení projektu 12
5.2 Katalog požadavků 12
5.2.1 Funkční požadavky 13
5.2.2 Nefunkční požadavky 14
5.3 Konkretizace požadavků 15
5.3.1 Kandidáti 15
5.3.2 Zdroje kandidátů 15
5.3.3 Dovednosti 16
5.3.4 Standard společnosti Baader Computer 16
5.4 Negativní vymezení 16
5.5 Uživatelské role 17
5.5.1 HR manažer 17
5.5.2 HR asistent 17
5.5.3 Hodnotitel 17
5.5.4 Vedoucí týmu 17
5.5.5 Managing Director 18
5.6 Proces výběrového řízení 18
5.6.1 Stavy výběrového řízení 18
5.6.2 Rozhodnout kandidáta 19
5.6.3 Hodnocení kandidáta 19
5.7 Model případu užití 19
5.7.1 Kandidáti 19
5.7.2 Výběrová řízení 20
5.7.3 Dovednosti 21
5.7.4 Hodnocení 22
6 Návrh 24
6.1 Zvolené technologie 24
6.1.1 Spring 24
6.1.2 Hibernate 24
6.1.3 Ext JS 24
6.1.4 Maven 25
6.1.5 Nástroje BC 25
6.2 Databázový model 26
6.2.1 Systém 26
6.2.2 Kandidáti 26
6.2.3 Zdroje kandidátů 27
6.2.4 Dovednosti 27
6.2.5 Výběrová řízení 27
6.3 Grafický návrh 29
6.3.1 Témata frameworku Ext JS 30
6.3.2 Uživatelské rozhraní 32
6.3.3 Rozložení obrazovky 32
6.3.4 Detail kandidáta 33
6.3.5 Hodnocení dovednosti 34
6.3.6 Výběrové řízení 35
6.3.7 Fáze výběrového řízení 36
6.3.8 Hodnocení kandidáta 36
6.3.9 Rozhodnout kandidáta 37
7 Implementace 39
7.1 Použité nástroje 39
7.2 Vývojové prostředí 39
7.3 Architektura 40

7.4 Prezentační vrstva 40
7.4.1 Architektura 41
7.4.2 Panely 41
7.4.3 Vlastí komponenty 41
7.5 Aplikační vrstva 41
7.5.1 Zabezpečení 42
7.5.2 Uživatelská oprávnění ... 43
7.6 Datová vrstva 44
8 Testování 45
8.1 Průběžné testování 45
8.2 Testovací prostředí 45
8.3 Integrační a jednotkové testy .. 45
8.3.1 Použitá technologie 46
8.3.2 Architektura testů 46
8.4 Kontinuální integrace 46
8.5 Zátěžové testování 47
8.6 Testování uživatelského rozhraní 48
8.6.1 Testovací metoda 48
8.6.2 Testovací scénáře 49
8.6.3 Výsledky testování 50
9 Budoucnost projektu 53
9.1 Nejbližší rozšíření 53
9.2 Další rozšíření 53
9.2.1 Modul zaměstnanců 53
9.2.2 Modul mentorský 54
9.3 Uplatnění projektu 54
10 Závěr 55
Literatura 56
A Seznam použitých zkratek 57
B Uživatelské příručka 58
C Instalační manuál 59
C.1 Instalace databáze 59
C.2 Instalace aplikace 59
D Databázové schéma 60
E Obsah přiloženého CD 61

Tabulky / Obrázky

8.1. Test vytvoření výběrového řízení 51
8.2. Test vytvoření kandidáta 51
8.3. Test ohodnocení kandidáta 52
8.4. Test rozhodnout kandidáta 52

2.1. Baader Computer 2
4.1. Aplikace Mark 8
4.2. Aplikace Teamio 10
5.1. Spirálový model 11
5.2. Stavový model výběrového řízení 18
5.3. Model případů užití - kandidáti 20
5.4. Model případů užití - výběrová řízení 21
5.5. Model případů užití - dovednosti 22
5.6. Model případů užití - hodnocení 23
7.1. Vícevrstvý model 40
7.2. Komponenta našeptávače 42
7.3. Vyhodnocení uživatelských práv 43
7.4. Dialog hodnocení kandidátových dovedností 44
8.1. Test pro smazání kandidáta ... 46
8.3. Zátěžové testování 48

Kapitola 1
Úvod

Na začátku mého magisterského studia se mi naskytla příležitost podílet se na trainee programu společnosti Baader Computer. V jeho úvodu mi pak byl představena myšlenka celé aplikace HR Cockpit, zabývající se správou lidských zdrojů a následně mi byla nabídnuta spolupráce při jejím vývoji v rámci mé budoucí diplomové práce.

Díky tomuto zadání jsem pak tedy měl příležitost zrealizovat kompletní vývoj této aplikace, jehož popis je v této práci obsažen. Součástí této práce tedy byla důkladná analýza problematiky lidských zdrojů a následný návrh vhodného řešení a jeho pozdější realizace.

Z hlediska nezbytné analýzy se to ukázalo jako velmi unikátní příležitost pro zmapování celkové problematiky lidských zdrojů a společně s pomocí pracovníků HR oddělení společnosti Baader Computer, identifikování všech procesů jejich oddělení a definování požadavků na aplikaci, která by jim při vykonávání těchto procesů pomáhala.

Hned v úvodu těchto analýz bylo zjištěno, že celková problematika HR oddělení středně velké firmy se jeví jako příliš rozsáhlá pro okamžitý vývoj v rámci studentské práce. Byla proto stanovena idea budoucí modulární aplikace, pokrývající kompletní potřeby pracovníků HR. Cílovým modulem takovéto aplikace, jejímuž dalšímu vývoji se pak dále tato práce věnuje, je tzv. modul kandidátů, na který by jednou v budoucnu měly navazovat další rozšiřující moduly.

Kapitola 2
O projektu

Za vznikem aplikace HR Cockpit, jejímž vývojem se celá tato práce zabývá, jsou dva hlavní důvody.

První a tím více očividná je skutečnost, že personální oddělení společnosti Baader Computer[1] skutečně potřebuje nový systém plně podporující všechny jeho procesy a činnosti.

Druhou motivací byl pak vznik studentského trainee programu, organizovaného touto společností. V plánu bylo, aby v rámci tohoto programu byli studenti zapojeni po období zhruba jednoho roku do celkového chodu společnosti. Za tímto účelem se tak společnost zapojila i do mentoringového programu ČVUT[2].

Jako ideální řešení se ukázalo možnost spojit potřebu vzniku nové aplikace a právě tohoto trainee programu. Vývoj aplikace od naprostého prvopočátku mohl nabídnout potenciálnímu účastníkovi toho programu unikátní příležitost pracovat na kompletním procesu reálné aplikace se skutečným koncovým uživatelem. Zároveň vývoj interní aplikace stále nabízel určitou volnost při samotném vývoji, který produktivní projekt těžko sám o sobě měl.

Samozřejmě obsahem celého programu nemohl být jen vývoj interního systému. V rámci toho programu též byla příležitost nastavit některé základní parametry projektu a v samotném vývoji pokračovat v rámci právě této studentské práce.

2.1 Baader Computer

Profil společnosti říká, že se jedná o,"...Baader Computer spol. s r.o. je česká společnost se zahraniční účastí s 20letou tradicí na českém i evropském trhu. Specializuje se na vývoj software, navrhování, konzultace a implementaci informačních systémů, poradenství v oblasti informačních technologií, Business Intelligence, CRM systémy a ERP systémy. Své zákazníky má společnost především v oblasti automotive, chemickém průmyslu, mezi výrobci plastů a dodává individuální řešení i do dalších segmentů"[1].

Obrázek 2.1. Baader Computer [1]

[1] www.bcpraha.com
[2] https://mentoring.cvut.cz

Kapitola 3
Řízení lidských zdrojů

Personalistika nebo také často používané pojmy jako personální práce, personální administrativa, personální řízení či řízení lidských zdrojů - jak upozorňuje profesor Koubek - tyto termíny jsou často v praxi ale i odborné literatuře často zaměňovány. Jejich rozdíl pak definuje takto: „termíny personální práce či personalistika se používají spíše pro nejobecnější označení této činnosti, zatímco termíny personální administrativa, personální řízení a řízení lidských zdrojů se používají spíše k charakterizování vývojové úrovně a koncepce personální práce“.[2]

Aktuálně nejvíce používaným termínem v rámci této oblasti je pak tedy řízení lidských zdrojů. Vyznačuje se tím, že klade zvýšený důraz na strategický aspekt personální práce. V závislosti na cíle celé firmy formuje dlouhodobé komplexní cíle personální práce. Kvůli tomu je nezbytné zabývat se i externími vlivy na formování a fungování pracovní síly společnosti. V rámci řízení lidských zdrojů je tak nutné sledovat vlivy trhu práce, ekonomické podmínky společnosti a uplatnění na trhu, vývoj technologií a hodnotové priority lidí a mnoho dalšího. Dalším typickým rysem řízení lidských zdrojů je, že čím dál tím větší část odpovědnosti a práce je postupně přesouvána na vedoucí pracovníky všech úrovní. Koubek dále soudí, že existuje výrazná vazba mezi spokojeností pracovníka a jeho výkonem a věrností firmě, kvůli čemuž se firmy podstatně více věnují kvalitě pracovního života zaměstnanců.

3.1 Personální činnosti

Jak už bylo naznačeno, tak v rámci řízení lidských zdrojů existuje celá řada potenciálních úkolů k jejich plnění - takto Koubek definuje tyto základní činnosti.[2]

- Vytvoření a analýza pracovních míst
- Personální plánování
- Získávání, výběr a následné přijímání pracovníků
- Hodnocení pracovníků
- Rozmísťování pracovníků a ukončování pracovního poměru
- Odměňování pracovníků
- Vzdělávání a rozvoj pracovníků
- Pracovní vztahy
- Péče o pracovníky
- Personální informační systém

3.1.1 Vytvoření a analýza pracovních míst

Definování potřebných pracovních úkolů, jejích odpovídajících odpovědností a nutných předpokladů na pracovníka, který by byl schopen je vykonávat. Na základě skupiny potřebných vykonávaných úkolů je nutné definovat samotnou pracovní pozici s jí přiřazenými úkoly a nutnou sadou kompetencí pracovníka.

3.1.2 Personální plánování

Plánování budoucích potřeb firmy na pracovníky a jejího pokrytí či plánování rozvoje zaměstnanců.

3.1.3 Získávání, výběr a následné přijímání pracovníků

Úkolem této činnosti je přílákání dostatečného množství kvalitních kandidátů na požadované pracovní pozice. Následně je nutné vybrat z nich nejlepšího možného kandidáta na požadovanou pozici.

3.1.4 Hodnocení pracovníků

Personální činnosti, jejímž cílem je zjistit, jak kvalitně vykonává pracovník své povinnosti. Součástí této úlohy je pak i v rámci procesu hodnocení konzultovat výkony přímo s dotyčným pracovníkem, aby bylo možné pochválit jeho výkony a případně se dohodnout na budoucí nápravě možných nedostatků.

3.1.5 Rozmísťování pracovníků a ukončování pracovního poměru

Proces umísťování pracovníků na požadované pozice. Do této kategorie spadá změna pracovní pozice, povyšování, odchod na důchod, ale i případně propuštění zaměstnance.

3.1.6 Odměňování pracovníků

Systém hmotného a nehmotného hodnocení, sloužícího k motivaci zaměstnanců. Jeho součástí jsou tak i kromě finančního ohodnocení i různé finanční benefity.

3.1.7 Vzdělávání a rozvoj pracovníků

Proces vzdělávání zaměstnanců. Jeho součástí je identifikování potřeby vzdělávání jednotlivých pracovníků a vytvoření plánu vzdělávání. Nutností pak je i pro další plánování potřeba vyhodnocovat jednotlivé probíhající či již ukončené vzdělávací programy.

3.1.8 Pracovní vztahy

Nejčastěji organizování jednání mezi zaměstnanci a jejich nadřízenými případně i s vedením firmy. Součástí této činnosti je často i nutnost udržovat dobré pracovní prostředí mezi samotnými zaměstnanci.

3.1.9 Péče o pracovníky

Jedná se o činnosti zaměřené na pracovní prostředí, bezpečnost práce, otázek pracovních podmínek - například. pracovní doby atp. Součástí je například i zajištění sociálních služeb pro zaměstnance.

3.1.10 Personální informační systém

Podstatnou částí této personální činnosti je schraňování veškerých informací o zaměstnancích. Tyto informace pak mohou sloužit jako podklady k různým analýzám pracovního výkonu, mezd, sociálních záležitostí nebo personálních činností.

3.2 Získávání pracovníků

Jak již bylo zmíněno, činností a úkolů, kterými se musí v rámci řízení pracovních zdrojů pracovníci HR oddělení zabývat, je mnoho. Vzhledem ke stanovenému cíli současně vyvíjeného modulu kandidátů pro HR Cockpit, který má sloužit hlavně k usnadnění procesu náboru nových pracovníků, se i v rámci rešerše dále zaměříme právě tímto směrem

3.2.1 Kompetence

Pokud mluvíme o zaměstnancích nebo kandidátech, velmi často se setkáváme s termínem kompetence. Dříve než se pustíme do samotného procesu získávání nových zaměstnanců je vhodné si tento termín definovat.

Pracovníka případně potenciálního kandidáta o zaměstnání je potřeba efektivním způsobem popsat, aby byla snadno definovatelná jeho kvalita. Obzvláště při výběru nového zaměstnance se jeví jako zásadní možnost popsat kvalitu kandidáta určitým, jednoduše hodnotitelným způsobem, aby bylo snadné identifikovat toho nejlepšího dostupného kandidáta. Rozšířeným modelem, tak je hodnocení tzv. kompetencí daných osob. Existuje více způsobů, jak lze termín kompetence přesně definovat. Například jako první v tomto kontextu definoval kompetence americký profesor Richard Elefther-ios Boyatzis jako „... schopnost člověka chovat se způsobem odpovídajícím požadavkům práce v parametrech daných prostředím organizace, a tak přinášet žádoucí výsledky“.[3] Pojem kompetence, je tak značně rozsáhlý a sloužící prvořadě k popisu schopností pracovníků či případných kandidátů na pracovní pozice. Jak upozorňuje Koubek, tak obzvláště v oblasti české personalistiky dochází v rámci termínu „kompetence“ ke spojení dvou různých termínů.[2]

- Odborné schopnosti - Odborná způsobilost či dovednosti vykonávat danou práci.
- Schopnosti chování - Schopnost žádoucího a efektivního pracovního chování. (v angl.. competency)

3.2.2 Definice pracovního místa

Pokud se rozhodneme obsadit pracovní pozici z externích zdrojů, je nejdříve nutné definovat několik následujících věcí.[4]

- Jaké místo chceme obsadit.
- Požadavky na kandidáta na tuto pozici.
- Míra tolerance při neplnění stanovených požadavků
- Tým zodpovědný za výběr
- Forma pro získání kandidáta.

3.2.3 Tvorba pracovní nabídky

Než je možné zveřejnit samotnou pracovní nabídku je nutné, aby HR oddělení přesně definovalo popis pracovní pozice. Nesmí se opomenout jasné stanovení vypisované pracovní pozice v rámci současné řídící struktury a rozhodovacích pravomocí. Potřebné je také jasně stanovit požadované kompetence pro vykonávaní dané vypisované pozice. Jejich nedílnou součástí je například stupeň teoretické přípravy (vzdělání) na celkovou nebo odbornou praxi. Pro výkon některých činností jsou zákonnou nebo jinou například interní podmínkou dále stanoveny požadovaná oprávnění (například řidičské oprávnění).[4]

Součástí tvorby pracovní nabídky ale není pouze stanovení požadavků na kandidáta. Nutné je také jasně stanovit co naopak nabízí zaměstnavatel.

3.2.4 Tým pro výběr zaměstnance

Podle charakteru obsazované pozice a její důležitosti je nutné stanovit tým odpovídající velikosti. Základní složkou tohoto týmu byl měl být personalista, jenž posuzuje obecné vlastnosti kandidáta a ručí na formální správnost procesu výběru nového zaměstnance. Dalším vhodným členem týmu by měl být případný přímý nadřízený nového pracovníka. Pokud to významnost pozice, na kterou je vedeno výběrové řízení vyžaduje, může být vhodné začlenit do týmu i nadřízeného přímého nadřízeného nového zaměstnance. Pokud to popis pracovní pozice vyžaduje, tak je možné do týmu odpovídajícímu za výběr nového pracovníka začlenit i odborníka, odpovídajícího za odbornou znalost uchazečů. Základním požadavkem na každého člena tohoto týmu pak je příslušná odborná fundovanost a dobré komunikační schopnosti. Každý člen týmu v této roli vystupuje jako reprezentant firmy a vysílá směrem ven určitý signál o úrovni celé společnosti.[4]

3.2.5 Forma výběru

V případě rozhodnutí získat nového zaměstnance, existuje několik základních způsobů, jak oslovit případné kandidáty.[4]

- Zveřejněním nabídky odpovídajícím způsobem a očekáváním reakcí.
- Přímým oslovením
- Výběrem z existující databáze
- Oslovením kandidáta na přímé doporučení důvěryhodného zdroje.

3.2.6 Výběr pracovníků

Jakmile společnost získala v předchozí fázi výběrového řízení soubor zájemců o zaměstnání, je nezbytné z nich v následujícím procesu vybrat toho nejkvalitnějšího kandidáta, kterému by bylo možné předložit nabídku k zaměstnání. Jak uvádí Koubek[2] - v praxi i literatuře se k tomuto účelu vžil následující postup

1. Zkoumání dotazníků a jiných dokumentů, předložených uchazečem.
2. Předběžný pohovor
3. Testování uchazečů pomocí tzv. testů pracovní způsobilosti.
4. Výběrový pohovor
5. Zkoumání referencí
6. Lékařské vyšetření
7. Rozhodnutí o výběru konkrétního uchazeče
8. Informování uchazečů o rozhodnutí.

Všechny tyto kroky samozřejmě není vždy nutné vykonat a celý proces je tak potřeba vždy přizpůsobit podle pozice, na kterou výběrové řízení probíhá. Dobrým zvykem je ale přihlédnutí k referencím, alespoň za předpokladu, že jsou objektivní. Vždy je ale dobré a důležité dodržovat následující zásady.[2]

- Veškeré informace poskytnuté kandidáty je nutné verifikovat, jelikož můžou mít různou úroveň pravdivosti. Kandidátem deklarované dovednosti je tak například možné ověřit pomocí testů atp.
- Je vhodné nepoměřovat kandidáty jen oproti stanoveným požadavkům, ale i mezi sebou a vybrat takového kandidáta, kterým našim požadavkům odpovídá nejlépe.
- Posuzovat schopnost uchazeče zapojit se do již existujícího pracovního týmu.

- Soustředit se nejen na skutečnost, zda je kandidát požadovanou práci schopen vykonávat, ale zda ji doopravdy i chce vykonávat a není jen motivován případnými benefity.
- Neopomenout skutečnost, že při výběrovém řízení se i sám kandidát rozhoduje jestli chce ve firmě pracovat. Výběrové řízení je oboustranný proces.

3.2.7 Pohovor

Jak již bylo zmíněno, základní částí výběru zaměstnance je pohovor. V rámci samotného procesu se může v různých formách dokonce vyskytovat vícekrát například „předběžný pohovor“ a „výběrový pohovor“. Pohovor slouží k celé řadě cílů - například.[5]

- Získání dodatečných a hlubších informací o kandidátovi.
- Možnost poskytnout kandidátovi informace o firmě a práci v ní.
- Možnost posoudit osobnost a charakterové rysy kandidáta.
- Možnost vytvářet dobrou pověst firmy.

Pohovor může mít více forem například podle počtu účastníků.

- 1+1 .Vhodný pro navození méně formální atmosféry. Klade vyšší nároky na zástupce firmy a nemusí být zajištěno objektivní hodnocení kandidáta.
- Pohovor před výběrovým týmem. Složitější na organizaci a vytváří větší tlak na uchazeče formální atmosférou pohovoru. Hodnocení kandidáta je ale objektivnější a přesnější.

Pohovory můžeme dále rozdělovat podle obsahu a průběhu takto:

- Nestrukturovaný - jeho forma i obsah jsou utvářeny v průběhu pohovoru.
- Strukturovaný - obsah, pořadí otázek, ale i čas vyčleněný jednotlivým tématům, je pevně stanoven.

3.2.8 Závěr výběrového řízení

V okamžiku, kdy je rozhodnuto o vítězi výběrového řízení, je nutné o tom daného kandidáta urychleně informovat. Vhodné je kontaktovat kandidáta nejdříve ústně (například telefonicky) a následně předložit písemnou nabídku zaměstnání s datem, do kterého by se měl kandidát rozhodnout, zda nabídku příjme či ne. Současně je nutné informovat i neúspěšné kandidáty, že je bohužel není možné zaměstnat. Výhodné však ale je, přímo neodmítat několik dalších nejlepších uchazečů, kteří tak mohou sloužit jako rezerva, kdyby preferovaný kandidát nabídku odmítl. Důležité je neodmítat kandidáty stroze, je nutné jim za účast poděkovat a vyjádřit lítost, že je tentokrát nelze zaměstnat.

Odmítnutí kandidáti tak nezaujmou oproti firmě automaticky negativní postoj. Nebudou tak v první řadě šířit špatnou pověst o firmě a zadruhé je možné je někdy v budoucnu oslovit v případě dalšího výběrového řízení.[2]

Kapitola 4
Alternativní aplikace

Jak už je asi po předchozí kapitole patrné, problematika správy lidských zdrojů je relativně obsáhlé téma. Zároveň platí, že tuto problematiku musí určitým způsobem řešit každá společnost bez ohledu na svou velikost. Velikost dané společnosti samozřejmě ovlivňuje způsob možného řešení, a tak u opravdu malých firem je možné vystačit si při správě lidských zdrojů s vedením záznamů pomocí obyčejných tabulek nebo jiných nespecializovaných metod.

U problematiky takto obecné a rozšířené je zákonitě dané, že na trhu vzbudí poptávku po softwarovém řešení. Najít tak aplikace, které se problematikou správy lidských zdrojů a vybíráním nových pracovníků zabývají není těžké. Dokonce i jen v rámci domácího prostředí se dají nalézt aplikace, které se zabývají požadovanými problémy.

4.1 Mark

Aplikace Mark nabízená společností Profesia[1], která mimo jiné provozuje již od roku 1997 pracovní portál Profesia.sk . Samotná společnost pak provozuje i několik dalších webových aplikací věnujících se převážně problematice lidských zdrojů.

Obrázek 4.1. Aplikace Mark [6]

Společnost sama pak aplikaci ve zkratce popisuje jako „... efektivní řešení pro online správu uchazečů ve výběrových řízeních. Nástroj je přehledný, intuitivní, pomáhá zlepšovat image společnosti a díky němu se v návalu uchazečů určitě neztratí ten nejlepší kandidát“[6].

Podle ceníku[6] pak v různě cenově ohodnocených balíčcích nabízí následující služby.

- Správa
  - třídění
  - poznámky
  - vyhledávání
  - kalendář pohovorů
  - historie aktivit
  - archiv uchazečů
- Komunikace
  - hromadné

[1] `http://www.profesia.cz/`

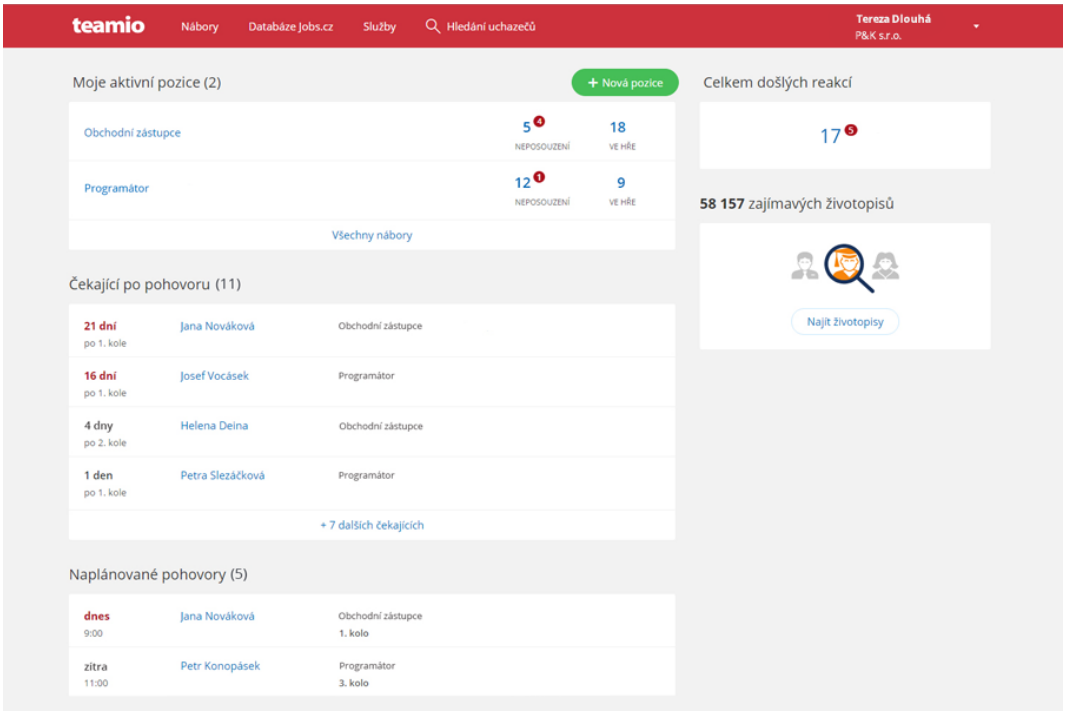
  - personalizované e-maily
  - vlastní šablony
  - přeposílání
- Automatizace
  - automatická e-mailová odpověď
  - duplicitní uchazeči
- Dotazníky
  - předvýběr uchazečů pomocí otázek
- Korporátnost
  - logo v e-mailové komunikaci
- Export
  - export informací o uchazeči
- Statistiky
  - měsíční statistické reporty

4.2 Teamio

Aplikace Teamio je podobně jako aplikace předchozí vystavěna okolo pracovního webového portálu, tentokát okolo jobs.cz a prace.cz. Jedná se produkt společnosti LMC[1]. Mezi hlavní propagované funkcionality aplikace patří následující:

- Podpora vytváření a propagace pracovních inzerátů
- Správa uchazečů, jejich kategorizace a jednoduché hodnocení
- Vedení vlastní databáze uchazečů
- Vyhledávání životopisů na pracovních portálech
- Možnost zapojit do procesu náboru další kolegy
- Spravování kol pracovních pohovorů a jejich organizace

[1] `www.lmc.eu`



Obrázek 4.2. Aplikace Teamio [7]

Kapitola 5
Analýza

Nutností pro vypracování této analýzy byla velmi těsná spolupráce s plánovanými typickými koncovými uživateli, tedy pracovníky HR oddělení. Přes veškerou možnou analýzu celkové problematiky lidských zdrojů je silně nevhodné navrhnout systém pro praktické použití bez konzultaci se skutečnými odborníky v dané problematice, kteří budou zároveň i koncovými uživateli celé aplikace.

Zároveň, jak už asi bylo patrné z rešerší části této práce je celá problematika lidských procesů značně rozsáhlá a ve svých postupech a procesech ne vždy stejná. Účelem této aplikace je tak vybudovat systém, který v prvním případě pokrývá použité procesy konkrétně určeného personálního oddělení, které má zákonitě svá vlastní specifika.

K jasnému odhalení všech požadavků a procesů koncových uživatelů byla nutná celá série schůzek s pracovníky oddělení HR. Mezi jednotlivými schůzkami a jednáními byly vždy zapracovány do dokumentace nové poznatky, aby mohly být při příští příležitosti verifikovány uživateli. Tímto interaktivním způsobem mohla vzniknout základní specifikace celé aplikace HR Cockpit, jenž se nachází v rámci této kapitoly.

5.1 Rozsah práce

Jedním z hned prvních zjištění při setkání s klientem byla vysoká očekávání na výslednou aplikaci. Prvotní představa celé aplikace byl systém pokrývající veškerou administrativní činnost, nutnou při správě lidských zdrojů středně velké společnosti. Proto prvním úkolem bylo dospět k vhodnému postupu vývojového procesu, jak k takovému výsledku jednou dojít.

Jako nereálné se jevilo pustit se přímo do vývoje takto rozsáhlé aplikace. Hned na prvním jednání začalo být zřejmě, že výsledná aplikace by měla desítky a možná i více uživatelských případů. Samotná kompletní analýza a příprava dokumentace by tak zabrala nemalé časové vytížení a náklady.

Obrázek 5.1. Spirálový přístup vývoje. zdroj wiki.

Zároveň bylo také hned jasné, že při vyhotovení takto rozsáhlého systému by bylo zcela nevhodné vycházet z některého jednoduchého vodopádového procesu vývoje softwaru. Je totiž téměř jisté, že by se nikdy nepodařilo navrhnout dokumentaci takto rozsáhlého systému, u které by se v pozdějších fázích vývoje nenašly žádné nedostatky či případně změnové požadavky. Zpětné změny v takto rozsáhlé dokumentaci by byly vysoce nákladné a mohly by dokonce ohrozit celou budoucnost projektu.

5.1.1 Rozdělení projektu

Proto bylo rozhodnuto změnit přístup a přejít spíše na spirálový proces vývoje. Snahou bylo základní představy o fungování celé aplikace rozdělit do menších fragmentů, které by bylo jednoduší a rychlejší navrhnout, implementovat a hned uvést do provozu. Na těchto menších částech by se pak mohlo bezpečně fungovat na základě vodopádového principu s určitým menším překryvem jednotlivých fázích.

Základním požadavkem na tyto nové celky celého projektu byla jejich funkcionální celistvost. Bylo tedy nutné, aby každá postupně vyvíjená část projektu mohla fungovat samostatně bez dalších, na později plánovaných rozšíření. Tímto rozhodnutím je dosaženo, že zatímco ostatní části projektu jsou teprve v různých fází vývoje, koncový uživatel už může předchozí aplikační celky bez větších omezení užívat.

Klíčové bylo určit, které části aplikace bude věnována v rámci vývoje věnována pozornost jako první. Jako ideální se tak jeví modul zabývající se výběrových procesem nových zaměstnanců. Jeví se tak svým rozsahem a jasně definovanými procesy, jenž musí aplikace svou funkcionalitou pokrývat. Zároveň je od ostatních plánovaných modulů dostatečně oddělen a v případě pozdějšího zrušení celého projektu nabízí užitečný a samostatný nástroj pro jednu z klasických činností správy lidských zdrojů.

Toto oddělení samozřejmě není kompletní, během analýzy byla nalezena celá řada případných kontaktních míst, kdy by tento modul měl nějakým způsobem interagovat, případně minimálně sdílet nějaké vlastnosti nebo vazby na určité elementy. Asi nejočividnější vazba je mezi úspěšné kandidáty výběrového a případné zaměstnance, kteří se v dalších modulech určitě budou vyskytovat. Další očividnou vazbou jsou dané kompetence, jenž se často zmiňovali v rámci rešerše. I v rámci analýzy toho jednoho modulu se ukáže, že tyto kompetence se vyskytují v mnoha různých kontextech a je téměř jisté, že tomu v dalších modulech nebude jinak. Z tohoto důvodu bylo potřeba na tyto eventuality brát potaz a některé součásti aplikace navrhovat robustněji než se může na první pohled zdát nutné, s důrazem na pozdější rozšíření.

V otázce prvně vyvíjené části aplikace zůstal jediný otazník. Jedná se o plánovaný modul administrace. Tento modul má za hlavní úkol spravovat obecná data vyskytující se v rámci aplikace, například uživatele nebo různé číselníkové entity. Sám o sobě nemá žádný smysl ani žádnou přímou vazbu na procesy nutné pro správu lidských zdrojů, a proto se nehodí pro začátek samotného vývojového procesu. Zároveň je zřejmé, že se bude muset postupně vyvíjet společně s růstem celé aplikace. Faktem zůstává, že jeho absence v rámci této první verze aplikace HR Cockpit způsobuje jisté uživatelské obtíže, kdy je nutné, aby některá potřebná data přímo vložena do databáze aplikace. Jedná se nepochybně o nepříjemnost, ale zároveň se nejedná o kritický problém, který by zabraňoval úvodní verzi aplikace úspěšně používat. Každopádně je tak alespoň lépe jasné, který část aplikace by měla být vyvíjena jako další.

5.2 Katalog požadavků

Jedním z klíčových prvků analýzy navrhovaného systému je vytvořit jasný a srozumitelný katalog všech požadavků na cílový systém. Tento katalog je pak vícenásobně

využíván v průběhu celého procesu vývoje koncové aplikace. V neposlední řadě také slouží, jako podklad ke koncovému akceptačnímu testování zákazníka finálního produktu. I z tohoto důvodu, je tak naprosto nezbytné, aby všechny požadavky byly opravdu jednoznačné a nemohlo během vývoje dojít k nějakým nejasnostem, vedoucím k případnému odklonu od zákazníkem požadovaných vlastností systému.

Tento soupis požadavků na systém můžeme rozdělit na dvě základní skupiny. První z nich jsou požadavky funkční. Tato skupina požadavků jasně definuje, jakou funkcionalitu zadavatel práce od aplikace očekává. Druhou skupinou požadavků jsou požadavky obecné, které se obecně vztahují na celý systém a upřesňují způsob navržení aplikace.

5.2.1 Funkční požadavky

V první skupině funkčních požadavků jsou uvedené veškeré požadavky, zabývající se problematikou správy kandidátů.

- Aplikace bude umožňovat CRUD operace nad entitou kandidáta
- U kandidáta budou uchovávány základní identifikační a kontaktní údaje (Jméno, adresa, telefon atd.)
- U každého kandidáta musí být uchována informace o udělení souhlasu se zpracováváním osobních údajů.
- U každého kandidáta musí být evidována představa jeho finančního ohodnocení.
- Ke kandidátovi musí být možno uchovávat textové komentáře uživatelů.
- U kandidáta se musí evidovat, o které lokace (pobočky) společnosti má zájem.
- Pro každého kandidáta musí být možné uchovávat dokumenty (životopis, fotografie apod.)
- Aplikace musí podporovat tzv. černou listinu kandidátů, kteří nebudou doporučováni do výběrových řízení
- U každého kandidáta musí být možné evidovat 0-N zdrojů, ze kterých byl kontakt na kandidáta získán
  - Zdroje kandidátů musí být možné třídit podle kategorie např. (Internetové portály-jobs.cz,novinová inzerce – Anonnce atd.)
  - U zdrojů může být dále uvedena tzv. probíhající akce(např. inzertní kampaň)
  - Musí být možno vložit i speciální typ zdroje: Osobní doporučení – konkrétní osoba (např. aktuální zaměstnanec). Je nutné, aby byl uchován přesný údaj o kandidátech, které daná osoba doporučila.
- Každý kandidát může mít 0-N hodnotitelných dovedností.
  - U každé dovednosti musí být možné evidovat kandidátovo sebehodnocení.
  - U každé dovednosti musí být evidována hodnota aktuálního (posledního) hodnocení.
  - Hodnotitelé mohou přidávat ke každé kandidátově dovednosti hodnocení (slovní komentář, úroveň kandidátovy dovednosti podle odpovídající stupnice viz. dovednosti)

Dále jsou uvedeny požadavky, které se nějakým způsobem dotýkají výběrových řízení a veškerých jejich procesních vlastností, například jejich fáze atd.

- Každé výběrové řízení se skládá z 0-N fází(kol).
- Aplikace bude umožňovat CRUD operace nad entitou výběrových řízení a fází.
- Každé výběrové řízení může mít definováno (0-N) dovedností (na definovaných úrovních).

- Každé výběrové řízení musí mít uvedeno (1-N) lokací.
- Ke každé fázi výběrového řízení musí být možné přizvat tzv. hodnotitele.
- Každý hodnotitel může zadat své hodnocení kandidáta v daném kole a případně doporučit/nedoporučit jeho postup do další fáze.
- Pracovník HR musí mít při rozhodování o postupu kandidáta do další fáze k dispozici všechny podklady od hodnotitelů (slovní hodnocení, doporučení a kandidátovo umístění v hodnotitelově žebříčku) a všeobecné hodnocení kandidátových hodnocení.

Další skupina funkčních požadavků je pak věnována požadavkům, týkajícím se dovedností neboli takzvaných kompetencí.

- Každá dovednost může mít 1-N, tagů, pomocí kterých lze dovednost v systému vyhledávat a zadávat.
- Každá dovednost může mít svou vlastní stupnici hodnocení (standardně 0-4)
- Při pokusu zadat neexistující dovednost, systém pomůže uživateli automaticky vytvořit novou dovednost.

V poslední skupině požadavků jsou dále uvedeny ostatní funkční požadavky, které se sice přímo nevztahují k žádné z předešlých kategorií, ale mají spíše globální charakter a tedy dopad na celkové chování systému.

- Aplikace musí fungovat v kontextu uživatele.
- Aplikace bude umožňovat přihlásit se uživateli do systému pomocí přihlašovacího jména a vlastního tajného hesla.
- Aplikace bude umožňovat odhlášení uživatele.
- Uživatel bude moci změnit svého současné heslo.
- Aplikace bude umožňovat práci v kontextu více „mandantů“ (subjektů). Tudíž je nutné od sebe oddělit veškerá soukromá data jednotlivých mandantů. Zároveň je ale nutné umožnit jednomu uživateli figurovat ve více mandantech v různých rolích, bez nutnosti se do aplikace znovu přihlašovat. Aplikace musí být schopna reflektovat na prezentační stránce právě aktivního mandanta (např logo atd.)

5.2.2 Nefunkční požadavky

- Podporované prohlížeče
  - Internet Explorer 6+
  - Firefox 3.6+ (PC, Mac)
  - Safari 4+
  - Chrome 10+
  - Opera 11+ (PC, Mac)
- Aplikace musí obsahovat mandanty s odpovídající grafickou modifikací aplikace.
  - Baader Computer
  - LHMS
- Systém bude pracovat nad databází PostgreSQL
- Systém bude běžet na serveru Tomcat 7
- Lokalizace - Aplikace bude vyhotovena s českou lokalizací, bude ale přizpůsobena i k jednoduchému přidání dalších lokalizačních balíčků.
- Aplikace musí podporovat případná další plánovaná rozšíření (např modul zaměstnanců). Možnost například o převedení úspěšného kandidáta do databáze zaměstnanců, případně spojení záznamu zaměstnance s osobou, doporučující kandidáta atd.
- Aplikace musí být vyvíjena podle standardů společnosti Baader Computer.

5.3 Konkretizace požadavků

Předchozí katalog požadavků slouží hlavně jako jejich přehled a ačkoli je mnoho z výše uvedených požadavků jednoznačných, některé vyžadují důkladnější specifikaci.

5.3.1 Kandidáti

Pro každého kandidáta obsaženého v systému musí být evidovány následující údaje.

- Jméno
- Příjmení
- Telefon
- E-mail
- Datum narození
- Lokalita
- Adresa
- Dovednosti
- Požadovaný plat
- Zdroj kandidáta
- Datum vložení kandidáta do systému
- Souhlas se zpracováním osobních údajů (datum souhlasu)
- Černá listina
- Dokumenty
- Poznámky – Seznam poznámek obsahující následující pole :
  - Autor
  - Datum přidání
  - Text poznámky

5.3.2 Zdroje kandidátů

U každého kandidáta evidovaného v systému může být uveden zdroj jeho CV. Každý kandidát může mít uveden 0-N různých zdrojů s tím, že u každého je nutné uvést jeho prioritu, která udává nejrelevantnější zdroj pro konkrétního kandidáta. Každý zdroj může zároveň spadat pod několik různých kategorií, podle kterých je možné jednotlivé zdroje filtrovat.

Příklady zdrojů a jejich kategorií.

- Internetový portál
  - jobs.cz
  - prace.cz
- Vysoká škola – inzerce
  - ČVUT FEL
  - ČVUT FIT
- Osobní doporučení
  - V takovém případě je uchováván záznam o osobě (případně zaměstnanci), která poskytla doporučení.

Mimo typu zdroje je možné u kandidáta evidovat také i konkrétní akci, která se k danému zdroji váže a jenž vedla k získání kandidáta. Příklad kompletního záznamu zdroje konkrétního kandidáta.

Internetový portál/jobs.cz – Inzerát (od 1.1.2014 do 1.2.2014)

5.3.3 Dovednosti

Parametr, sloužící ke specifikaci kandidátů/zaměstnanců a zároveň vypsaných výběrových řízení. Každou dovednost lze dále ve vztahu k dané entitě definovat pomocí tzv. úrovně ovládnutí dovednosti. Ovládnutí dovednosti může nabývat hodnot 0-4 . (0 – kandidát dovednost neovládá, 4 kandidát je v dané oblasti expertem)

- Kandidát – Každý kandidát má 0 – N dovedností, která ovládá. Úroveň dovednosti specifikuje úroveň, na které daný kandidát dovednost ovládá.
- Výběrové řízení – Každé řízení má 0-N požadovaných vlastností. Úroveň dovednosti specifikuje minimální úroveň dovednosti, na které by ji měli kandidáti ovládat.
- Každý hodnotitel má 0-N dovedností, které může hodnotit.

Každá dovednost musí mít svou vlastní sadu úrovní. Jako základní sada úrovní musí být použita výše uvedené prostá číselná stupnice, ale aplikace musí umožňovat případné rozšíření o správu dovedností a tudíž i jejich sad. Dá se totiž očividně očekávat, že některé druhy dovedností budou moci mít své vlastní specifické úrovně - například jazyky.[1]

5.3.4 Standard společnosti Baader Computer

Tento nenápadný požadavek má na celou práci zásadní vliv. Pravděpodobně každá společnost zabývající se vývojem softwaru musí mít jasně stanoveny své standardy, postupy a samozřejmě i požadavky na výslednou kvalitu koncového produktu.

Zároveň je silně praktické mít napříč projekty určitou alespoň v určitých aspektech společnou architekturu vyvíjených systémů. Důvody k tomu jsou celkem zřejmé. V případě změn ve vývojovém týmu případě změnách pracovního zaměření jednotlivých vývojářů mezi projekty je nutné, aby tyto projekty byly určitým způsobem konzistentní. Bylo by silně nevhodné, aby při každé změně ve vývojovém týmu, museli ostatní vývojáři dlouze zjišťovat základní fungování daného projektu a organizaci zdrojových kódů atd.

S ohledem na případné budoucí rozšiřování této aplikace o nové funkcionální moduly, je nutné aby i tato aplikace splňovala všechny tyto standardy. Je totiž velmi pravděpodobné, že se na rozšiřování této aplikace budou jednou podílet i vývojáři relativně v těchto postupech nezkušení, ať buď v rámci svých studentských prací či jen trainee programů, a tak je ještě více nutné, aby se tato aplikace těmito standardy přísně řídila a nevzbudila v potenciálních nových vývojářích špatné návyky.

5.4 Negativní vymezení

V rámci tvorby specifikace vyvíjené aplikace je také doporučeno uvést kromě katalogů požadavků a funkcionalit, které aplikace bude obsahovat naopak i seznam těch funkcionalit, které obsahovat nebude. Díky tomu je lepší šance zajistit realistické očekávání na vyvíjený systém. V rámci tohoto seznamu, tak musí být vyjmenovány takové funkce, které by zákazník mohl očekávat a které přesto v rámci navrhovaného systému obsažené nejsou[8].

Důležitost takového vymezení obzvláště narůstá v tomto konkrétním případě, kdy původní představy zadavatele na systém byly značně rozsáhlé. V rámci počáteční analýzy byla specifikace a rozsah této případně první verze aplikace jasně definována s ohledem

[1] Společný evropský referenční rámec `http://www.coe.int/t/dg4/linguistic/DNR_EN.asp`

na další plánovaná rozšíření, a proto je obzvláště nutné se v této části ohradit od funkcionalit, které jsou sice zákazníkem požadované, ale v rámci dohody plánované až do verzí příštích

Součástí verze aplikace HR Cockpit vyvíjené v rámci této práce, tak nejsou následující funkcionality.

- Modul administrace
  - Správa uživatelů
  - Správa dovedností
  - Správa výčtových entit (pracovní pozice,zdroje kandidátů, dostupné lokality atd.)
- Reporty
  - Generování reportů sloužící k vyhodnocování výběrových řízení, reklamních kampaní atd.
- Kalendář a možnost plánovat pohovory
- Upomínkový systém pro hodnotitele
- Správa komunikace s kandidátem
- Modul zaměstnanců

5.5 Uživatelské role

Cílovými uživateli aplikace jsou hlavně zaměstnanci HR oddělení, ale počítá se samozřejmě se skutečností, že do výběrového řízení zasahují i případní vedoucí oddělení, pro která jsou hledáni noví pracovníci, případně zaměstnanci vyššího managmentu společnosti. Dále je nutné brát v potaz, že do samotného výběrové řízení musí být možné přizvat i další uživatele, jejichž hlavním úkolem je hodnotit odborné dovednosti jednotlivých kandidátů. Aplikace tak umožňuje figurování uživatelů v následujících rolích.

5.5.1 HR manažer

Nejvýše postavené uživatelské role v rámci celé správy lidských zdrojů. Je odpovědná za všechny kandidáty a výběrová řízení. Právě tato uživatelské role rozhoduje zda nějaký kandidát bude přijat nebo vyřazen z výběrového řízení.

5.5.2 HR asistent

Uživatel odpovědný za spravování databáze výběrových řízení a kandidátů. Hlavním omezením oproti HR manažerovi je, že nemůže rozhodovat o postupu kandidátů v rámci výběrového řízení.

5.5.3 Hodnotitel

Uživatel určený k hodnocení kandidátů ve vybraném výběrovém řízení či fázi. K výběrovému řízení je tato role přiřazena pracovníkem HR. Hodnotitel má omezené možnosti sledování probíhajícího výběrového řízení a má možnost se k jednotlivým kandidátům vyjadřovat.

5.5.4 Vedoucí týmu

Uživatelské role velmi podobná hodnotiteli jen s omezenější přístupem k datům kandidátů a výběrových řízení.

5.5.5 Managing Director

Uživatel s možností sledování všech výběrových řízení a ke všem odpovídajícím statistikám. Nejedná se o přímého účastníka procesů odehrávajících se v rámci aplikace, ale musí mít možnost je všechny neomezeně sledovat.

5.6 Proces výběrového řízení

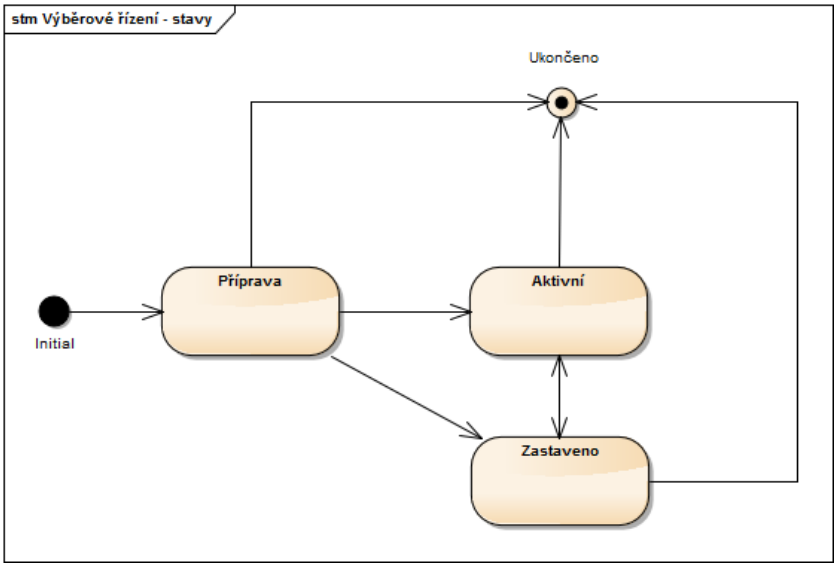
Nejdůležitějším aspektem celé navrhované aplikace je samotný proces náboru nových zaměstnanců. Aplikace musí tudíž pokrývat celý životní cyklus výběrového řízení. Každé výběrové řízení se skládá z 1-N fází. Fází je myšlená možná fáze výběrového řízení, ve které se mohou nacházet kandidáti, kteří se ho účastní. Např. úvodní selekce, 1.-N. kolo výběrového řízení. Každé výběrové řízení má automaticky vygenerovanou první fázi - tzv. selekci, do které jsou vkládáni všichni kandidáti daného výběrového řízení. Každé výběrové řízení se skládá ze sekvence sériově uspořádaných fází - např: úvodní selekce, 1. kolo,2 kolo.

5.6.1 Stavy výběrového řízení

Každé výběrové řízení se musí nacházet v jednom z několika možných stavů.

- Příprava
- Aktivní
- Zastaveno
- Ukončeno

V rámci přípravy je možné definovat jednotlivé vlastnosti výběrového řízení, včetně počtu a pořadí jednotlivých fází. Během tohoto stavu je možné přidávat do výběrového řízení kandidáty, ale nelze je zatím hodnotit. Zároveň pak už v pozdějších fázích není možné zasahovat do struktury jednotlivých fází, aby byly zaručeny rovné podmínky pro všechny kandidáty a všichni tak museli projít stejnými fázemi výběrového řízení.



Obrázek 5.2. Stavy výběrového řízení

5.6.2 Rozhodnout kandidáta

Rozhodnutí, zda některý kandidát postupuje či nepostupuje do další fáze, záleží na HR manažerovi. Jako podklady pro toto rozhodnutí mu slouží kandidátovo hodnocení. Rozhodnout kandidáta může nabývat jednu z následujících hodnot.

- Schválen
- Neschválen
- Čeká na rozhodnutí

5.6.3 Hodnocení kandidáta

Hodnocení kandidáta se skládá ze dvou nezávislých částí.

- Hodnocení dovedností
- Hodnocení fáze

Hodnocení dovedností je nezávislé na hodnocení v rámci kola nebo výběrového řízení a mohou se ho účastnit všichni hodnotitelé. Ohodnotit lze každou kandidátovu dovednost a přiřadit k hodnocení textový komentář.

Hodnocení v rámci fáze může provést pouze uživatel přiřazený k dané fázi jako hodnotitel. Hodnocení v rámci fáze se skládá z textového hodnocení kandidáta hodnotitelem a jeho výsledného návrhu na rozhodnutí kandidáta. Hodnotitel může mít na kandidáta jeden z následujících názorů.

- Schválen
- Neschválen
- Čeká na rozhodnutí

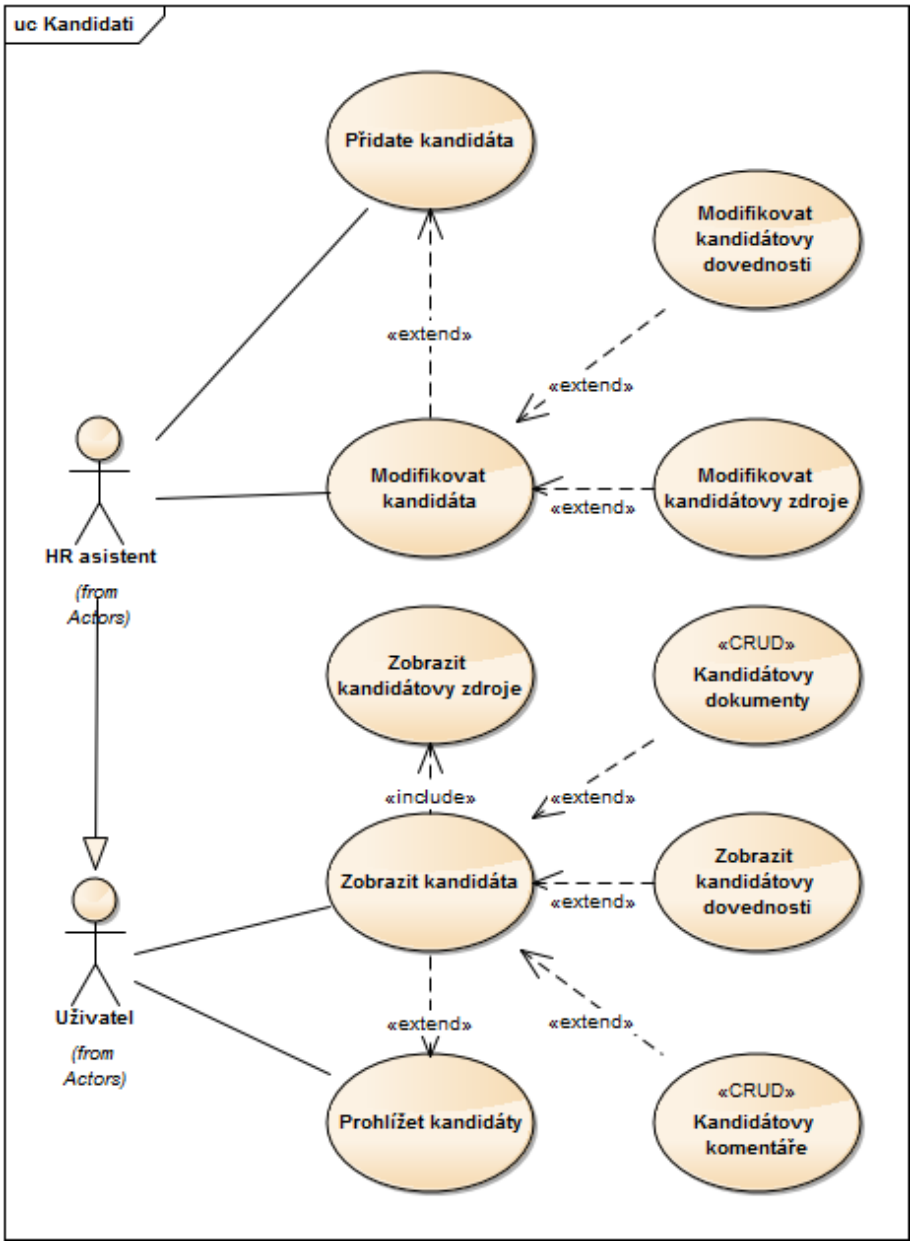
5.7 Model případu užití

Dalším stupněm zpracování požadavků je model případů užití. V tomto modelu jsou na základě funkčních požadavků vytvářeny tzv. use case, neboli případy užití. Případy užití v rámci mapování požadavků plní funkční požadavky a graficky znázorňují vztahy mezi sebou navzájem a případnými uživateli. Navrhovaná aplikace čítá zhruba něco kolem třiceti různých případů užití různé složitosti. Pro snadné znázornění byl model případů užití rozdělen do několika částí podle oblasti funkcionality, do které jednotlivé případy užití spadají.

Součástí návrhu modelu případu užití byl i podrobný návrh scénářů k jednotlivým případům užití. Vzhledem k celkovému většímu počtu těchto případů, je dokumentace scénářů značně rozsáhlá. Proto se nachází pouze v elektronické podobě na CD, které je přiložené u této práce.

5.7.1 Kandidáti

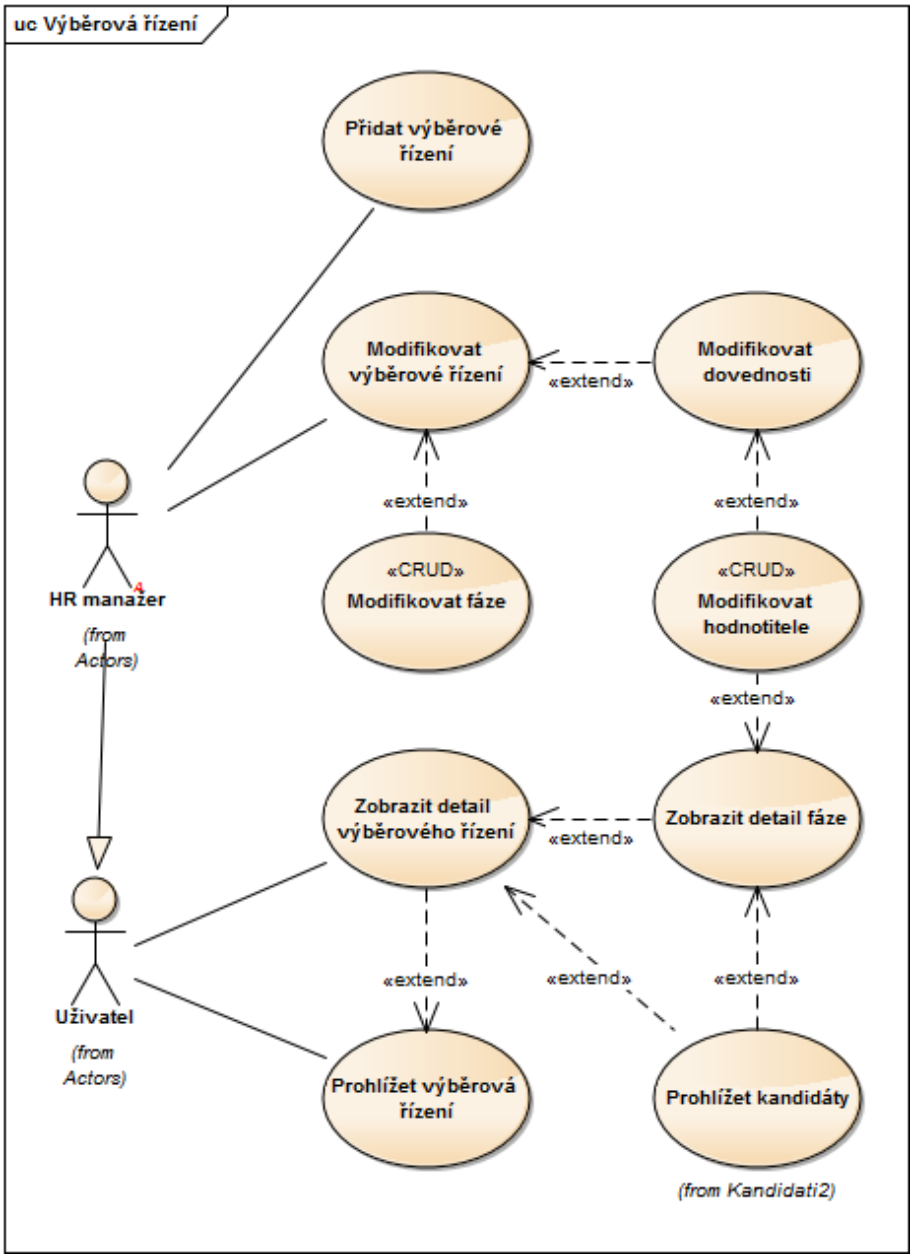
Skupina základních případů užití sloužící ke správě kandidátů. Uživatel s rolí „HR asistent“ slouží jako základní správce všech kandidátů, jehož úkolem je přidávat a upravovat základní údaje kandidátů, jako jsou například jeho jméno, datum narození, životopis, ale i třeba zdroje, ze kterého byl kontakt na kandidáta získán.



Obrázek 5.3. Model případu užití - kandidáti

5.7.2 Výběrová řízení

Další skupinou případů užití aplikace jsou výběrová řízení. V rámci výběrových řízení je, kromě základní možnosti prohlížení samotného výběrového řízení a jeho parametrů včetně jeho fází, nutné pokrýt i požadavky na základní administraci výběrových řízení. Klíčovou uživatelskou rolí v tomto případě je tzv. „HR manažer“. Jeho úkolem je spravovat výběrová řízení, tudíž musí samotná výběrová řízení vytvářet a následně případně editovat. V rámci této správy výběrových řízení HR manažer nastavuje i počet fází neboli kol výběrového řízení v jejichž průběhu jsou kandidáti hodnocení a postupně tříděni. Dále HR manažer může pro každou fázi zvlášť nastavit skupinu tzv. hodnotitelů, kteří se v dané fázi můžou ke kandidátům vyjádřit a tím se podílet na výsledném rozhodnutím. zda konkrétní kandidát postoupí či nepostoupí do případné další fáze



Obrázek 5.4. Model případu užití - výběrová řízení

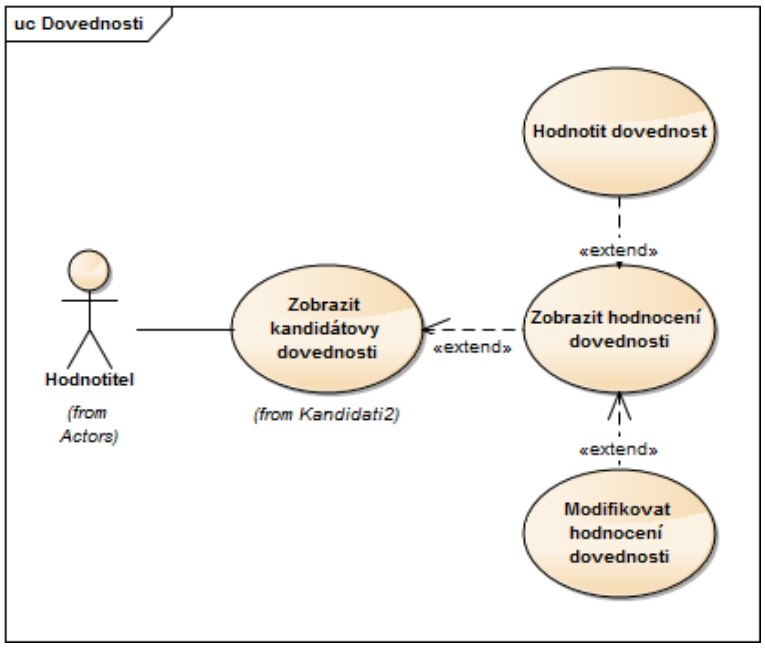
výběrového řízení. Dalším důležitým parametrem výběrového řízení, který je potřebné nastavit. jsou dovednosti, vyžadované od kandidátů v rámci výběrového řízení.

5.7.3 Dovednosti

Případy užití spadající do této kategorie se týkají dovedností neboli kompetencí jednotlivých kandidátů. Případy užití v této skupině jsou velmi úzce spjaty s dalšímu případy užití ze sekce „Kandidáti“, jelikož podobně jako ony, nastavují jeden ze základních parametrů kandidátů. Dá se zároveň očekávat, že HR asistent bude při vkládání nového kandidáta do systému rovnou přidávat i jeho dovednosti. Ostatní případy užití v této skupině se už ale týkají spíše než zadáváním samotné dovednosti kandidáta určením úrovně, na které tuto dovednost kandidát ovládá. Na rozdíl od hodnocení kandidáta popsaného v následující sekci se hodnocení samotných dovedností nevztahuje přímo

k žádnému výběrovému řízení či jeho fázi. Vyplývá to z toho, že informace o úrovni dovednosti kandidáta je všeobecnější údaj, který může být relevantní při hodnocení více fází, ale třeba i různých na sobě nezávislých výběrových řízení.

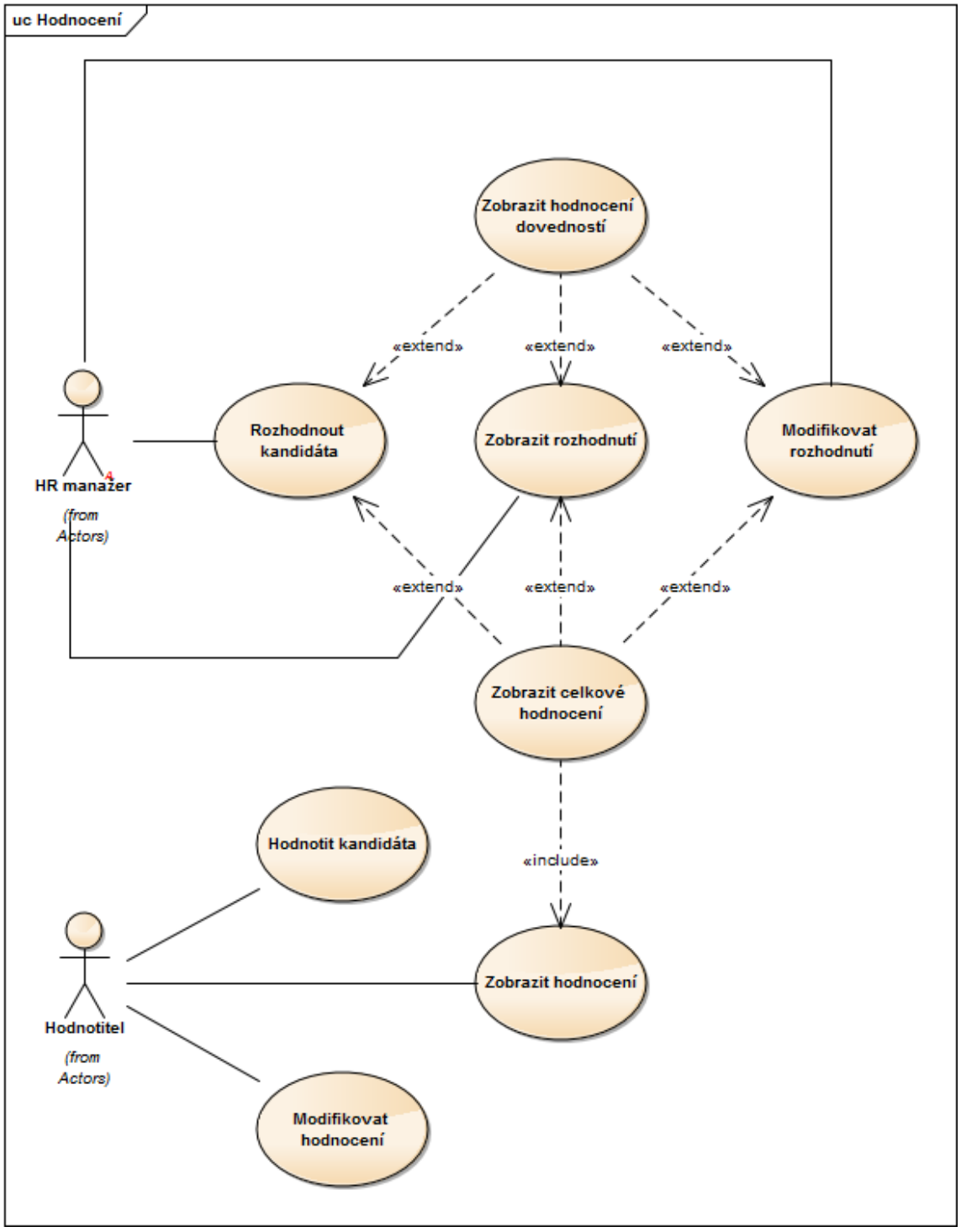
Systém počítá s dvěma teoretickými možnostmi vzniku hodnocení dovednosti. První variantou může být doložení úrovně ovládané dovednosti kandidáta z externího zdroje, například kandidátem dodaným certifikátem. V takovém případě může HR asistent do systému přímo vložit danou úroveň dovednosti. Další možností ověření dovedností kandidáta může být ale i interní testování kandidátových dovedností, například v rámci některé fáze výběrového řízení. V takovém případě má možnost ohodnotit dovednost některý z hodnotitelů přiřazený k fázi výběrového řízení, jehož se kandidát účastní.



Obrázek 5.5. Model případu užití - dovednosti

5.7.4 Hodnocení

Poslední skupina případů užití popisuje hodnocení kandidátů a rozhodování o jejich postupu do další fáze případně o jejich vítězství v rámci výběrového řízení. Uživatelé s uživatelskou rolí „hodnotitel“ mají možnost spravovat svá hodnocení pro jednotlivé kandidáty v rámci fáze výběrového řízení, ke které byli přiřazeni. Uživatel s uživatelskou rolí „HR manažer“ pak má možnost učinit závazné rozhodnutí na základě systémem poskytnutých informací o kandidátovi, úrovni jeho dovedností a doporučení všech zainteresovaných hodnotitelů.



Obrázek 5.6. Model případu užití - hodnocení

Kapitola 6
Návrh

6.1 Zvolené technologie

Vzhledem k požadavkům zadavatele je nutné respektovat, kvůli případnému rozšíření aplikace, běžně používané technologie společnosti Baader Computer. Tudíž bude vývoj koncové aplikace probíhat na platformě Java EE a využívat frameworky Spring, Hibernate a Ext JS.

6.1.1 Spring

Spring je rozsáhlý framework pro platformu JAVA, nabízející svými moduly řešení pro mnoho problematik spjatých s vývojem JAVA EE aplikací. Spring, ale díky své modularitě umožňuje používat jen ty části frameworku, které uživatel opravdu potřebuje a tudíž nehrozí nebezpečí nutností využívat takové části, které uživatel skutečně nevyužije. Mezi hlavní funkcionality a poskytované moduly pak patří řešení vkládání závislostí (Dependency Injection), práce s daty (Data Access/Integration), MVC či podpora aspektově orientovaného programování, řešení zabezpečení a mnoho dalšího.

6.1.2 Hibernate

Dalším použitým frameworkem je Hibernate. Hibernate - slouží k tzv. ORM nebo objektově relačnímu mapování. Jedná se o jednu z implementací Java Persistence API (JPA). Jeho využití tak velmi zjednoduší a zpříjemní manipulaci s datovou vrstvou vyvíjené aplikace.

6.1.3 Ext JS

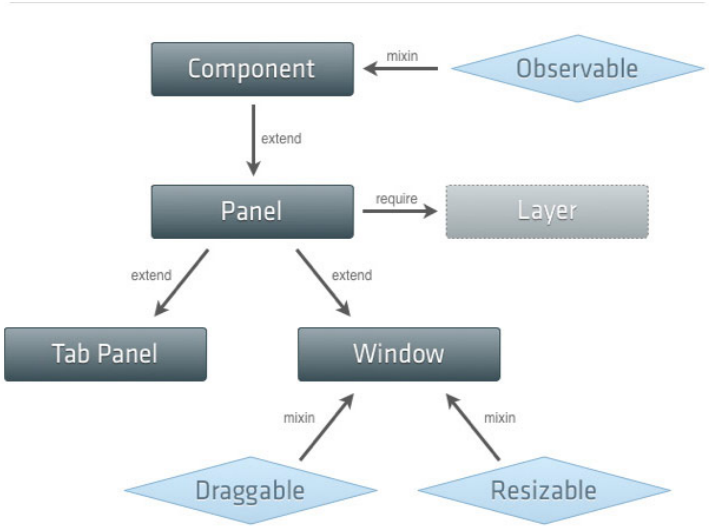
Ext JS je JavaScriptový framework, sloužící k tvorbě webových aplikací. Používá technologie jako je AJAX, DOM a DHTML. Ext JS obsahuje široký katalog plně uživatelsky rozšiřitelných tzv. widgetů, které se dají využít k postavení dobře vypadající a velmi výkonné, čistě javascriptové prezentační vrstvy aplikace.

Jedním z velmi silných znaků tohoto frameworku je jeho podpora napříč webovými prohlížeči[9]. Díky této vlastnosti se tak při vývoji vyhneme nezanedbatelným problémům při zajišťování požadované funkčnosti vyvíjené aplikace na všech požadovaných prohlížečích.

Ext JS také podporuje oddělení vizuálních komponent od datových struktur, v jejichž rámci nabízí širokou škálu prostředků pro jejich správu a manipulaci s daty na straně prezentační vrstvy. Framework dokonce podporuje implementaci návrhového vzoru MVC[1] a v novějších verzích MVVM[2].

Obecně framework Ext JS se také vyznačuje silně objektově orientovaným přístupem. V rámci tohoto frameworku je snaha skloubit to nejlepší z flexibility vyplývající ze

[1] Model-view-controller
[2] Model–view–viewmodel



Obrázek 6.1. Systém tříd v Ext JS 4[10]

samotné podstaty programovacího jazyku javascript a s, na třídy orientovaným, OOP přístupem.[10]

Další silnou výhodou je efektivní práce s vizuálními tématy, jak se ukáže při návrhu grafické podoby aplikace.

Samozřejmě Ext JS má i své nevýhody a záludnosti. Asi nejpatrnější nevýhodou je relativně velká velikost frameworku a delší čas pro první načtení aplikací, jež tento framework využívají. Další z často uváděných potíží s ní, je její relativně pomalá učební křivka[11] , kdy tak trvá dost dlouho než se s ní vývojář naučí řádně pracovat.

6.1.4 Maven

Pro pomoc při vývoji aplikace HR Cockpit byl zvolen nástroj Maven. Maven slouží k usnadnění správy vyvíjeného projektu, udržování přehledu o závislostech projektu a jeho verzí. Neocenitelný je samozřejmě při samotném sestavování projektu. Maven uvádí následujících pět základních oblastí na které se zaměřuje[12].

- Usnadnění procesu sestavování.
- Poskytování jednotného systém sestavení.
- Poskytování kvalitních informací o projektu.
- Poskytování rad pro dodržování osvědčených postupů.
- Poskytnutí transparentního přidávání nových funkcí.

6.1.5 Nástroje BC

Vzhledem k tomu, že jedním ze základních nefunkčních požadavků bylo vytvořit aplikaci HR Cockpit podle standardů a postupů společnosti Baader Computer, je nutné využívat i určité specifické nástroje a knihovny této společnosti. V prvé řadě se jedná o rozšíření a určité nástavby nad některými veřejně dostupnými knihovnami například nad frameworkem Spring, Hibernate či Ext JS.

Mezi další, při vývoji hojně používaným nástroji poskytnutými společností Baader Computer, jsou různé generátory metadat, které tak značně usnadňují vývoj a ubírají určité množství vývojového času stráveného při psaní různých metadat.

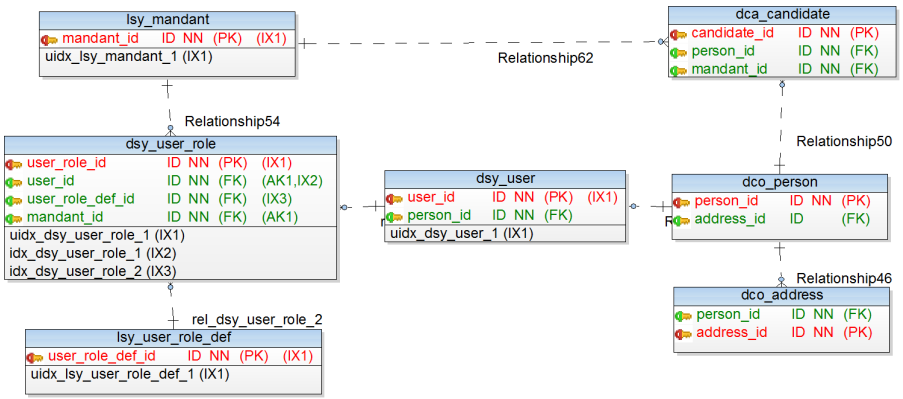
6.2 Databázový model

Důležitým aspektem návrhu této aplikace je navržení struktury databáze, která bude sloužit k ukládání veškerých potřebných dat pro její chod. Relační model pokrývající veškeré požadavky na vlastnosti aplikace, specifikované v předchozí kapitole, se nachází v příloze. Vzhledem k určité rozsáhlosti navrhovaného databázového modelu byl za účelem lepší grafické přehlednosti pro popsání v této kapitole rozložen do několika částí, které jsou ale ve skutečnosti součástí jednoho celku, a také bylo nutno omezit výčet jednotlivých atributů navrhovaných tabulek pouze na primární a cizí klíče.

U zde navrhovaného datového modelu je nutné, také upozornit na jeho na první pohled možná až zbytečnou složitost. Pokud se vezmou v potaz jen současně definované funkční požadavky, tak je téměř jisté, že by bylo možné vytvořit databázové schéma podstatně jednoduší a s menším počtem entit. Důvod, proč je i přes tuto skutečnost navrhováno schéma o této složitosti, je jednoduchý. V rámci původní analýzy bylo definováno a zároveň následně i zahrnuto mezi požadavky, že aplikace musí podporovat plánovaná rozšíření. Z tohoto důvodu bylo databázové schéma navrhnuto už s očekáváním těchto plánovaných rozšíření, aby se tak v budoucnu zabránilo zbytečným problémům se změnou datového schématu a případných změn v již funkční aplikaci.

6.2.1 Systém

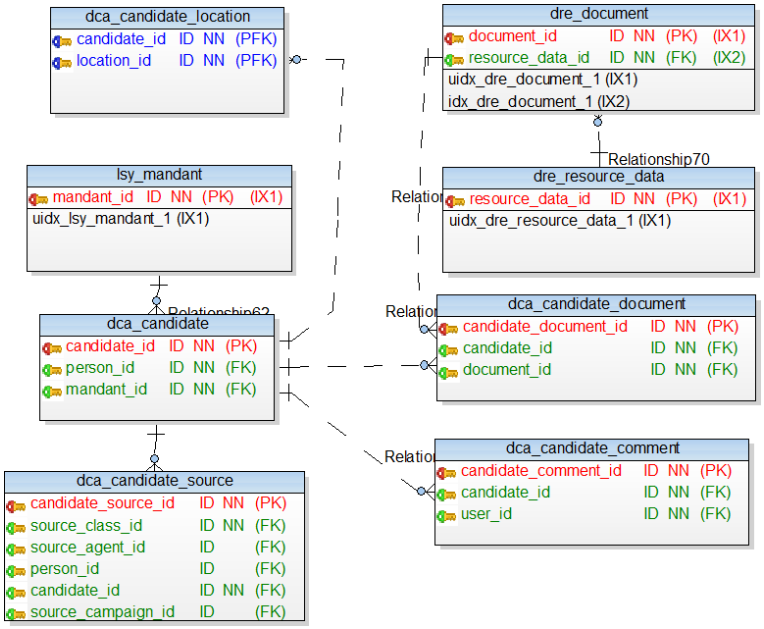
Základní část databázového schéma, pokrývající systémové požadavky na uchovávání dat o uživatelích, jejich rolích v rámci systému a samozřejmě informace o kontextech tzv. mandantech, ve kterých aplikace pracuje a pro které je nutné určitá další data v rámci aplikace důsledně oddělovat. Díky takto navrženému schématu, je tak splněn jeden z požadavků na možnost figurování uživatele v rámci více různých mandantů s odlišnými uživatelskými rolemi. Ze schématu je také zřejmé, že jsou odděleny veškeré informace o kandidaturách stejných osob pro každého mandanta zvlášť.



Obrázek 6.2. Databázové schéma - systém

6.2.2 Kandidáti

Na tomto schématu je zachycena struktura uchovávání dat jednotlivých kandidátů. V rámci těchto uvedených tabulek jsou ukládána veškerá data, která jsou specifikací vyžadována pro uchovávání informací jednotlivých kandidatur. Personální informace, jako jsou jméno atp. jsou pak ukládány v tabulce sloužící globálně pro evidování osob viz předchozí systémový diagram.



Obrázek 6.3. Databázové schéma - kandidáti

6.2.3 Zdroje kandidátů

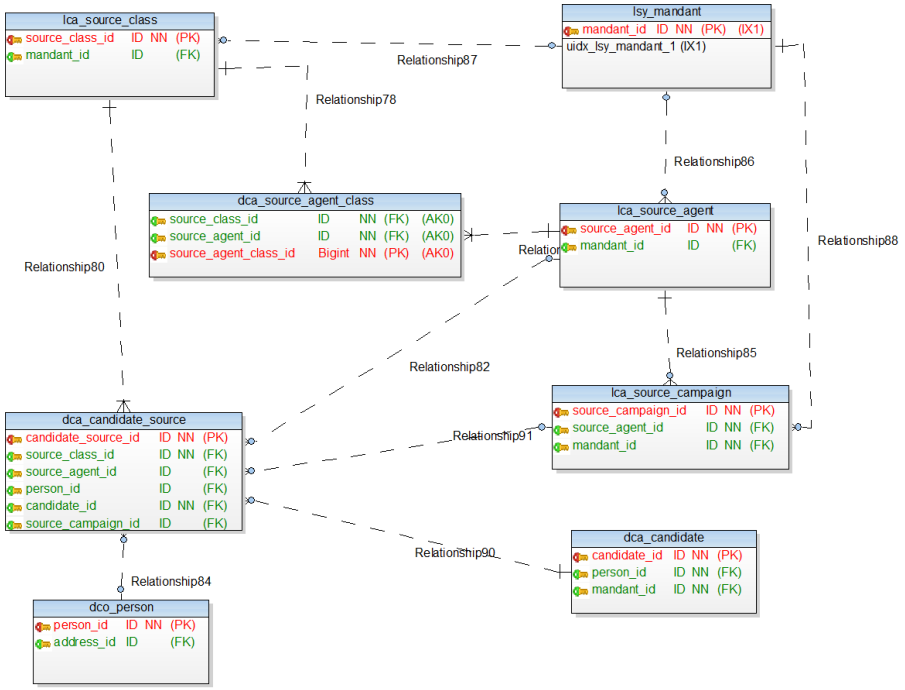
Schéma sloužící k uchování informací o zdroji jednotlivých kandidátů. Jak definuje specifikace, zdroje kandidátů se musí dát řadit do různých kategorií s tím, že dále v rámci určitých zdrojů musí být možno evidovat i časově omezenou kampaň s vlastními atributy. Další podmínkou specifikace je, že zdrojem kandidáta může být i přímo konkrétní osoba evidovaná v aplikaci, a tudíž systém musí udržovat přímou referenci mezi doporučovaným kandidátem a osobou, která ji doporučila. Na základě těchto požadavků, tak bylo vytvořeno toto schéma, které umožňuje všechny definované vlastnosti splnit a v rámci kontextu mandantů uchovávat veškerá požadované data.

6.2.4 Dovednosti

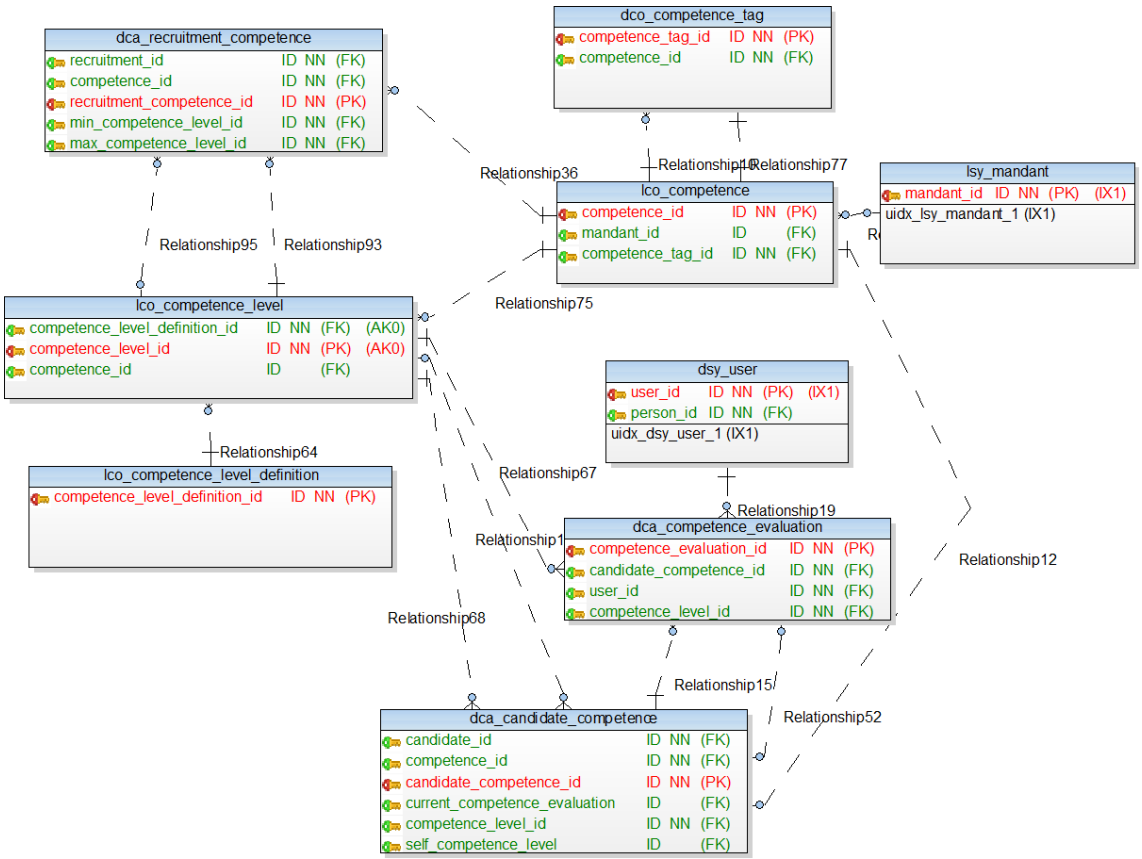
Tato část databázového schéma znázorňuje způsob uchovávání informací o dovednostech v systému a také ukazuje relace s entitami kandidátů, výběrových řízení a hodnotitelů. Jak je na schématu vidět, kandidáti ovládají své dovednosti na určité úrovni dané hodnocením, které do systému vložil uživatel - nejčastěji právě hodnotitel. Dovednosti se ale vztahují i k výběrovým řízení, kde specifikují požadované dovednosti a jejich minimální úroveň ovládání kandidáta. V rámci splnění definovaných požadavků může mít každá dovednost svůj specifický tag, pod kterým ji je možné systémem identifikovat. Každá dovednost pak má svůj vlastní systém hodnocení a každé dovednosti je tak možné definovat vlastní hodnotící stupnici, sloužící k definování zvládnutí dané dovednosti kandidáty.

6.2.5 Výběrová řízení

Nejkomplexnější část databázového schématu zachycuje datovou strukturu samotných výběrových řízení. Na schématu je vidět členění výběrových řízení do jednotlivých fází. Relace mezi fázemi a jednotlivými účastníky výběrového řízení dobře znázorňují samotný proces výběrového řízení. Schéma zachycuje způsob definování jednotlivých hodnotitelů a případných dovedností, jenž jsou v rámci fáze oprávněni účastnícím se kandidátům, dané fáze ohodnotit. Schéma taky zachycuje způsob rozhodnutí o úspě-

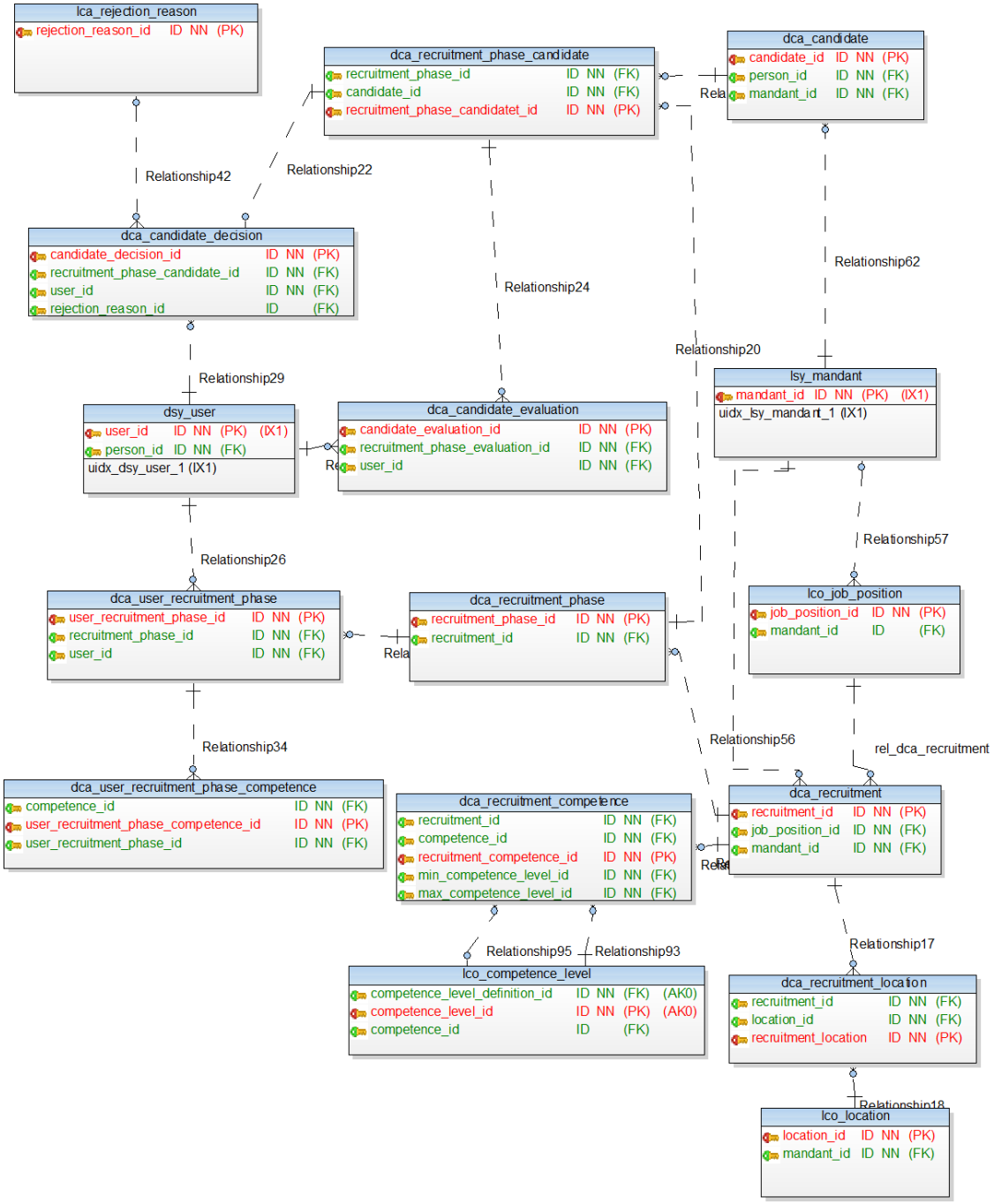


Obrázek 6.4. Databázové schéma -zdroje kandidátů



Obrázek 6.5. Databázové schéma - dovednosti

chu kandidáta v dané fázi a případné možnosti přímo evidovat důvody, proč kandidát v dané fázi neuspěl.



Obrázek 6.6. Databázové schéma - výběrová řízení

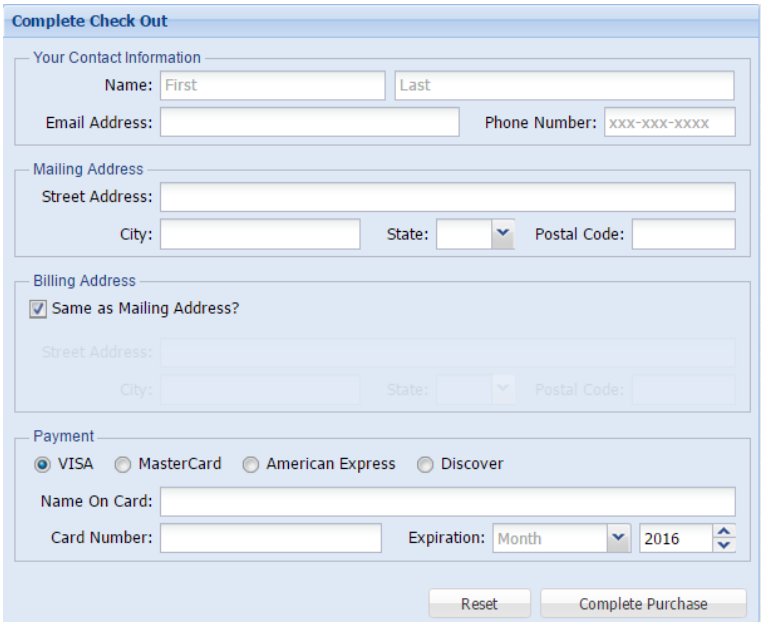
6.3 Grafický návrh

V rámci návrhu této aplikace je nezbytné stanovit si cílovou grafickou podobu výsledné aplikace Dobrý návrh uživatelského rozhraní může být často jednoduše upozaděn oproti jiným na první pohled možná důležitějším částem systému. Přesto je ale nutno brát v potaz, že samotné uživatelské rozhraní je hlavní částí aplikace, se kterou se koncový uživatel nejčastěji setkává. Jedná se tak o přímý komunikační kanál aplikace s uživatelem nebo v tomto případě řečeno zákazníkem, a tudíž má přímý vliv na celkový úspěch aplikace a její případnou ziskovost a životnost. Samotné rozdělení obra-

zovek může ovlivnit uživatele mnoha způsoby, kdy špatná organizace může uživatelům značně zkomplikovat práci a v nejhorším případě je dokonce od užívaní dané aplikace zcela odradit. [13]

6.3.1 Témata frameworku Ext JS

Vzhled vyvíjené aplikace je přímo ovlivněn zvoleným frameworkem pro vytvoření prezentační vrstvy. Naštěstí framework umožňuje tvorbu velmi kvalitního, čistého a profesionálně vypadajícího, uživatelského rozhraní.[10]. Což je také zároveň i jedním z důvodů proč je vlastně tento framework v této práci využit.



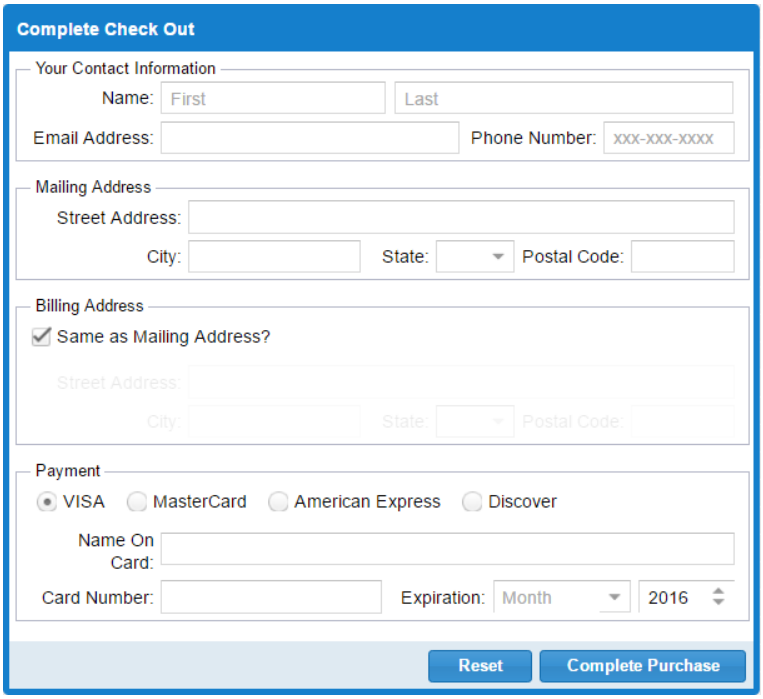
Obrázek 6.7. Téma classic

Ext JS nabízí již sám o sobě několik základních grafických témat, které mohou být přímo využity při samotné tvorbě aplikace, aniž by se při samotném vývoji musel brát zřetel na většinu grafických elementů aplikace. Tento postup samozřejmě není ve výsledku většinou žádoucí, ačkoliv jsou tedy jednotlivá témata skutečně zdařilá a daří se jim tak plnit, již dodavatelem frameworku deklarovanou čistotu návrhu a profesionální dojem výsledných aplikací. Je také dobré, že framework umožňuje kompletní přizpůsobení použitého tématu, kdy je možné vytvořit si svůj vlastní, vzhledem nejlépe odpovídající cílové aplikaci.

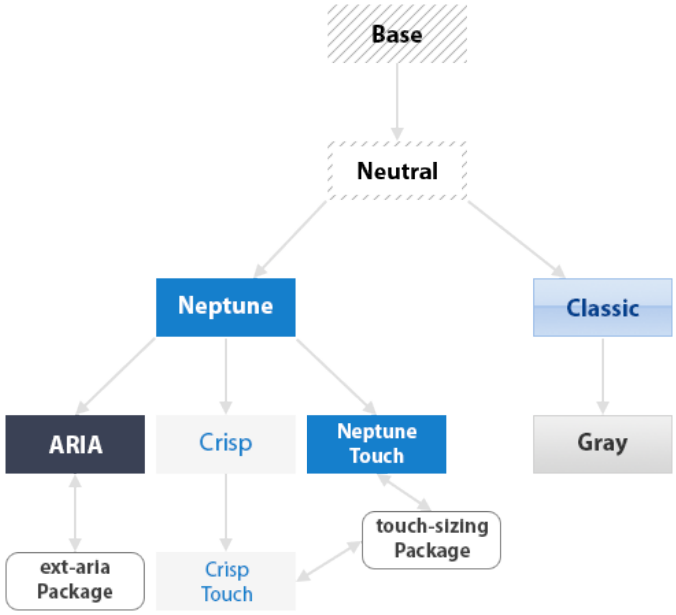
Vytvoření takového tématu je pak díky vlastnostem frameworku velmi jednoduché. Všechna témata, používaná pro definování vzhledu jednotlivých komponent frameworku Ext JS, jsou totiž součástí většího hierarchického systému tématických balíčků. Díky tomu musí všechna nová témata dědit již od nějakého existujícího rodičovského balíčku. Díky tomu je pak velmi jednoduché vybrat si některý existující balíček, jenž nejvíce odpovídá zamýšleným představám o výsledném vzhledu aplikace a jen si ho dále v novém, poděděném balíčku vhodně upravit.

Samotná úprava tématu následně probíhá změnou zdrojových souborů daného tématu, jenž pomocí jazyku SASS[1], definují vzhled jednotlivých komponent. Díky vhodné organizaci těchto zdrojových souborů, je tak jednoduché provádět relativně významné

[1] `http://sass-lang.com/`



Obrázek 6.8. Téma neptun



Obrázek 6.9. Hierarchie tématický balíčků [10]

změny jen pomocí minimálních změn v kódu. Pomocí prosté editace globálních proměnných, je tak například možné změnit kompletně hlavní barvu tématu.

```
$base-color: #317040 !default;
```

Samozřejmě v rámci editace tématu je možné provádět i složitější změny a vytvářet kompletně vlastní pravidla pro používané komponenty.

Výsledné editované zdrojové kódy tématického balíčku je pak možné jednoduše nechat zpracovat pomocí nástroje Sencha CMD, jenž slouží ke kompilaci Ext JS a Sencha Touch[1], aplikací. Výsledkem je sada css souborů a složek, se všemi potřebnými zdrojovými soubory, které jsou nezbytné pro použití daného tématu vyvíjenou aplikací.

Pro aplikaci vyvíjenou v rámci této práce, tak byl zvolen jako základní tématický balíček tzv. classic. Tento balíček nabízí přehledný a konzervativní grafický návrh, který se dle všeho dobře hodí k celkovému zamýšlenému rázu aplikace. Na základě tohoto balíčku bylo pak samozřejmě postaveno specifické téma používané koncovou aplikací, ale to již nemuselo přinést žádné razantní změny, protože již v základu balíček classic dobře plní představy o výsledné grafické podobě vyvíjené aplikace.

6.3.2 Uživatelské rozhraní

Ačkoli sám framework Ext JS, použitý pro vytvoření prezentační vrstvy vyvíjené aplikace, poskytuje solidní základní předpoklad pro vytvoření graficky a uživatelsky přívětivé aplikace, tak to ve výsledku bohužel samo o sobě nemůže stačit. Při samotném vývoji by i se správnou sadou grafických prvků a komponent poskytovaných frameworkem bylo jednoduché vytvořit uživatelsky těžko ovladatelné rozhraní.

Pro doplnění pouze slovně psané dokumentace je vhodné vytvořit nějakou formu vizualizace navrhovaného systému. To umožní samotné ujasnění o navrhovaných případech užití a dovolí předem připravit kvalitní návrh rozdělení těchto scénářů na jednotlivé obrazovky. Velkou výhodou tohoto postupu je také možnost konzultovat vývojářem zamýšlenou podobu systému se zákazníkem, a tak případně předejít nákladům a problémům při změnách již implementovaných obrazovek.

V rámci této práce jsem se rozhodl vytvořit několik mockupů aplikace HR Cockpit. Tyto tzv. mockupy slouží k popisu uživatelského rozhraní a umožňují definovat důležité stavy aplikace. Nejsou sice určeny k přímému uživatelskému testování, ale přesto je lze použít k případné diskuzi se zákazníkem.[14] V rámci těchto modelů je tak možné ujasnit si celkové rozložení aplikace a zaměřit se na několik případů užití, které by, ze psané dokumentace, nemusely být úplně jasné. Zároveň také platí, že nebyla vyložená nutnost navrhovat tyto modely pro všechny obrazovky, jelikož některé jsou jasné i přímo ze psaného popisu.

6.3.3 Rozložení obrazovky

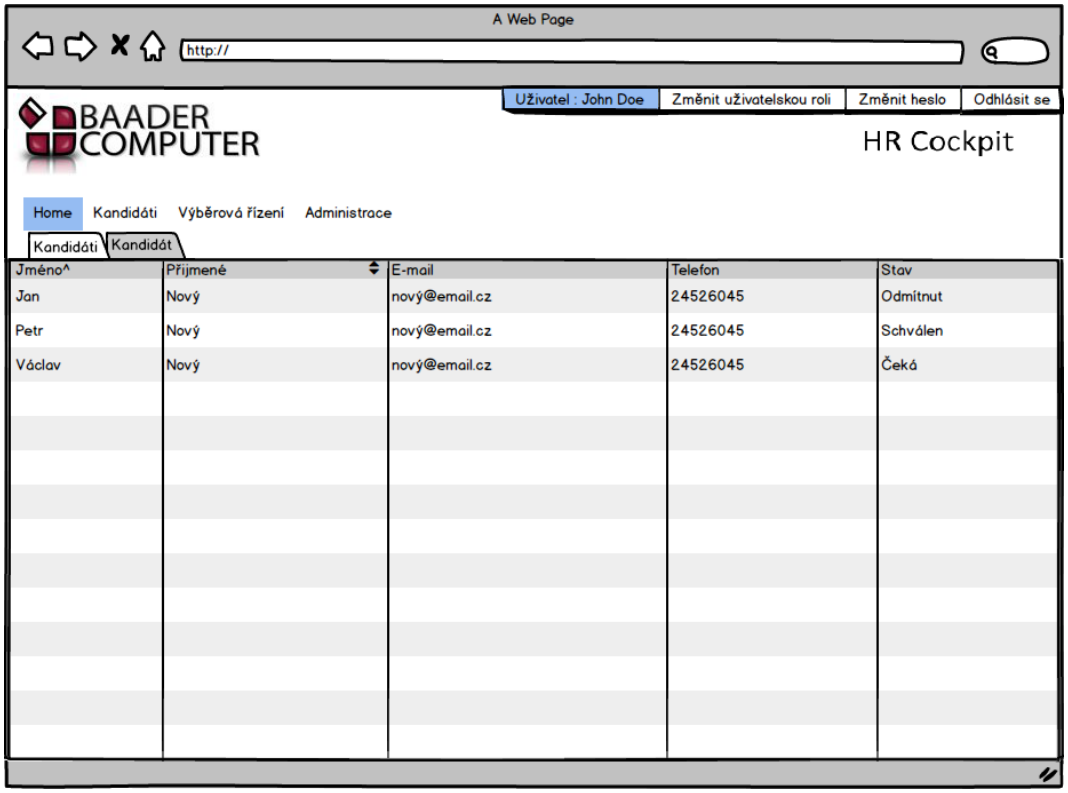
Základní rozložení obrazovky je jednoduché. V horním pravém rohu se nacházejí základní uživatelské ovládací prvky, které jsou jinak nezávislé na ostatním zobrazovaném obsahu. Tento ovládací panel, tak umožňuje, aby měl uživatel vždy na dosah základní funkce aplikace týkající se právě přihlášeného uživatele. Mezi tyto funkce patří

- Změna hesla
- Změna aktivní uživatelské role
- Odhlášení z aplikace

Pod tímto ovládacím panel se pak nachází dvě obrázková loga. Na levé straně je vidět logo aktuálně přihlášeného mandanta a na pravé je statické logo samotné aplikace.

Pod těmito grafickými prvky je již zobrazeno hlavní menu celé aplikace, které uživateli umožňuje přepínat se mezi funkcemi aplikace. Uživatelem otevřený obsah je poté otevírán v oblasti pod samotným menu. Jedním, z uživatelského hlediska, důležitým

[1] `https://www.sencha.com/products/touch/`



Obrázek 6.10. Mockup - Základní rozložení obrazovky aplikace

prvkem aplikace je možnost mít otevřeno aktivně několik obrazovek ve volně přepínatelných panelech. Díky této vlastnosti, tak může mít uživatel možnost volně přepínat mezi profily dvou kandidátů atp.

6.3.4 Detail kandidáta

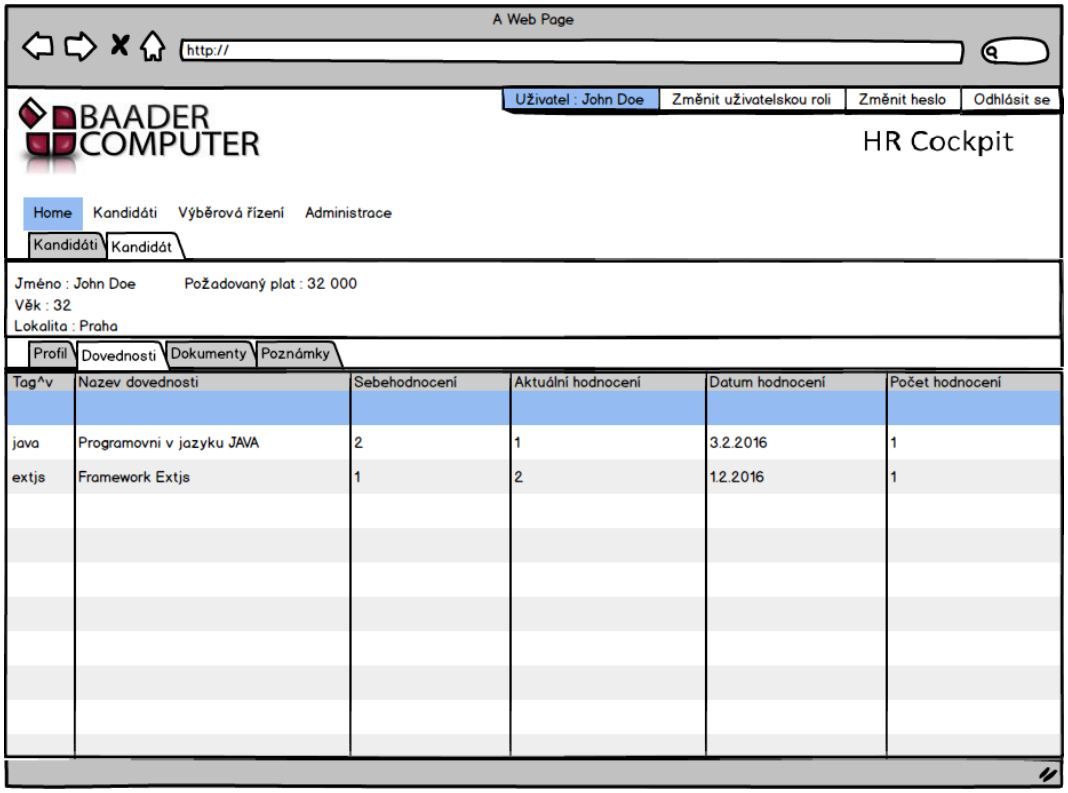
V rámci návrhu obrazovky obsahující data o kandidátovi a naplňující případy užití kolem této kandidátské entity bylo nejdůležitější určit způsob práce s definovanými dovednostmi. Jedním z požadavků na aplikaci byl jednoduchý a rychlý způsob zadávání nových dovedností využívající principu tzv. tagů, který by dále umožňoval automatické vytváření nových dovedností v případě, že by uživatele nezadal žádný již existující platný tag.

V rámci postupu návrhu konečného řešení, jež by bylo z uživatelského hlediska nejlepší, se pracovalo s několika možnými variantami.

- Modifikace dovednosti v novém dialogovém okně
- Implementace intuitivního našeptávače
- Přímá editace zobrazeného záznamu v gridu dovedností

První varianta se jeví jako nejpřímější metoda editace záznamu a za normálních okolností je v rámci aplikace často používána. Z gridu, zobrazující seznam záznamů je jeden z nich vybrán a dále editován ve zvláštní dialogu, obsahujícím formulářová pole pro jeho editaci. Každopádně v tomto případě se tento postup nejeví vhodný, jelikož očividně nesplňuje požadavek na rychlost editace záznamu.

Varianta našeptávače se proto dlouho dobu jevila jako ideální řešení problematiky a bylo podporováno i zákazníkem. Z tohoto důvodu bylo toto řešení i dlouhou dobu vedeno v rámci návrhové dokumentace. Na problémy z tou variantou řešení, upozornily



Obrázek 6.11. Mockup - Kandidátovy dovednosti

až snahy zachytit tento přístup pomocí právě mockupu. V původním návrhu by se zadal do našeptávajícího políčka text tagu a za něj by se dále odděleny dvojtečkou zapisovali další parametry dovednosti.

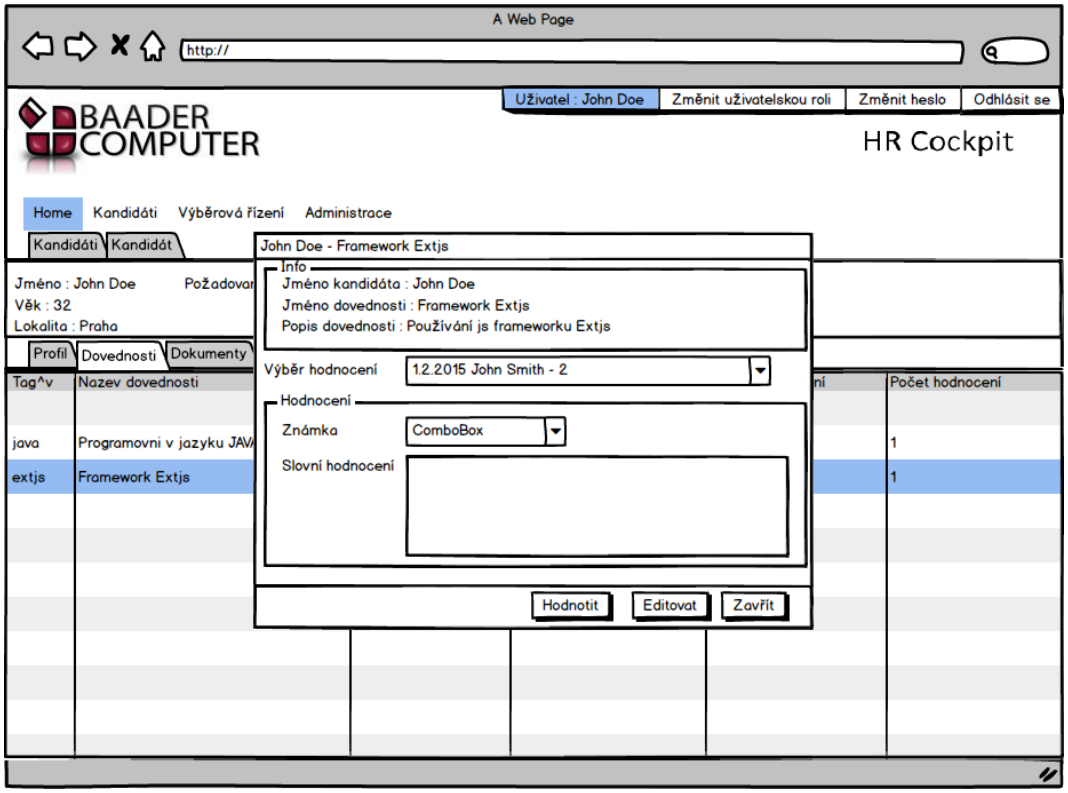
Například zápis : `java : 2` by znamenal, že kandidát ovládá dovednost s tagem `java` na úrovni 2. Problém by nastal v případě, kdy by tato dovednost v systému zatím neexistovala a aplikace by musela pouze na základě těchto dvou údajů vytvořit kompletně nový databázový záznam o této dovednosti. Tím pádem by vznikla nová dovednost bez jména definovaná pouze pomocí tagu. Případně by se musel do našeptávače za dvojtečku doplnit i název dovednosti. Takovéto řešení by možná bylo ještě použitelné, ale už dobře poukazuje na celkové nedostatky tohoto konceptu jako celku. Jednoduše je problém vytvářet nový záznam a oddělovat jednotlivé parametry tohoto záznamu pouze pomocí oddělovacího znaku a zároveň udržet přehlednost celé operace. Tento problém je pak ještě viditelnější v případě, kdy je potřeba zadat dodatečných parametrů více, jako například při zadávání dovedností v rámci výběrového řízení a jeho požadavků na kandidáty.

Z těchto důvodů bylo nakonec zvolena třetí varianta řešení této problematiky. Uživatel tak může editovat zobrazený záznam přímo v seznamu dovedností. Pokud uživatel označí některý ze zobrazených záznamů, zobrazí se řádkový editor, který uživateli umožní přímo modifikovat hodnoty. Díky tomu je zaručeno, že uživatel ví přesně jakou hodnotu modifikuje a zároveň je splněn požadavek na rychlost celé operace, jelikož není potřeba celý záznam otevírat v novém okně.

6.3.5 Hodnocení dovednosti

Důležitou součástí aplikace je funkce hodnocení dovedností. Za tímto účelem bylo navrženo dialogové okno sloužící k zobrazení všech hodnocení dané kandidátovy dovednosti

a k jejich případné modifikaci. Pro samotné hodnocení je nutné zobrazit základní informace o kandidátovi k čemuž slouží informativní box v horní části dialogového okna. Jistým problémem pak byl způsob zobrazení historie hodnocení dané dovednosti. Snahou bylo vyhnout se nutnosti zobrazovat všechna předešlá hodnocení v seznamu, ze kterého by si uživatel musel vybrat, se kterým plánuje dále pracovat a případně ho opět otevírat v dalším dialogovém okně. Také se vzhledem k možné délce slovního hodnocení nejevilo jako optimální zobrazovat kompletní hodnocení v jednom řádku.



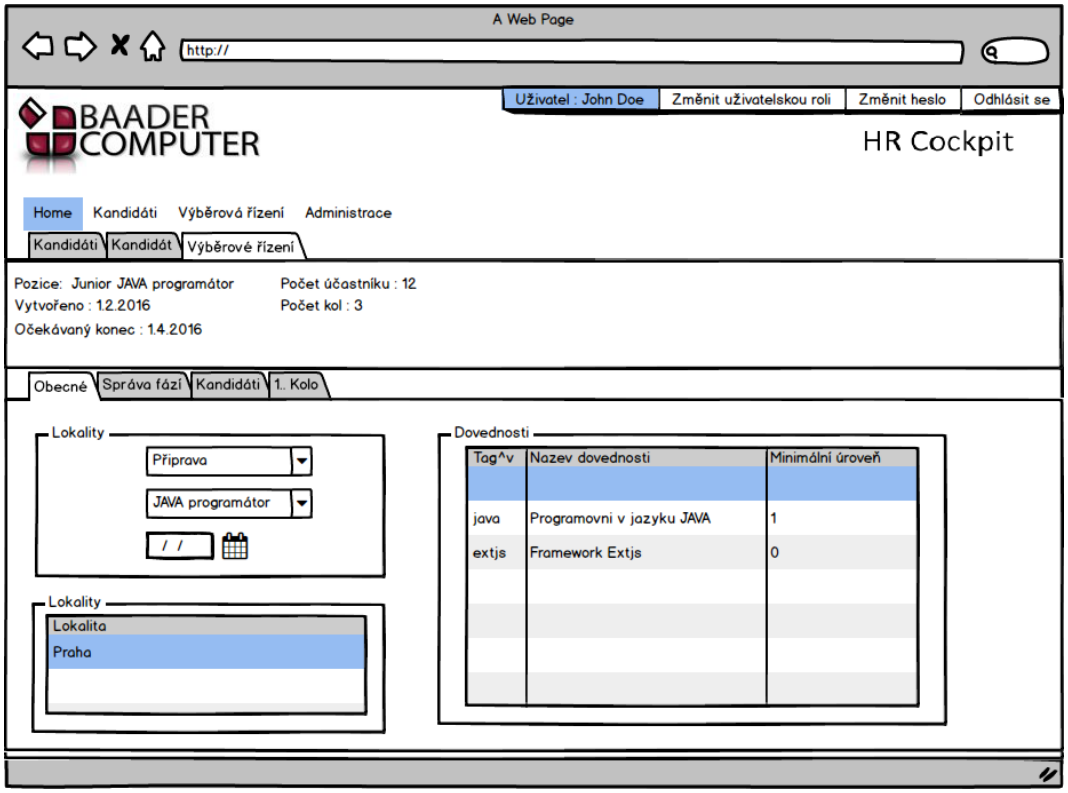
Obrázek 6.12. Mockup - Dialog hodnocení kandidátovy dovednosti

Navrhované řešení, tak používá jednoduchý kombinovaný seznam ve kterém je stručně uveden přehled daného hodnocení a pokud v tomto seznamu uživatel některé hodnocení vybere, načte se do - pod ním připraveného formuláře, kde ho může uživatel případně modifikovat.

6.3.6 Výběrové řízení

Detail výběrového řízení se drží podobného rozložení jako detail kandidáta. V horní části obrazovky je vidět souhrn informací o daném výběrovém řízení a pod ním jsou jednotlivé modifikovatelné údaje rozděleny do několika přepínatelných panelů. Největším problémem při návrhu obrazovky výběrového řízení se stalo zobrazování fází výběrového řízení. Při původním návrhu případů užití se zpočátku počítalo s tím, že bude mít i tato entita fáze podobnou detailní obrazovku jako právě kandidát či výběrové řízení. Při jejím navrhování a pokusech vytvořit vizuální model se ale došlo ke zjištění, že tento přístup možná nebude nutný a ani nejvhodnější.

Původní předpoklad počítal s variantou, kdy by v jednom z panelů výběrového řízení byl seznam všech jeho fází, které by při uživatelově volbě zobrazily v novém panelu detail fáze. Výsledek pokusů o tvorbu vizualizace tohoto panelu, ale vedlo ke zjištění,



Obrázek 6.13. Mockup - Detail výběrového řízení

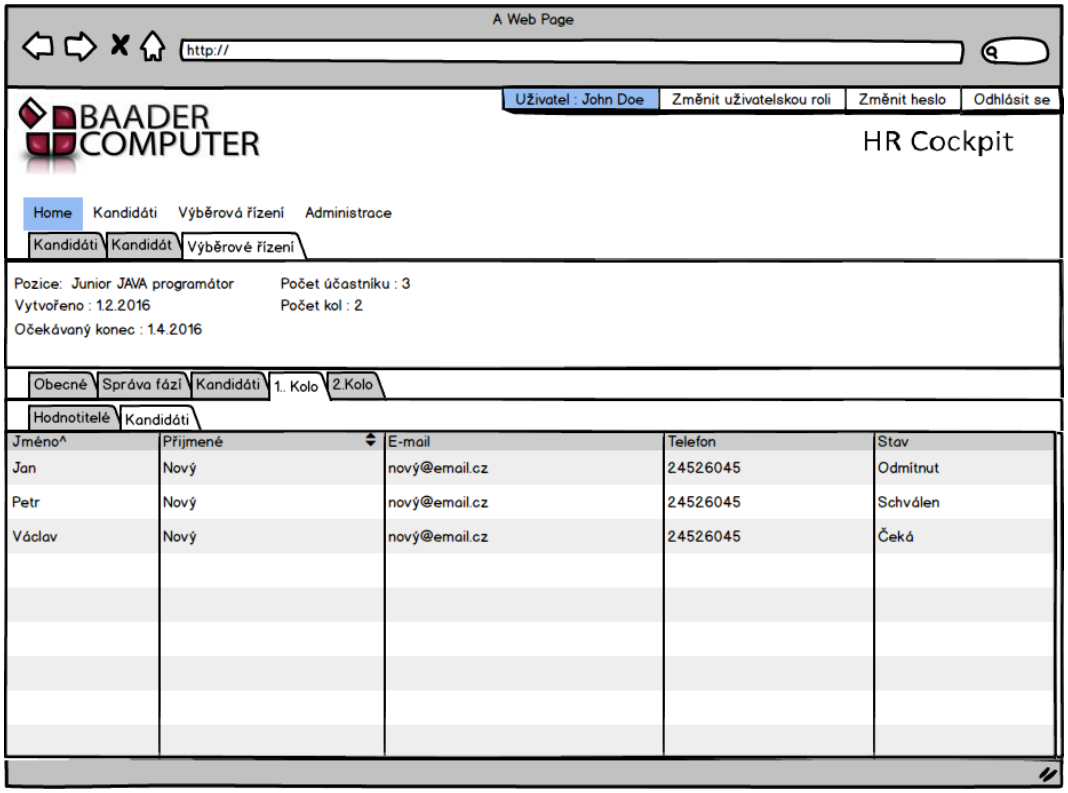
že samotná entita fáze nemá příliš parametrů na to, aby takovýto detail potřebovala. Z tohoto důvodu jsou panely jednotlivých fází zobrazeny přímo v detailu výběrového řízení. Součástí tohoto detailu je sice stále i panel se seznamem fází, ve kterých je možné provádět základní správu těchto fází, ale samotné detaily jednotlivých fázích už jsou připraveny přímo v detailu výběrového řízení. Díky tomu je dosaženo větší přehlednosti o fázích a většího uživatelského komfortu, kdy odpadá nutnost otevírat nový panel pro každou fázi. Výhodou také je, že pokud se nachází v tzv. detailu fáze má pořád možnost vidět i základní informace o výběrovém řízení, do kterého daná fáze spadá.

6.3.7 Fáze výběrového řízení

Jak už bylo zmíněno, detail výběrového řízení je vnořen do detailu samotného rodičovského výběrového řízení. Jeho samotným obsahem pak jsou jen dva panely, ve kterém jsou zobrazeni jeho hodnotitelé a kandidáti.

6.3.8 Hodnocení kandidáta

Dialog sloužící k hodnocení kandidáta v rámci dané fáze. V rámci tohoto dialogu musí mít uživatel možnost navrhnout svůj názor na kandidátův postup do dalšího kola výběrového řízení. Určitou komplikaci v rámci toho dialogu působí požadavek na aplikaci, který vyžaduje, aby v rámci tohoto případu užití bylo zobrazeno větší množství informací. Kvůli tomu musí být celé dialogové okno relativně velké a aby i přes to byla zachována přehlednost je nutné, aby bylo celé okno vhodně rozčleněno. V horní části okna se nacházejí základní údaje o kandidátovi - jako je jeho jméno a jeho aktuální stav v rámci výběrového řízení. V levé části jsou pak editovatelná pole pro zadání samotného uživatelova hodnocení kandidáta. V pravé části je pak přepínatelný panel, který slouží k zobrazení všech dále požadovaných údajů. Díky této vlastnosti tohoto panelu je tak



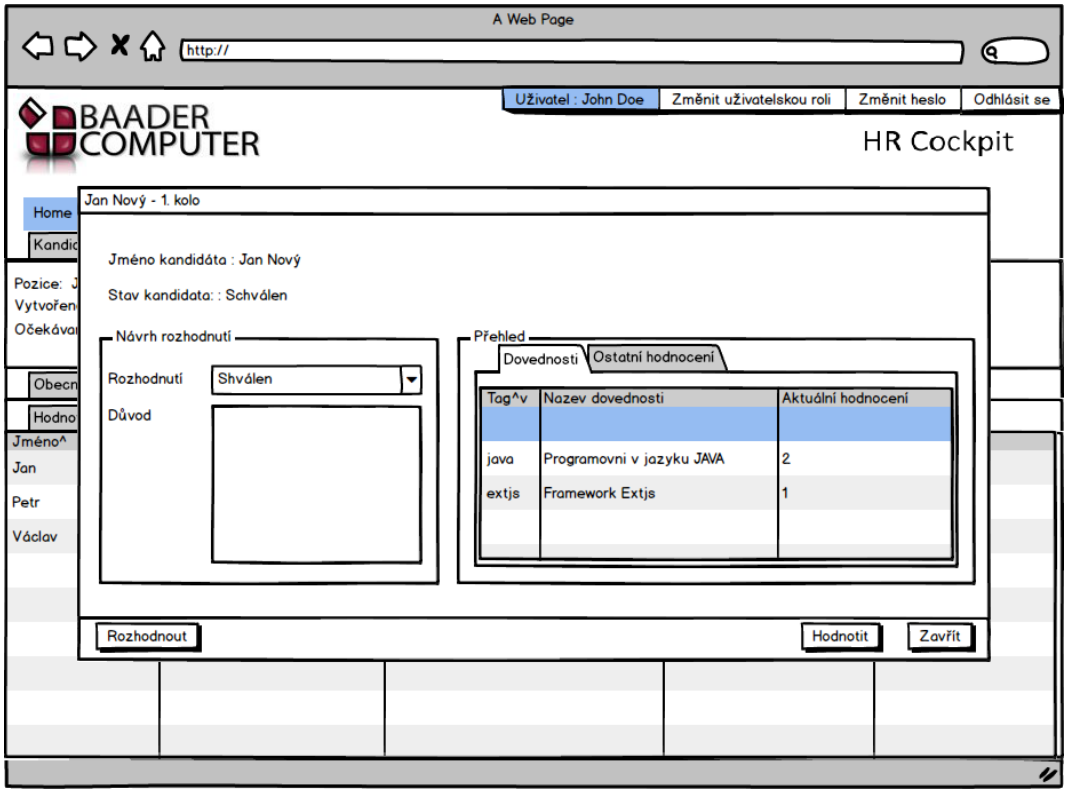
Obrázek 6.14. Mockup - Detail fáze

dosaženo alespoň částečné redukce vyžadovaného zobrazovacího prostoru. V těchto přepínatelných panelech jsou pak dále zobrazeny souhrnné informace o názorech ostatních hodnotitelů na kandidáta a informace o dovednostech kandidáta v kontextu výběrového řízení, kdy je zobrazeno, jak kandidát splňuje požadavky daného výběrového řízení.

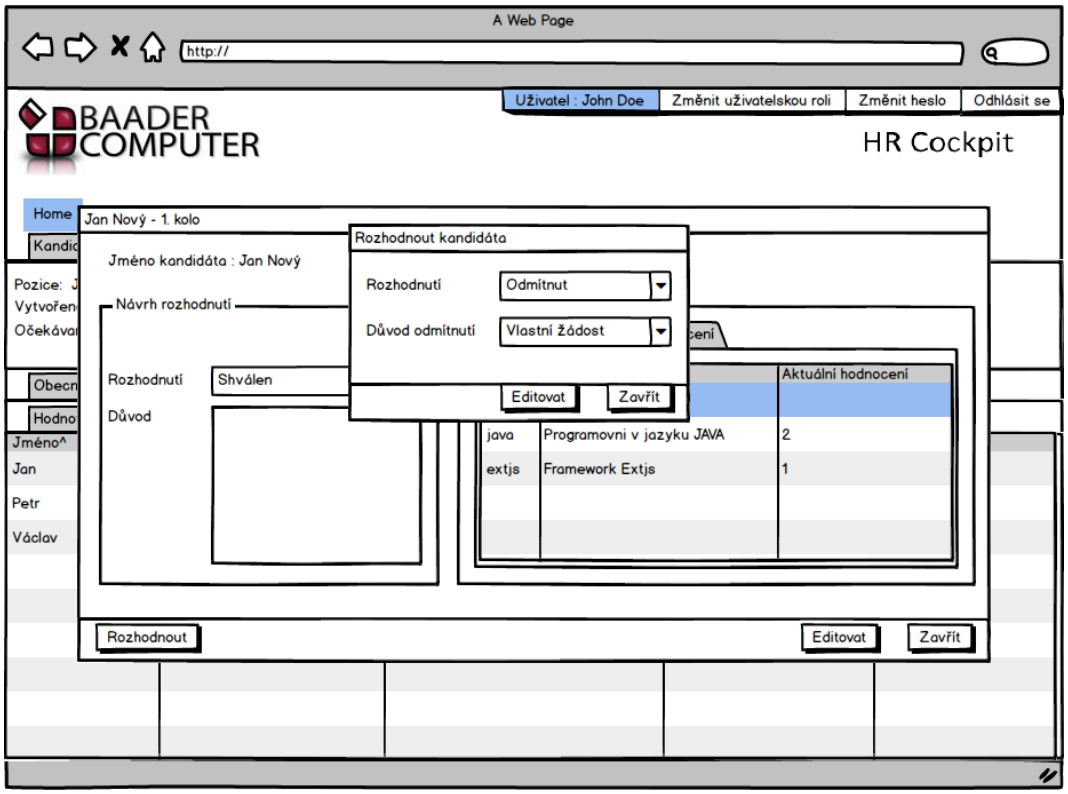
Problematická tak zůstává jen skutečnost, že je nutné v rámci jednoho řádku seznamu hodnocení ostatních hodnotitelů zobrazit i slovní hodnocení, které může svou délkou překročit rozsah jednoho řádku v seznamu. Řešením je využít tzv. rozšíření řádku, kdy při uživatelově akci je řádek rozšířen a jsou zobrazeny dodatečné informace. Díky tomu je v samotném řádku zobrazen jen hodnotitelův návrh na kandidátův postup či nepostup do další fáze a slovní zdůvodnění je zobrazeno až při rozevření řádku seznamu.

6.3.9 Rozhodnout kandidáta

Jedná se o jednoduchý dialog sloužící k finálnímu rozhodnutí o kandidátově osudu v dané fázi výběrového řízení. Podle požadavků by měl uživatel před učiněním toho rozhodnutí opět vidět stejné informace, jako při samotném kandidátově hodnocení. Aby se předešlo zbytečné nutnosti vytvářet podobný dialog, jako pro hodnocení kandidáta, je vymezena nutnost tento jednoduchý dialog otevírat z předchozího dialogu, sloužícího k samotnému hodnocení. Uživatel tak má před samotným rozhodnutím možnost prohlédnout si všechny potřebné podklady, které pro své rozhodnutí potřebuje.



Obrázek 6.15. Mockup - Hodnocení kandidáta



Obrázek 6.16. Mockup - Rozhodnout kandidáta

Kapitola 7
Implementace

V rámci této kapitoly bych rád trochu přiblížil postup samotné implementace aplikace HR Cockpit. Není příliš praktické zacházet do úplných detailů samotné aplikace a popisovat každou třídu výsledné aplikace, a tak bych se raději jen zaměřil na implementační principy a typové ukázky vypracovaného řešení.

7.1 Použité nástroje

Samotná implementace by byla jen těžko uskutečnitelná, kdyby se nevyužilo vhodných nástrojů. Následuje tedy jen několik hlavních nástrojů, které byly při samotné implementaci použity

- Eclipse IDE [1]
- Spket IDE [2]
- pgAdmin [3]
- Toad Data Modeler[4]
- Subversion [5]

7.2 Vývojové prostředí

Kvůli vývoji aplikace je třeba rozlišovat běh v několika různých prostředích. V rámci prostředí musí být všechny zdroje aplikace včetně databáze od sebe náležitě odděleny. V rámci vývoje této verze aplikace se používalo zatím následující prostředí.

- Vývojové
- Testovací
- Demonstrační

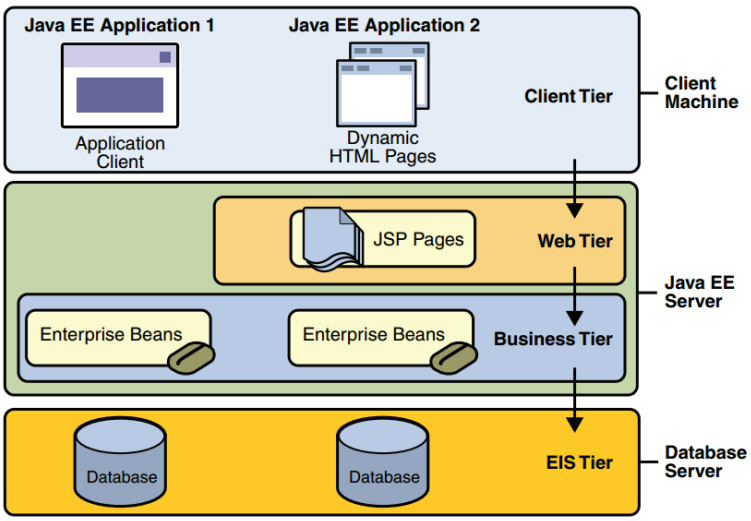
Testovacímu prostředí je pak dále věnováno více prostoru ve specifické kapitole. Vývojové prostředí slouží, jak název sám napovídá k vývoji. Takové prostředí slouží k oddělení vývojáře a jeho vyvíjené verze aplikace od potenciálně cenných dat, aby nehrozilo nebezpečí jejich modifikace nebo zničení. Vývojář je tak také v bezpečí před případnými chybami vývojové verze aplikace a může ji volně sám průběžně testovat.

[1] https://eclipse.org/
[2] http://www.spket.com/
[3] www.pgadmin.org/
[4] http://www.toad-data-modeler.com/
[5] https://subversion.apache.org/

7.3 Architektura

Vyvíjená aplikace HR Cockpit v souladu se standardy společnosti Baader Computer používá více vrstvou architekturu, jak ji například popisuje samotný manuál pro JAVA EE[15]. Aplikaci tedy dělíme na několik od sebe oddělených vrstev.

- Prezentační vrstva
- Webová vrstva
- Aplikační vrstva
- Externí vrstva



Obrázek 7.1. Vícevrstvý model [15]

Vrstva webová pak také bývá často zahrnována pod vrstvu aplikační. Pod externí vrstvou potom obvykle spadá databáze. Vezmeme-li toto v potaz uvidíme, že ve výsledku je použitá klasická třívrstvá architektura, kterou dělíme na následující složky.

- Prezentační vrstva
- Aplikační vrstva
- Datová

Obecné rozdělení aplikace na více vrstev vede k větší přehlednosti, lepší udržitelnosti kódu a efektivnějšímu testování aplikace. Již z předchozích kapitol je potom zřejmé, pomocí jakých technologií budou jednotlivé vrstvy výsledné aplikace realizované.

- Prezentační vrstva - Kompletní js klient implementovaný pomocí frameworku Ext JS
- Aplikační vrstva - Realizovaná pomocí JAVA EE a frameworku Spring
- Datová - Představovaná PostgreSQL serverem.

7.4 Prezentační vrstva

Prezentační vrstva je na zbytku naprosto nezávislá, což teoreticky umožňuje její jednoduchou výměnu - případně jednoduché změny, aniž by ohrozily stav zbytku aplikace.

Veškerá komunikace mezi prezentační vrstvou a zbytkem aplikace je realizována pomocí AJAX požadavků směřovaných na aplikační server. K veškerému přenosu dat tak slouží pouze formát JSON.

7.4.1 Architektura

Jak už bylo dříve zmíněno, framework Ext JS má silnou podporu objektového a třídního systému, což velmi napomáhá při samotném vývoji. Také již bylo poukázáno, že tento framework podporuje a vybízí k použití návrhového vzoru MVC.

Jeho skutečné využití je sice určitě standardní a bez pochyb dobrou volbou, ale zároveň platí, že to framework nijak nevynucuje a ani to není jediný způsob, jak celý framework efektivně použít.

Pokud se vezme v potaz, jak celý framework funguje a jakým způsobem nabízí jednotlivé komponenty, které jdou navzájem dále skládat , tak se nabízí další, šikovně použitelný přístup.

V celé prezentační vrstvě tak hojně využíváme systému výrobních tříd, jenž nám vrací již sestavené komponenty. Díky tomu, že jsou funkční bloky rozděleny na menší části, je možné zdatně využívat metody skládání a je tak velmi jednoduché znovu používat kód.

7.4.2 Panely

Jako komponenta s nejvyšší pozicí je potom využívána tzv. karta (tab). Jedná se o panel, představující hlavní zobrazenou část obrazovky. V něm jsou pak pomocí skládání obsaženy menší komponenty. Vzhledem k možnosti frameworku, inicializovat objekty až při jejich skutečné potřebě, je dále výhodné z již zmíněných výrobních metod vracet pouze konfigurační objekty, ze kterých je framework schopen sám vytvořit třídu požadovaného typu. Níže je například vidět výsledný konfigurační objekt, sloužící k vytvoření obrazovky detailu, jakým je například detail kandidáta. Na této ukázce je také dobře vidět proces skládání. V parametru items jsou obsaženy další komponenty nebo pouze jejich konfigurace, jenž jsou v rámci tohoto detailu potřeba. Zde se například může jednat o panel obsahující formulář skládající se z různých polí, tlačítek a popisků.

```
return {
xtype : 'bcmastertabdetail',
title : title,
items : [panel],
bbar : btns,
currentMode : sharedSettings.openMode,
slaveTabPanel : slaveTabPanel,
autoModeSetting : true,
};
```

7.4.3 Vlastí komponenty

Díky OOP přístupu frameworku je možné vytvářet vlastní třídy, ze kterých je pak možné vytvářet instance v rámci aplikace. V předcházející ukázce je vidět konfigurační objekt sloužící k vytvoření a konfiguraci instance typu „bcmastertabdetail“. Což je vlastně jen třída rozšiřující o některé funkcionality svého předka, kterým je třída panel. Je tudíž možné ve vhodných případech vytvořit třídu novou, kterou lze pak při normálním používání frameworku dále používat. Celá konstrukce takové třídy je relativně jednoduchá a je na příkladě ukázaná na obrázku.

7.5 Aplikační vrstva

Aplikační vrstva systému je implementovaná pomocí jazyku JAVA, kdy využívá framework Spring MVC. Základní architekturou celé této vrstvy je návrhový vzor fasáda.

```
Ext.define('HRCO.competence.form.CompetenceWhispererCombo', {
  extend : 'HRCO.common.component.form.WhispererCombo',
  alias : 'widget.hrcocompetencewhisperercombo',

  keyDataType : 'String',
  cls : 'hrco-whisperer-combo',

  initComponent : function() {

    var fCompetenceTagKeyValueOTO = HRCO.JsonTO.competence.CompetenceTagKeyValueOTO.fields;

    this.model = BC.util.RecordUtils.createModelFromFields('CustomerKeyValueOTO', [{
              name : this.valueField,
              dataType : this.keyDataType
            }, {
              name : this.displayField,
              dataType : 'String'
            }, {
              name : fCompetenceTagKeyValueOTO.competenceId.name,
              dataType : fCompetenceTagKeyValueOTO.competenceId.dataType
            }, {
              name : fCompetenceTagKeyValueOTO.competenceName.name,
              dataType : fCompetenceTagKeyValueOTO.competenceName.dataType
            }, {
              name : fCompetenceTagKeyValueOTO.competenceDescription.name,
              dataType : fCompetenceTagKeyValueOTO.competenceDescription.dataType
            }], this.modelIdProperty);
    this.callParent();
  }
});
```

Obrázek 7.2. Ukázka definice jednoduché komponenty našeptávače

Veškerá aplikační logika je dostupná z několika, podle oblasti zaměření uskupených, fasád. Tento zvolený přístup zajišťuje transparentní rozhraní celé aplikace. Tato fasádová rozhraní, tak mohou být jednoduše dostupná řadičům, které jsou díky frameworku Spring schopny zachycovat případné požadavky z prezentační vrstvy.

V rámci tohoto přístupu je také snaha dodržovat pravidlo, že jeden požadavek prezentační vrstvy je splněn v rámci fasády jednou veřejnou metodou. Díky tomu je tak jasně transparentní mapování vnějších funkcionalit na vnitřní implementaci aplikace. V tomto případě se dále fasáda snaží vracet data ve formátu připraveném pro použití mimo aplikační vrstvu. Kvůli tomuto účelu je nutné odstranit pevné vazby na datové entity a častokrát agregovat data do jednoho přenosného objektu. Za tímto účelem používá aplikace DTO[1] vzor, kdy takovýto přenosný objekt naplní daty, jenž jsou na prezentační vrstvě nezbytné. Díky tomuto přístupu je tak zajištěno lepší oddělení mezi jednotlivými vrstvami a zároveň se sníží počet volání mezi jednotlivými vrstvami, protože se potřebná data přenesou najednou.

7.5.1 Zabezpečení

Jelikož aplikace funguje v uživatelském režimu a tím tedy umožňuje svým uživatelům přihlašovat se do aplikace pomocí svého tajného hesla, je potřeba vyřešit v rámci aplikace problematiku správy uživatelských hesel.

Na první pohled je zřejmé, že je nemožné uchovávat v databázi hesla v původním formátu. Relativně klasickým postupem je udržovat hesla v zašifrované podobě pomocí

[1] Data transfer object

jednosměrného algoritmu, např. pomocí SHA-1. Každopádně ani to není dostatečné a proto se považuje za doporučené doplnit tuto metodu o další bezpečnostní prvky. Je vhodné samotný převodní algoritmus spouštět ve více iteracích a do samotného hesla přidat navíc tzv. sůl, sérii znaků atd.

K zajištění spolehlivého uchovávání hesel, tak byla zvolena knihovna Jasypt[1], která splňuje všechny nezbytné požadavky pro uchovávání hesel. Tato knihovna navíc dokáže uchovávat již zmíněnou sůl v jednom řetězci znaků společně s šifrovaným heslem a není tedy ani nutné upravovat databázové schéma.

Knihovna Jasypt navíc podporuje jednoduché zapojení do aplikací využívající Spring, což ji dělá ještě lákavější volbou pro použití v rámci aplikace HR Cockpit.

7.5.2 Uživatelská oprávnění

Pomocí balíčku Spring Security a drobných rozšíření je docíleno toho, že před provoláním každé řádně anotované fasádové metody je spuštěn speciální autorizátor, jehož úkolem je ověřit jestli má daný uživatel požadované oprávnění ke spuštění dané akce. V rámci tohoto autorizátoru se vyhodnocuje tzv. zabezpečená operace, jež je v rámci dané anotace fasádové metody uvedena.

Na základě podkladů z analýzy byla sestavená jednoduchá tabulka, které konkrétní akce v rámci aplikace mohou jednotlivé uživatelské role vykonávat. Obdoba takovéto tabulky s definicemi práv pro jednotlivé uživatelské role je pak implementována i v rámci aplikace a při provolání dané metody je vyhodnoceno zda má aktuálně přihlášený uživatel oprávnění tuto konkrétní systémovou akci vyvolat.

Tímto způsobem je možné ověřit základní uživatelské práva, což ale nemusí ve všech případech stačit. Kromě samotného ověření práv uživatelské role je v mnoha případech důležité ověřit kontext, ve kterém je konkrétní akce vykonávaná.

Nejjednodušším takovým případem je například situace při modifikování uživatelova komentáře u kandidáta. V prvé řadě je sice skutečně nutné zda daná uživatelská role může přidávat nebo přímo modifikovat komentáře, ale navíc je v tomto konkrétním případě nutné ověřit, že je autorem upravovaného komentáře skutečně aktuálně přihlášený uživatel.

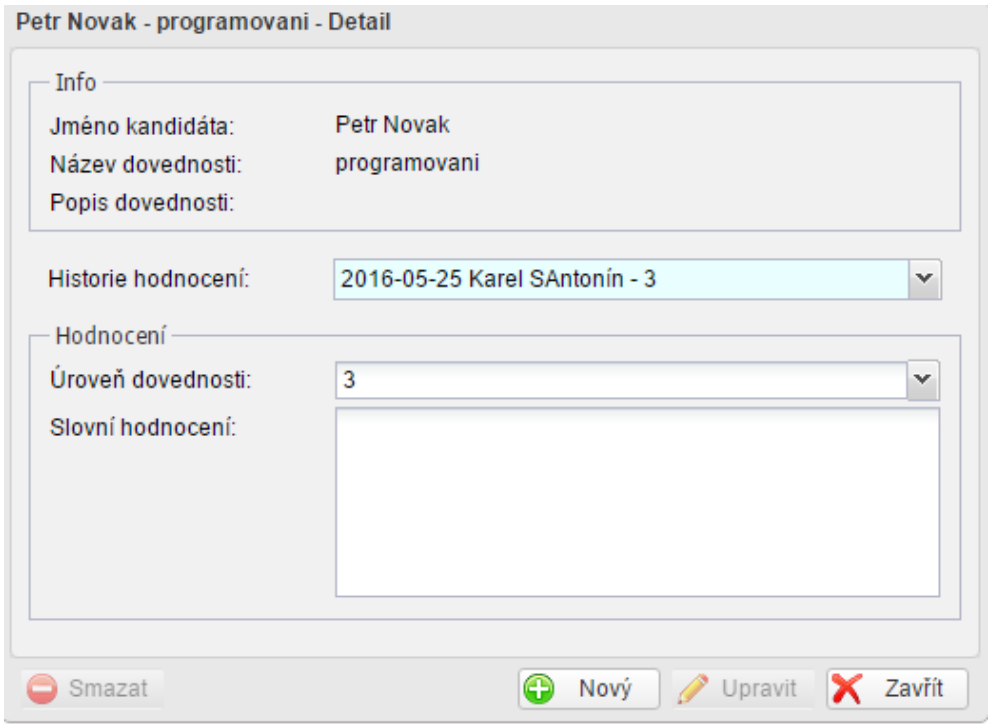
```
@Override
public BCnPermissionLevel resolveCandidateCommentModificationPermissionLevel(BCiSessionUser aSessionUser,
    BCcDataId aCandidateCommentId) {
  BCiSecurityManager securityManager = aSessionUser.getCurrentSecurityManager();

  boolean possibleToModifyCandidateDocument =
      securityManager.isPossibleTo(BCnCandidateSecurityAction.MODIFY_CANDIDATE_COMMENT);
  if (!possibleToModifyCandidateDocument) {
    return BCnPermissionLevel.DISABLED;
  }

  BCiCandidateDAO candidateDAO = getDaoManager().getCandidateDAO();
  BCcCandidateCommentEO candidateComment = candidateDAO.getCandidateComment(aCandidateCommentId,
      LockMode.NONE);
  BCcUserEO commentAuthor = candidateComment.getUser();
  if (BCsObjectUtils.objectsEqual(commentAuthor.getUserId(), aSessionUser.getUserId())) {
    return BCnPermissionLevel.ENABLED;
  }

  return BCnPermissionLevel.DISABLED;
}
```

Obrázek 7.3. Ukázka vyhodnocení oprávnění modifikovat komentář

Tyto třídy, sloužící k vyhodnocování uživatelských práv k vykonávání jednotlivých operací, jsou pak dále ještě využity pro přípravu dat pro prezentační vrstvu. V rámci

[1] http://www.jasypt.org/

prezentace je nutné rozhodnout jestli má uživatel oprávnění vidět některá data či případně spustit některou akci, a zda v tom případě může vidět příslušný ovládací prvek grafického rozhraní, například tlačítko. Díky tomu je možné odeslat v rámci dat určených prezentační vrstvě i informace o relevantních uživatelských oprávněních pro dané zobrazení. Prezentační vrstva se tak může na základě těchto informací rozhodnout, které ovládací prvky bude zobrazovat, případně i jak je bude zobrazovat.



Obrázek 7.4. Dialog hodnocení dovednosti s neaktivními ovládacími prvky

7.6 Datová vrstva

Datová vrstva aplikace je realizována pomocí PostgreSQL serveru a pro komunikací s ní je pak využit ORM framework Hibernate. V rámci implementace aplikace bylo celé databázové schéma udržováno pomocí aplikace Toad Data Modeler. Jako velmi cenná se pak ukázala schopnost právě této aplikace, pracovat s několika různými verzemi stejného schématu. Aplikace je schopná porovnat dvě různé verze téhož schématu, vyhodnotit jejich rozdíly a na jejich základě vygenerovat odpovídající změnové skripty pro databázový server. Tato schopnost se ukázala mnohokrát jako velmi přínosná a značně zjednodušovala celkovou práci s databázových schématem.

Další velmi přínosná funkcionalita pro vývoj této aplikace byla možnost spouštět nad spravovaným schématem uživatelské skripty. Díky tomu bylo možné generovat nad schématem skripty sloužící ke generování metadat popisující schéma, jenž mohly být importovány přímo do zdrojových kódů aplikace. Tato metadata jsou pak používána v datové části aplikace, kde slouží k dotazům nad databází, čímž je značně zefektivněna práce a zároveň klesá pravděpodobnost vzniku chyby.

Kapitola 8
Testování

Testování vyvíjené aplikace je naprosto zásadní částí celého vývojového procesu. Ačkoli je velmi jednoduché na ní jednoduše zapomenout, je nutné vždy pamatovat, že proces testování má na finální produkt zcela zásadní význam. Je totiž celkem evidentní, že pokud nejsou chyby vyvíjeného systému objeveny v rámci samotného vývojového systému, zcela určitě budou odhaleny zákazníkem během verifikačních testů, případně až koncovým uživatelem. Taková situace jak pak pro jakýkoli vývoj naprosto nepřípustná.V souladu se zadáním a se striktními požadavky na kvalitu zadavatele byla aplikace podrobena celé řadě testů.

8.1 Průběžné testování

V klasickém pohledu na vývojový proces se běžně testování uvádí jako poslední krok celého vývoje softwaru. Každopádně není to tak nejšťastnější a naštěstí řada metodik už k celé problematice přistupuje zcela jinak.

Jedním z důvodu, proč tomu tak je, je skutečnost, že čím později je daná chyba objevena, tím hůře a samozřejmě i nákladněji je odstraněna[16]. Je tudíž možné, aby chyba vzniklá v raných fázích samotného projektu mohla, pokud zůstala nepovšimnuta, způsobit škody o několik řádů větší.

Je proto důležité při celém procesu brát testování vždy v úvahu a uvědomovat si vazbu každé vývojové fáze k samotnému testování. Tento vztah je často dokládán tzv. „V“ modelem, na kterém je dobře vidět, že skutečně každá část vývojového procesu má přímý dopad na testování.

8.2 Testovací prostředí

Jak už bylo zmíněno v minulé kapitole pro vývoj této aplikace bylo mimo jiné vytvořeno speciální testovací prostředí. Jak už jeho název napovídá opravdu slouží v rámci testování aplikace. Jeho součástí je tak oddělená verze databáze, obsahující všechna data nutná pro spouštění testů. V tomto případě je tak opravdu absolutně nutné, aby byla databáze oddělená a všechny zásahy v ní byly prováděny s naprostou opatrností. Data v této databázi uložená, jsou totiž přímo využívána testy aplikace a jejich neuvážená změna může vést k náhlému selhání dříve implementovaných a bez problému funkčních testů.

8.3 Integrační a jednotkové testy

V rámci jednotkového testování se běžně testuje jasně daná uzavřená část kódu, často třída či metoda. Jedná se o první stupeň testování samotného kódu aplikace, kdy tento test provádí přímo vývojář, aby si ověřil správnost svého zásahu do aplikace. V těchto testech musí být často nějakým způsobem ošetřena nedostupnost ostatních částí systému.

Testy integrační jsou pak obvykle rozsáhlejší a testují vzájemnou integraci většího počtu komponent. Typicky tak například oproti klasickým jednotkovým testům dochází k zapojení i datové vrstvy aplikace a komunikaci se skutečnou databází.

V rámci dodržovaných standardů a výhod zvolené architektury došlo v rámci testování systému HR Cockpit k částečnému spojení těchto dvou druhů testů.

Vzhledem k použitému návrhovému vzoru fasáda, kdy je vnitřní logika aplikace zastřešena sérií fasádových metod se velmi dobře nabízí testovat funkcionalitu právě těchto tříd. Tudíž z pohledu jednotkového testování je zvolena za testovanou jednotku právě vždy jedna veřejná metoda fasády. Vzhledem k tomu, že pod fasádou jsou používány i datové zdroje, tak v rámci tohoto testu dochází k testování i integrace aplikační vrstvy s datovou.

8.3.1 Použitá technologie

K napsání a konfiguraci těchto testů pak byl použit framework TestNG[1]. Tanto framework inspirován klasickým JUnit frameworkem nabízí řadu nových funkcí, a tak velmi dobře plní všechny požadavky pro testování této aplikace.

8.3.2 Architektura testů

Jelikož jsou testovány všechny fasádové třídy a jejich veřejně dostupné metody, je snahou dodržet pravidlo, aby každé testované třídě odpovídala jedna třída specializovaná na její testování. Framework TestNG pak spouští sérii definovaných testů postupně se všemi připravenými testovacími parametry. Pro tyto testy je tudíž nutné připravit adekvátní testovací vstupní data, jenž náležitě otestují danou funkcionalitu. Součástí těchto testovacích parametrů je obvykle i případná očekávaná chyba, aby se ověřily i případné chybové vstupy aplikace. Součástí těchto testů také vždy bývá testování uživatelských oprávnění, a tak je každý test spouštěn pod jasně definovaným uživatelským profilem.

```
parameterBag.add(new BCcCandidateFacadeTestParameter(
    BCsTestUserHelper.HRASSISTENT_LHMS,
    new BCcCandidateTestParameter(candidateId, 2, new BCcExpectedExceptions(null,
        null,null, null, null, AccessDeniedException.class)),
    new BCcCandidateCommentTestParameter(BCsFacadeTestUtils.createCandidateCommentId(1L),
        2,
        new BCcExpectedExceptions(
        null,
        null,
        null,
        null,
        AccessDeniedException.class,
        AccessDeniedException.class)), new BCcCandidateDocumentTestParameter(
        BCsFacadeTestUtils.createDcaDocumentId(560L),
        1,
        noExpectedExceptions)));
```

Obrázek 8.1. Definice testovacího vstupu

8.4 Kontinuální integrace

Při každém novém zásahu do zdrojových kódů aplikace existuje nezanedbatelná šance zanesení chyby i v naprosto neočekávatelných místech. Toto riziko je pak samozřejmě ještě větší při provádění větších zásahů. Pokud je aplikace dobře pokryta jednotkovými

[1] http://testng.org/

```
@Test
public void testGetDeleteCandidate() throws Exception {
  BCiCandidateTestParameter candidateTestParameter = candidateFacadeTestParameter.
      getCandidateTestParameter();

  BCcDataId candidateId = candidateTestParameter.getCandidateId();

  BCiMethodCallResult<Object> result = BCsAssert.assertMethodCall(() -> {
    this.candidateFacade.deleteCandidate(candidateId);
    return null;
  }, candidateTestParameter.getExpectedExceptions().getExpectedDeletionException());

  if (result.isTestSkipped()) {
    return;
  }

  BCsAssert.assertMethodCall(() -> {
    return this.candidateFacade.getCandidateDetail(candidateId);
  }, BCxDataNotFoundException.class);

}
```

Obrázek 8.2. Jednoduchý test smazání kandidáta

a integračními testy, tak by měly být takovéto chyby s relativně vysokou šancí odhaleny. Každopádně v takovém případě by bylo nutné před každou provedenou změnou spouštět všechny testy aplikace.

Taková operace může být často značně náročná, jak časově tak i po stránce výpočetního výkonu. Také se nedá vyloučit lidská chyba a tudíž nejde zaručit, že vývojář vždy všechny testy při provádění změny spustí. K eliminaci je tedy vhodné použít nějakou formu automatického testování. Použije-li se některý nástroj sloužící k automatickému testování ke spouštění regresivních testů při každé změny provedených v aplikaci, je možné značně snížit riziko vnesení chyby do již původních částí aplikace.

Při vývoji aplikace HR Cockpit byl využit volně dostupný nástroj sloužící kontinuální integrace Jenkins[1]. Jedná se o velmi silný nástroj s mnoha dále dostupnými rozšířeními, v rámci tohoto vývoje byl však použit hlavně kvůli možnosti automatického spouštění testů nad projektem.

Se správným nastavením je Jenkins schopen monitorovat repositář sloužící k ukládání zdrojových kódů aplikace. Při každé změně provedené v repozitáři pak aplikaci sestaví a rámci sestavovacího cyklu spustí i všechny definované testy. Tudíž je zajištěno, že při jakékoli změně se ověří pomocí regresivních testů konzistence aplikace.

8.5 Zátěžové testování

Dalším často využívaným testem je testování zátěže systému. Pro případ správného vyhodnocení tohoto druhu testů je důležité pevně stanovit rozmezí povolených hodnot. V rámci specifikace, takové hodnoty stanoveny nebyly, není tedy příliš dobře možné vyhodnotit výsledky tohoto testování. Zároveň má na výsledek tohoto značný vliv server, na kterém aplikace běží. Jelikož zadavatel a v této situaci koncový zákazník požaduje nasazení na svém vlastním zařízení a o celkové technické zázemí se stará sám, ztrácí tento test ve výsledku určitou relevanci.

Samotná aplikace navíc momentálně zatím ani neběží na koncovém zařízení a testování tak proběhlo pouze v rámci vývojového prostředí.Vzhledem k těmto důvodům

[1] https://jenkins.io/

je pak hlavním výsledkem pouze to, jestli se při zatížení aplikace nevyskytla v rámci aplikace nějaká závažná chyba. Samotná diskuze nad odezvou a dalších parametrech týkající se výkonu celé infrastruktury ztrácí na významu.

K testování byl použit nástroj jMeter, který sloužil k simulaci požadavků na aplikaci. Vzhledem k tomu, že aplikace nemá žádné statické stránky, které by tento případně jiný obdobný nástroj mohl během testu automaticky najít, bylo potřeba připravit pro test manuálně několik požadavků. Průběh testu tak byl relativně jednoduchý.

- Přihlášení do aplikace
- Volba uživatelské role a mandanta
- Zobrazení výpisu kandidátů
- Zobrazení detailu jednoho z kandidátů

Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Latency	Connect Time(ms)
1	21:53:46.365	Thread Group 1-1	HTTP Request	239		736	239	2
2	21:53:46.606	Thread Group 1-1	HTTP Request	178		339	178	0
3	21:53:46.571	Thread Group 1-2	HTTP Request	401		736	400	1
4	21:53:46.973	Thread Group 1-2	HTTP Request	321		339	321	0
5	21:53:46.775	Thread Group 1-3	HTTP Request	532		736	532	2
6	21:53:46.978	Thread Group 1-4	HTTP Request	533		736	533	2
7	21:53:46.785	Thread Group 1-1	HTTP Request	849		2528	848	0
8	21:53:47.308	Thread Group 1-3	HTTP Request	354		339	354	0
9	21:53:47.165	Thread Group 1-5	HTTP Request	646		736	646	2
10	21:53:47.513	Thread Group 1-4	HTTP Request	337		339	337	0
11	21:53:47.375	Thread Group 1-6	HTTP Request	660		736	660	2
12	21:53:47.562	Thread Group 1-7	HTTP Request	521		736	521	1
13	21:53:47.812	Thread Group 1-5	HTTP Request	431		339	431	0
14	21:53:47.295	Thread Group 1-2	HTTP Request	949		2528	949	0
15	21:53:47.762	Thread Group 1-8	HTTP Request	670		736	670	2
16	21:53:48.084	Thread Group 1-7	HTTP Request	389		339	389	0
17	21:53:48.036	Thread Group 1-6	HTTP Request	469		339	469	0
18	21:53:47.963	Thread Group 1-9	HTTP Request	542		736	542	2
19	21:53:47.663	Thread Group 1-3	HTTP Request	883		2528	883	0
20	21:53:48.163	Thread Group 1-10	HTTP Request	483		736	483	1
21	21:53:48.433	Thread Group 1-8	HTTP Request	242		339	242	0
22	21:53:47.851	Thread Group 1-4	HTTP Request	824		2528	824	0
23	21:53:48.505	Thread Group 1-9	HTTP Request	255		339	255	0
24	21:53:48.363	Thread Group 1-11	HTTP Request	454		736	454	1
25	21:53:48.244	Thread Group 1-5	HTTP Request	695		2528	695	0
26	21:53:48.647	Thread Group 1-10	HTTP Request	304		339	304	0
27	21:53:48.561	Thread Group 1-12	HTTP Request	531		736	531	2
28	21:53:48.474	Thread Group 1-7	HTTP Request	683		2528	683	0
29	21:53:48.818	Thread Group 1-11	HTTP Request	351		339	351	0
30	21:53:48.505	Thread Group 1-6	HTTP Request	694		2528	694	0
31	21:53:48.762	Thread Group 1-13	HTTP Request	530		736	530	2
32	21:53:49.093	Thread Group 1-12	HTTP Request	292		339	292	0
33	21:53:48.675	Thread Group 1-8	HTTP Request	739		2528	739	0
34	21:53:48.761	Thread Group 1-9	HTTP Request	747		2528	747	0
35	21:53:48.962	Thread Group 1-14	HTTP Request	572		736	572	2
36	21:53:49.293	Thread Group 1-13	HTTP Request	322		339	322	0
37	21:53:49.163	Thread Group 1-15	HTTP Request	533		736	533	2
38	21:53:48.952	Thread Group 1-10	HTTP Request	759		2528	759	0
39	21:53:49.535	Thread Group 1-14	HTTP Request	324		339	324	0
40	21:53:49.362	Thread Group 1-16	HTTP Request	551		736	551	2
41	21:53:49.170	Thread Group 1-11	HTTP Request	758		2528	758	0
42	21:53:49.697	Thread Group 1-15	HTTP Request	324		339	324	0
43	21:53:49.563	Thread Group 1-17	HTTP Request	512		736	512	2
44	21:53:49.386	Thread Group 1-12	HTTP Request	753		2528	753	0
45	21:53:49.914	Thread Group 1-16	HTTP Request	280		339	280	0
46	21:53:49.763	Thread Group 1-18	HTTP Request	530		736	530	2

Scroll automatically? Child samples? No of Samples 450 Latest Sample 717 Average 444 Deviation 202

Obrázek 8.3. Zátěžový test

S ohledem na omezený výpočetní výkon pak samotné testování dopadlo uspokojivě.

8.6 Testování uživatelského rozhraní

Dalším aspektem aplikace, které je potřeba vystavit testování, je uživatelské rozhraní. Jak už bylo zmíněno během návrhu grafické podoby aplikace a návrhu jednotlivých obrazovek, vliv uživatelského rozhraní na celkový dojem celé aplikace je značný. Už kvůli tomu je zřejmé, že uživatelské rozhraní je něco co má skutečně význam testovat.

8.6.1 Testovací metoda

Pro samostatné testování existuje celá řada různých metod, ať už se jedná o testování s uživateli nebo bez něj. Při testování aplikace HR Cockpit byla použita tzv. metoda kognitivního průchodu. Jedná se o metodu testování uživatelského rozhraní bez přítomnosti uživatele. Jejím klíčovým aspektem je, aby se osoba, která samotný test provádí dokázala vhodně vžít do role cílového uživatele. Kvůli tomuto účelu musí pro průběh

testu jasně definována persóna typického uživatele aplikace. Na základě této charakteristiky se může značně lišit výsledek, dá se totiž očekávat, že uživatelům s jinými zkušenostmi může přijít ovládání aplikace naprosto rozdílně intuitivní.

Typický uživatel aplikace HR Cockpit je pak nejčastěji pracovník HR oddělení případně manažer nebo specialista v oboru dotýkajícího se pozice, na kterou je vypsánou výběrové řízení. U všech těchto uživatelů se poté dá očekávat minimálně uživatelské úroveň s využíváním klasických kancelářských a webových aplikací.

Během samotného testování pak tester v roli koncového uživatele prochází posupně jednotlivé kroky testovacího scénáře a po každém kroku odpovídá na následující otázky. Případné jiné problémy se samozřejmě zaznamenají také.

- Q1 : Bude uživateli zřejmé co dělat?
- Q2 : Spojí si uživatel správně popisek akci se svým cílem?
- Q3 : Dostane uživatel dostatečnou odezvu?

Za účelem tohoto testování bylo sestaveno několik testovacích scénářů, pokrývající ty nejtypičtější případy užití.

8.6.2 Testovací scénáře

TC1 Vytvoření výběrového řízení

1. Přihlásit se do aplikace jako HR manažer
2. Přepnout se do pohledu výběrových řízení
3. Otevřít dialogové okno pro vytvoření nového výběrového řízení
4. Zadat pracovní pozici „JAVA programátor“ a potvrdit hodnoty
5. Přepnout se do pohledu správy kol výběrového řízení
6. Zapnout editaci hodnot
7. Vytvořit jednu fázi výběrového řízení
8. Uložit změny
9. Přepnout se do pohledu hodnotitelů vytvořeného kola
10. Vyvolat dialog přidání nového hodnotitele
11. Vybrat hodnotitele
12. Přidat mu dovednost „java“.
13. Uložit změnu
14. Přepnout se do pohledu detailu výběrového řízení
15. Zahájit editaci
16. Zadat jako požadovanou dovednost „java“ s minimální úrovní 2 a maximální 4.
17. Zadat stav výběrového řízení aktivní
18. Uložit změny

TC2 Vytvoření kandidáta

1. Přihlásit se do aplikace jako HR manažer
2. Přepnout se do pohledu kandidátů
3. Otevřít dialogové okno přidání nového kandidáta
4. Vyplnit požadované údaje a z nabídky vybrat dříve založené výběrové řízení
5. Potvrdit hodnoty
6. Přepnout na panel s kandidátovými dovednostmi
7. Přidat dovednost „java“ se sebehodnocením 3.
8. Uložit změny

TC3 Ohodnocení kandidáta

1. Přihlásit se do aplikace jako hodnotitel
2. Přepnout se do pohledu výběrových řízení
3. Zobrazit detail dříve vytvořeného výběrového řízení
4. Přepnout se na pohled prvního kola výběrového řízení
5. Otevřít detail dříve vytvořeného kandidáta
6. Přepnout se panel kandidátových dovednosti
7. Otevřít dialog hodnocení dovednosti „java“
8. Přidat nové hodnocení
9. Uložit hodnocení
10. Přepnout se na dříve otevřenou kartu detailu kola výběrového řízení
11. Otevřít dialog hodnocení kandidáta
12. Přidat nové hodnocení
13. Uložit změny

TC4 Rozhodnout kandidáta

1. Přihlásit se do aplikace jako HR manažer
2. Přepnout se do pohledu výběrových řízení
3. Zobrazit detail dříve vytvořeného výběrového řízení
4. Přepnout se na pohled prvního kola výběrového řízení
5. Otevřít detail dříve vytvořeného kandidáta
6. Otevřít dialog hodnocení kandidáta
7. Prohlédnout si hodnocení kandidáta
8. Otevřít dialog rozhodnout kandidáta
9. Odmítnout kandidáta pro příliš vysoké platové požadavky
10. Uložit rozhodnutí

8.6.3 Výsledky testování

Během testování se objevilo několik chyb. Žádná svým typem přímo nebrání používání aplikace, každopádně i tak jsou nepříjemné a mohly by uživateli bránit ve správném a efektivním používání celé aplikace.

- TC1 Vytvoření výběrového řízení
  - Krok 9 - Při vytvoření nové fáze se objeví nová záložka s danou fází, ve které se nachází detail fáze. Uživateli v tento okamžik nemusí být úplně zřejmé, co očekávat a kde tento detail najít. Jako možné vylepšení by bylo vhodné přidat do seznamu fází tlačítko, které by tento detail přímo otevíralo.
  - Krok 17 - Při aktivování výběrového řízení není jasné, že již nepůjde znovu modifikovat seznam fází výběrového řízení a tedy, že je tento krok nevratný. Správně by se tak před uložením této změny mělo otevřít varování, informující uživatele o této skutečnosti.
- TC3 Ohodnocení kandidáta
  - Krok 5 - Tlačítko pro otevření detailu kandidáta má stejnou ikonu jako otevření dialogu pro kandidátovo ohodnocení. Tlačítka mají sice správný popisek, ale přesto by stejná ikona mohla být matoucí.
- TC4 Ohodnocení kandidáta
  - Krok 10 - Po rozhodnutí o kandidátovi se tato volba automaticky nepromítne do dialogu s přehledem kandidátových hodnocení.

krok testu	Q1	Q2	Q3
1.	ANO	ANO	ANO
2.	ANO	ANO	ANO
3.	ANO	ANO	ANO
4.	ANO	ANO	ANO
5.	ANO	ANO	ANO
6.	ANO	ANO	ANO
7.	ANO	ANO	ANO
8.	ANO	ANO	ANO
9.	NE	ANO	ANO
10.	ANO	ANO	ANO
11.	ANO	ANO	ANO
12.	ANO	ANO	ANO
13.	ANO	ANO	ANO
14.	ANO	ANO	ANO
15.	ANO	ANO	ANO
16.	ANO	ANO	ANO
17.	ANO	ANO	NE
18.	ANO	ANO	ANO

Tabulka 8.1. TC1 Vytvoření výběrového řízení

krok testu	Q1	Q2	Q3
1.	ANO	ANO	ANO
2.	ANO	ANO	ANO
3.	ANO	ANO	ANO
4.	ANO	ANO	ANO
5.	ANO	ANO	ANO
6.	ANO	ANO	ANO
7.	ANO	ANO	ANO
8.	ANO	ANO	ANO

Tabulka 8.2. TC2 Vytvoření kandidáta

Zároveň byly ještě objeveny různé drobnosti, například špatně lokalizované texty atd. Opět se nejedná o žádné závažné chyby přesto je ale nutné je důsledně odstranit.

krok testu	Q1	Q2	Q3
1.	ANO	ANO	ANO
2.	ANO	ANO	ANO
3.	ANO	ANO	ANO
4.	ANO	ANO	ANO
5.	ANO	NE	ANO
6.	ANO	ANO	ANO
7.	ANO	ANO	ANO
8.	ANO	ANO	ANO
9.	NE	ANO	ANO
10.	ANO	ANO	ANO
11.	ANO	ANO	ANO
12.	ANO	ANO	ANO
13.	ANO	ANO	ANO

Tabulka 8.3. TC3 Ohodnocení kandidáta

krok testu	Q1	Q2	Q3
1.	ANO	ANO	ANO
2.	ANO	ANO	ANO
3.	ANO	ANO	ANO
4.	ANO	ANO	ANO
5.	ANO	NE	ANO
6.	ANO	ANO	ANO
7.	ANO	ANO	ANO
8.	ANO	ANO	ANO
9.	NE	ANO	ANO
10.	ANO	ANO	NE

Tabulka 8.4. TC4 Rozhodnout kandidáta

Kapitola 9
Budoucnost projektu

Jak už bylo v samotné analýze projektu zmíněno, pro aplikaci je v plánu celá řada rozšíření. Tato rozšíření jsou několika různých druhů, kdy některé se přímo dotýkají tohoto modulu kandidátů a pouze rozšiřují jeho funkcionality. Jiné na druhou stranu směřují ke společnému většímu cíli návrhu aplikace kompletně pokrývající potřeby správy lidských zdrojů. V rámci dalšího vývoje se tak dá očekávat, že příští verze projektu by měly pouze rozšířit modul současný a až později přinést kompletně nové funkcionality vyvíjené v rámci dalších nových modulů

9.1 Nejbližší rozšíření

Jako nejbližší rozšíření se nabízí přidání několika funkcí, které byly z první verze aplikace vyloučeny. Jedná se například o tyto:

- Generování reportů
- Plánování schůzek
- Rozšířené vyhledávání kandidátů podle dovedností
- Správa e-mailové komunikace
- Notifikace systému

Jejich přidání do systému není naprosto nutné a podobu další verze je potřeba ještě konkretizovat, ale jedná se funkcionality, jež při původní analýze se zákazníkem někdy zazněly, a tudíž se dá očekávat v budoucnu jejich potřeba.

Jako další blízké rozšíření se poté nabízí zahájení vývoje administračního modulu, jenž by nabízel správu základních dat aplikace. Nejdůležitější se potom zdá správa uživatelů, dovedností a pracovních pozic.

9.2 Další rozšíření

Dalších modulů, které by svým zaměřením také pokrývaly část problematiky lidských zdrojů, je relativně hodně. Proto je jich uvedeno jen několik, u kterých se jeví pravděpodobnost implementace největší.

9.2.1 Modul zaměstnanců

Pro samotný cíl celé původně zamýšlené aplikace se jedná o velmi důležitý modul. Jeho primární funkcí by byla správa zaměstnanců společnosti využívající aplikaci. Funkcí, které by mohly být obsaženy v tomto modulu, je relativně hodně a některé by pak mohly mít svým rozsahem i celý vlastní modul. Například očekávaná správa zaměstnaneckých smluv včetně vývoje platových podmínek by tak mohla přejít až do modulu, zabývajícím se samotnými platy, případně správou účetnictví a financí atp. Součástí tohoto modulu by pak pravděpodobně byly nástroje na vedení rozvoje zaměstnanců a zvyšování jejich dovedností.

9.2.2 Modul mentorský

V rámci sběru původních požadavků na systém se také objevily zmínky o procesu mentorování nových zaměstnanců či členů trainee programů. Modul by pak podporoval vedení tohoto procesu včetně pravidelného hodnocení mentorované osoby. Umožňoval by vedení plánu rozvoje a jeho případné plnění.

9.3 Uplatnění projektu

Plánem celého projektu bylo od samého začátku nasazení v reálném prostředí HR oddělení společnosti Baader Computer. V nejbližší budoucnosti se tak dá očekávat delší iterace testování aplikace a následné spuštění testovacího provozu aplikace.

Primární cílem aplikace je sice interní využití projektu, přesto se tu počítá s určitou teoretickou možností dalšího externího využití aplikace. Jak už bylo v úvodu práce ukázáno, na trhu se podobné aplikace úspěšně vyskytují. Zřejmým problémem je nejspíše silná konkurence, která je navíc podpořena zavedeným jménem produktu a často se opírá o přímou podporu některého pracovního portálu.

Další formy využití vyvinuté aplikace jsou tak momentálně nejasné a zůstávají otázkou do budoucna i v závislosti na případná rozšíření aplikace.

Kapitola 10
Závěr

Obsahem této práce byl kompletní vývojový proces aplikace sloužící ke správě lidských zdrojů, konkrétně v této první verzi ke správě výběrových řízení. V rámci práce na tomto projektu byla provedena důkladná analýza celkové problematiky včetně úvodního základního výzkumu v samotné oblasti problematiky lidských zdrojů a procesu výběrových řízení. Výsledkem je pak funkční aplikace, při jejímž vývoji bylo dodrženo všech nezbytných postupů a zásad, aby splňovala ty nejvyšší požadavky na výslednou kvalitu. Aplikace byla podrobena všem požadovaným testům.

Vyvinutá aplikace pak také měla být v blízké budoucnosti spuštěna do ostrého provozu. V plánu je i vývoj dalších funkcionalit a rozšíření, jež byly již dříve podrobněji popsány.

Osobně mi pak práce na tomto projektu poskytla unikátní příležitost zpracovat zadání z praxe a projít celým vývojovým procesem. Velmi si cením možnosti setkání se zvyklostmi a metodikami vývoje softwaru v praxi díky spolupráci se společností Baader Computer a odborníky z praxe, se kterými jsem se během práce na této aplikaci mohl setkat.

Literatura

[1] *Baader Computer* [online]. [cit. 2016-05-25]. Dostupné z: www.bcpraha.com

[2] KOUBEK, Josef. *Personální práce v malých a středních firmách*. 4., aktualiz. a dopl. vyd. Praha: Grada, 2011. Management (Grada). ISBN 9788024738239.

[3] BOYATZIS, Richard E. *The competent manager: a model for effective performance*. New York: Wiley, c1982. ISBN 047109031X.

[4] MATĚJKA, Marek a Pavel VIDLAŘ. *Vše o přijímacím pohovoru: jak poznat druhou stranu*. 2., přeprac. a aktualiz. vyd. Praha: Grada, 2007. ISBN 9788024719726.

[5] KOCIANOVÁ, Renata. *Personální činnosti a metody personální práce*. Praha: Grada, 2010. Psyché (Grada). ISBN 9788024724973.

[6] *Profesia.sk: Mark* [online]. [cit. 2016-05-23].Dostupné z: http://www.profesia.sk/mark/

[7] *Teamio* [online]. [cit. 2016-05-24]. Dostupné z: www.teamio.com

[8] WIEGERS, Karl Eugene. *Požadavky na software*. Brno: Computer Press, 2008. ISBN 9788025118771.

[9] *Tutorials Point* [online]. [cit. 2016-05-23]. Dostupné z: http://www.tutorialspoint.com/extjs/

[10] *Ext JS Guides* [online]. [cit. 2016-05-23]. Dostupné z: https://docs.sencha.com

[11] *ExtJS vs AngularJS* [online]. [cit. 2016-05-23]. Dostupné z: http://www.techferry.com/articles/ExtJS-vs-AngularJS.html

[12] *Maven* [online]. [cit. 2016-05-23]. Dostupné z: http://maven.apache.org/

[13] GALITZ, Wilbert O. *The essential guide to user interface design: an introduction to GUI design principles and techniques*. 2nd ed. New York: Wiley Computer Pub., c2002. ISBN 0471084646.

[14] *Stránky předmětu* `Návrh uživatelských rozhraní` [online]. [cit. 2016-05-23]. Dostupné z: http://nur.felk.cvut.cz/

[15] SUN MICROSYSTEMS, INC. *The Java EE 5 Tutorial* [online]. In: . [cit. 2016-05-24]. Dostupné z: https://docs.oracle.com/javaee/5/tutorial/doc/javaeetutorial5.pdf

[16] MINDUCA, ARTHUR. Quality assurance in software development: When should you start the testing process? In: *FIRST, SOLVE THE PROBLEM. THEN, WRITE THE CODE* [online]. 2014 [cit. 2016-05-25]. Dostupné z: https://arthurminduca.com/2014/03/07/quality-assurance-in-software-development-when-should-you-start-the-testing-process/

Příloha A
Seznam použitých zkratek

BC	Baader Computer
CRM	Řízení vztahů se zákazníky
CRUD	vytvoření (C), čtení (R), editace (U), smazání (D)
ČVUT	České vysoké učení technické v Praze
DTO	Objekt sloužící k přenosu dat
ERP	Plánování podnikových zdrojů
HR	Human resources
JS	JavaScript
JSON	JavaScriptový objektový zápis
MVC	Model-view-controller
MVVM	Model-View-ViewModel
OOP	Objektově orientované programování
ORM	Objektově relační mapování
SASS	Rozšíření kaskádových stylů
TC	Testovací případ

Příloha B
Uživatelské příručka

Obsahem této příručky jsou základní informace nutné k přihlášení do aplikace.

V době vytvoření této práce běží aplikace v testovacím režimu na serveru společnosti Baader Computer a je tedy přístupná, jako ukázka výsledné aplikace.

```
https://tomcat7.bcpraha.com/hrco/
```

Případně je možné nainstalovat aplikaci z přiloženého CD dle pokynů v další příloze.

V aplikaci jsou pro testovací účely připraveni tito uživatelé.

přihlašovací jméno	heslo	uživatelská role
hrmanager	1	HR manažer
smith	1	HR asistent
svoboda	1	Manažer
kantonin	1	Team leader
stastny	1	Hodnotitel

Příloha C
Instalační manuál

C.1 Instalace databáze

Aplikace ke svému běhu vyžaduje server PostgreSQL, který je volně stažitelný a jednoduše instalovatelný.

```
https://www.postgresql.org
```

Pro vytvoření potřebné databáze s testovacími daty je na přiloženém CD dvojice skriptů ve složce

```
instalace\db
```

C.2 Instalace aplikace

Na přiloženém CD se ve složce instalace nachází již připravený soubor hrco.war, který je možné nahrát na cílový server. Tato verze aplikace je konfigurována k připojení na lokální PostgreSQL server s následujícími parametry.

```
adresa : localhost
port : 5432
db : hrcockpit
uživatel : hrcockpit_user
heslo : hrcockpit_user
```

Pokud je potřeba tuto konfiguraci změnit, je nutné aplikaci sestavit znovu s upravenými parametry. Požadovanou změnu konfigurace je nutné provést v následujícím souboru

```
src\environments\webclient\webclient-hrco-local\war\WEB-INF\
config\jdbc.properties
```

K sestavení aplikace je nutné použít nástroj Maven. Do místního repozitáře Mavenu je dále potřeba překopírovat obsah složky

```
instalace\maven\repository
```

K samotnému sestavení je nutné využít Java ve verzi 8. Nad složkou „webclient-hrco-local“ je pak nutné spustit následující příkaz.

```
mvn clean package -U -o
```

Následně vygenerovaný soubor hrco.war stačí poté opět nahrát na webový server.

Příloha D
Databázové schéma

Příloha E
Obsah přiloženého CD

- /src/ - Zdrojové soubory aplikace HR Cockpit
- /prilohy/db-schema - Databázové schéma
- /prilohy/scenare.pdf - Scénáře k případům užití
- /instalace/hrco.war - Aplikace HR Cockpit
- /instalace/db/ - Databázové skripty
- /instalace/maven/ - Maven repozitář
- /dp/tex/ - Zdrojové soubory diplomové práce
- /dp/kohousim2016dp.pdf - Diplomová práce v pdf