UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY
A INFORMATIKY

PRACOVNÁ PLOCHA VÝSKUMNÍKA

BAKALÁRSKA PRÁCA

2014 ONDREJ BRICHTA

UNIVERZITA KOMENSKÉHO V BRATISLAVE FAKULTA
MATEMATIKY, FYZIKY A INFORMATIKY

PRACOVNÁ PLOCHA VÝSKUMNÍKA

BAKALÁRSKA PRÁCA

Študijný program: Aplikovaná informatika
Študijný odbor: 2511 Aplikovaná informatika
Školiace pracovisko: Katedra aplikovanej informatiky
Školiteľ: Mgr. Pavel Petrovič, PhD.

Bratislava, 2014

Ondrej Brichta

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Ondrej Brichta
Študijný program: aplikovaná informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: 9.2.9. aplikovaná informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský

Názov: Pracovná plocha výskumníka / *Researcher's Desktop*

Cieľ: Cieľom práce je nadviazať na predchádzajúcu bakalársku prácu, ktorej výsledkom bolo vytvorenie systému pre spravovanie informácii výskumníka. Predchádzajúca práca vytvorila kvalitný základ, avšak vynechala implementáciu viacerých podstatných funkcií, nedotiahla používateľské rozhranie, funkcie vyhľadávania, archivácie, previazania. Úlohou študenta je preskúmať existujúci model, urobiť v ňom potrebné modifikácie a implementovať chýbajúcu funkcionalitu, aby sa systém stal nasaditeľným v prevádzke. V prípade nutnosti môže študent navrhnúť a implementovať systém nový.

Literatúra:
1. Katarína Matysová: Plocha výskumníka, bakalárska práca FMFI, 2011.
2. Google Web Toolkit, Developer Guide, on-line: http://code.google.com/webtoolkit/doc/latest/DevGuide.html

Kľúčové slová: webová aplikácia, kalendár, osobný plán

Vedúci: Mgr. Pavel Petrovič, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: doc. PhDr. Ján Rybár, PhD.

Dátum zadania: 22.10.2013

Dátum schválenia: 22.10.2013

doc. RNDr. Damas Gruska, PhD.
garant študijného programu

.....
študent

Pavel Petrović
.....
vedúci práce

ČESTNÉ VYHLÁSENIE

Vyhlasujem, že som diplomovú prácu vypracoval samostatne a uviedol som všetku použitú literatúru.

.....

Ondrej Brichta

ABSTRAKT

Cielom tejto práce je nadviazať na existujúcu bakalársku prácu ktorej výsledkom bolo vytvorenie systému pre spravovanie informácií výskumníka a pokračovať v implementácii chýbajúcej funkcionality, používateľského rohrania a celkovo dotvoriť aplikáciu do funkčnej podoby. Medzi hlavné ciele patrí vytvorenie chýbajúceho e-mailového klienta, databázy kontaktov a projektov. Popri tom treba dokončiť a zfunkčniť niektoré už implementované funkčné prvky.

Kľúčové slová: webová aplikácia, Ajax, Google Web Toolkit, MVP, Java

ABSTRACT

The goal of this bachelor thesis is to enhance the functionality of a previously submitted bachelor thesis. The original bachelor thesis created an information management system for a research team. The goal of this thesis is to continue with the implementation of missing funcionality and of the graphic interface, and to give the underlying application a finishing touch. The main goals are: the creation of a missing email client, as well as a database of contacts and projects. The side goal is to enhance some of the already implemented functionality.

Keywords: web application, Ajax, Google Web Toolkit, MVP, Java

Obsah

1. PREHĽAD PROBLEMATIKY

1.1 MOTIVÁCIA A ZÁKLADNÉ CHARAKTERISTIKY APLIKÁCIE

Táto aplikácia je určená najmä pre výskumníkov, ktorí sú veľmi pracovne vyťažení. Tí potrebujú synchronizáciu množstva činností, efektívnu organizáciu svojho pracovného času, miesto na odkladanie a jednoduchý prehlad svojich, ale i cudzích prác, časopisov, poznámok a podobných vecí, ktoré sú pre dnešného výskumníka neoddeliteľnou súčastou pracovného života. Najmä ale, aby vsetky tieto informácie mohli byť poprepájané a dostupné na jednom mieste odkiaľkovek.

Keďže každý už vytvorený program je príliš obmedzený, alebo naopak príliš univerzálny, teda obsahuje príliš vela nepotrebnej funkcionality, je potrebné vytvoriť samostatnú aplikáciu, ktorá spĺňa presné požiadavky výskumníkov.

Na dosiahnutie tohto cieľa slúži práve pracovná plocha výskumníka. Je to webová aplikácia, ktorá umožní výskumníkovi zhromažďovať a ukladať štrukturované informácie (projekty, dokumenty, publikácie, softver, a pod.), spravovať kalendár (udalosti, konferencie, harmonogram projektov, atď.), zoznam úloh, databázu kontaktov a dôležité zapisovať poznámky. Hlavnou výhodou pracovnej plochy výskumníka, je nepochybne to, že na rozdiel od iných programov ktoré podporujú vyššie uvedené schopnosti, pracovná plocha výskumníka ich nielenže podporuje všetky naraz, ale najmä ich dokáže prepájať. Každý prvok sa dá prepojiť s akýmkoľvek iným prvkom, čo veľmi zrýchľuje vyhľadávanie súborov, projektov, ľudí alebo čohokolvek iného, čo práve výskumník potrebuje. Ďalšia výhoda programu je možnosť roztrieďovať informácie podľa kategórií, pričom ich zoznam nie je konečný, ale užívateľ si ho môže definovať sám.

Okrem spomenutých pozitív je ešte nepochybne výhodou aj to, že celá aplikácia je tvorená ako Open Source. To znamená, že po skončení tejto práce, bude môcť byť aplikácia dalej modifikovaná, tak, ako ju konkrétny užívateľ práve potrebuje.

1.2 SÚČASNÝ STAV EXISTUJÚCICH RIEŠENÍ

V súčastnej dobe internetového rozmachu existuje mnoho programov a internetových služieb, ktoré pokrývajú časť funkcionality plochy výskumníka. Vieme nájsť programy, ktoré sú vynikajúcimi organizérmi, programy, ktoré dokážu efektívne spravovať prácu s projektami, kontaktmi, atď. Žiaden z nich ale neposkytuje takú funkcionalitu a prepojenosť ako práve pracovná plocha výskumníka. Tieto programy dokážeme rozdeliť na dve skupiny. Skupinu

manažérov projektov a skupinu organizérov. Skupina manažérova projektov vie obvykle veľmi dobre zvládať organizáciu dokumentov, softveru a iných štrukturovaných informácií. Nedokáže ale spájať jednotlivé projekty podla typu, nevie pripájať ľudí ku projektom, spájať projekty s udalosťami a kalendárom a nedokaže pripájať poznámky. Naproti tomu, skupina organizérov je výborná v ukladaní informácií o ľuďoch, udalostiach, konferenciách a zapisovaní poznámok, ale nedokážeme do nich ukladať štrukturované dokumenty, odkladať na ne dáta a pod.

Toto a vyššie uvedené dôvody sú hlavnou motiváciou na pokračovanie v projekte pracovnej plochy výskumníka.

1.3 Cieľ práce

Cieľom bakalárskej práce je pokračovať v načatej práci. Zároveň analyzovať existujúce riešenia, aby sa mohli porovnávať a zistiť aké by malo byť optimálne riešenie tohto zložitého problému.

1.4 Štruktúra práce

V druhá kapitola sa rozoberá východisková práca v ktorej pokračujem a iné existujúce riešenia, ktoré sú vzájomne porovnané.

Tretia kapitola sa zaoberá prehľadom technológií ktoré sú použité pri vývoji aplikácie. Sú podrobne analyzované a zdôvodnené ich pozitíva a negatíva.

Štvrtá kapitola sa zaoberá funkčnou špecifikáciou programovanej časti aplikácie.

Piata kapitola sa zaoberá návrhom riešenia po uživateľskej a grafickej stránke aplikácie.

Šiesta kapitola sa zaoberá implementáciou aplikácie. Je tu podrobne rozpísané ako jednotlivé prvky fungujú po technickej stránke.

Siedma kapitola sa zaoberá možnostami ako by sa aplikácia mohla rozvíjať v budúcnosti. Obsahuje návrhy na funkcionalitu ktorá sa nestihla implementovať v tejto práci.

Ôsma kapitola obsahuje inštrukcie ako nainštalovať túto aplikáciu.

Deviata kapitola obsahuje zhrnutie celej práce.

2. EXISTUJÚCE RIEŠENIA

2.1 PREDCHÁDZAJÚCA BAKALÁRSKA PRÁCA

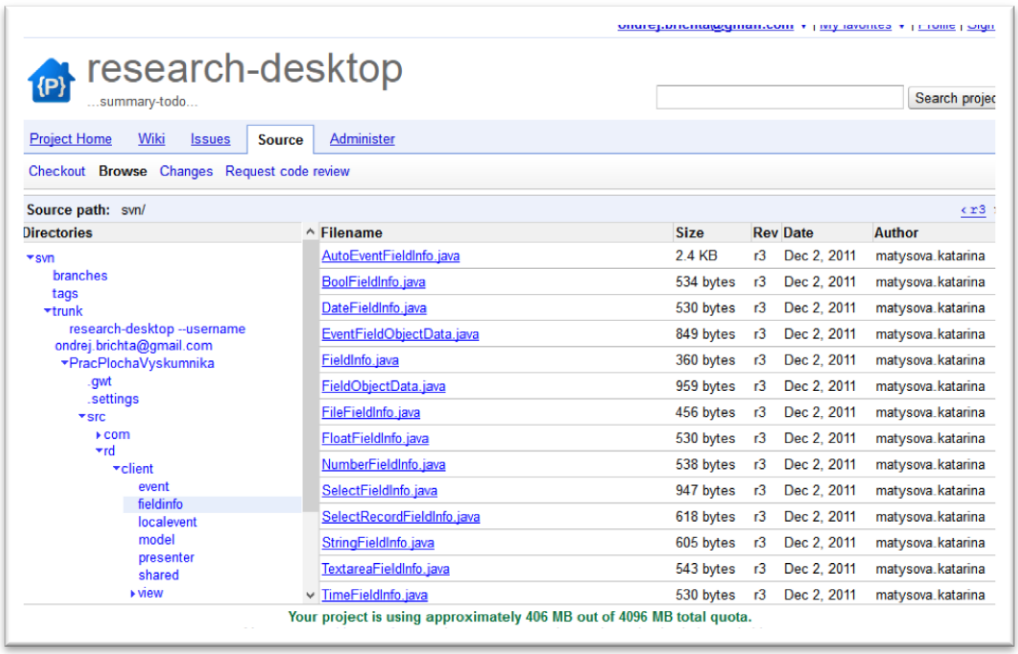
Bakalárska práca, z ktorej sa vychádza a v ktorej pokračujem (1), bola naprogramovaná ako samostatne fungujúca webová aplikácia, ktorá spĺňa väčšinu z vyššie spomenutých požiadaviek. Dá sa do nej pridávať informácie, prepájať ich a filtrovať podla kategórií, má implementovaný kalendár a poznámky. Chýba jej ale podstatná časť v podobe projektov a kontaktov. Týmto častiam sa bude venovať táto bakalárska práca.

2.2 PODOBNÉ PROGRAMY

Na internete sa dá nájsť množstvo programov, ktoré sú podobné nášmu programu. Pre porovnanie si vyberieme niekoľko najznámejších a porovnáme ich.

2.2.1 GOOGLE CODE

Google code je jedna z najpoužívanejších freeware webových aplikácií. Umožňuje nahrávať zdrojový kód projektov a ich spracovávanie, zdielanie so spolupracovníkmi, upravovanie pre viac užívateľužívateľov naraz, pripájanie cez SVN, verzionovanie projektov, komentáre a má aj možnosť pridávania zoznamov úloh. Dokumenty sa dajú pridávať do zložkových štruktúr da sa nahrávať a stiahnut aj celý projekt naraz.



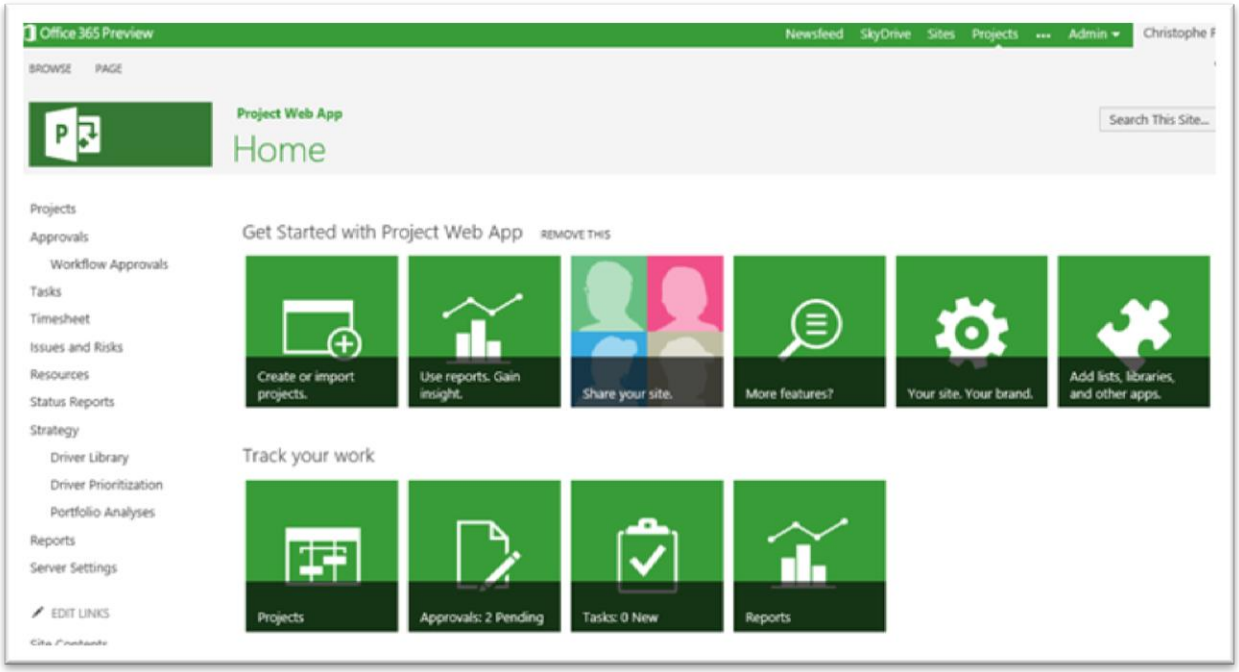
Obrázok 1. Ukážka webovej aplikácie Google Code

Medzi hlavné nedostatky patrí najmä nemožnosť pridávať prepojenia na iné projekty a to, že neobsahuje kalendár ani kontakty.

2.2.2 MICROSOFT PROJECT

Microsoft Project patrí medzi jeden z najpoužívanejších projektových manažérov v profesionálnych firmách. Existuje v troch verziách z ktorých sa dá vybrať podľa požiadaviek užívateľa. Pre užívateľov s nenáročnými požiadavkami, ktorým stačí samostatne fungujúca verzia nepotrebujúca pripojenie na server, je určený Standard. Pre užívateľov, ktorí potrebujú pokryť väčšie projekty a zároveň chcú mať k nim prístup odkiaľkoľvek existuje verzia Profesional a pre užívateľov, ktorí chcú pristupovať najmä z neznámych počítačov existuje čisto webová verzia Web Access. Medzi hlavné výody Microsoft Project patrí variabilnost. Vďaka obrovskému množstvu vývojárov a používateľov má veľmi veľa možností, aby vyhovel akémukoľvek projektu. Taktiež je dokonale previazaný na kancelársky balík Microsoft Office, s ktorým dokáže bezchybne komunikovať a spolupracovať.

Medzi hlavné nedostatky patrí najmä cena, ktorá sa pri základnej verzii pohybuje v stovkách eur a pri vyšších verziách dokonca až v tisíckach. Daľšou značnou nevýhodou je, že nepodporuje iné operačné systémy ako Windows. Taktiež je príliš zložitý a pokiaľ ho nepoužívate na koordinovanie veľkého tímu, tak jeho potenciál nevyužijete.

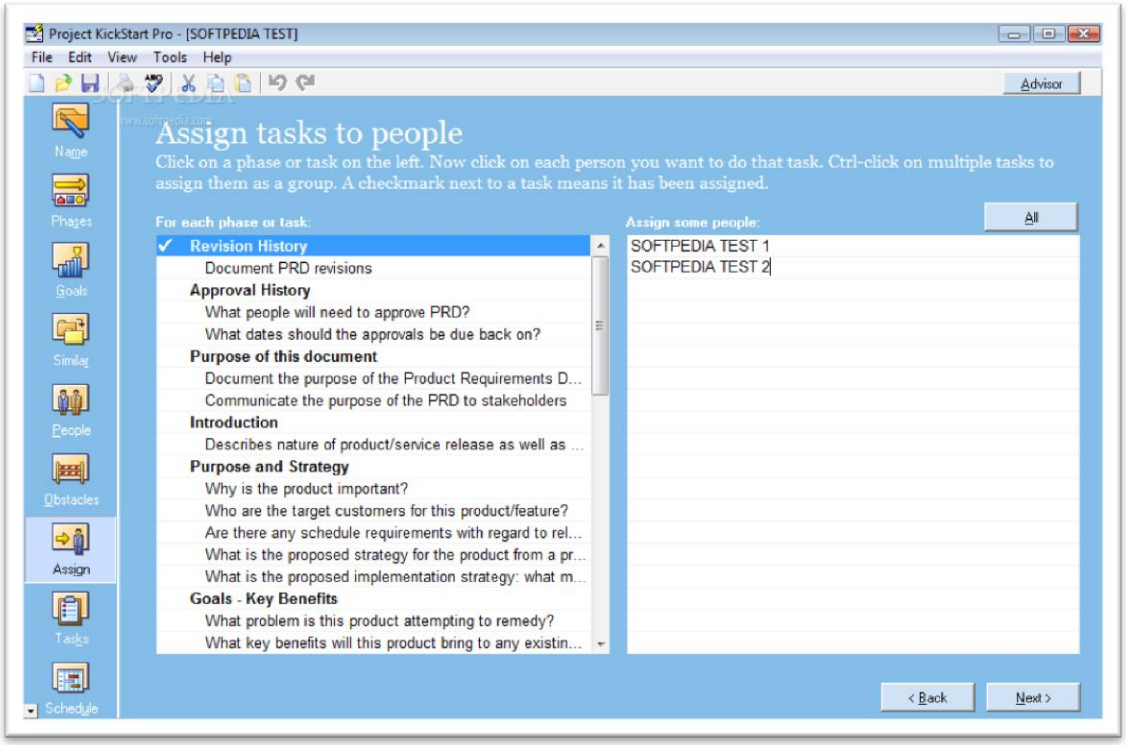


Obrázok 2. Ukážka programu Microsoft Project

2.2.3 KICKSTART

Projekt KickStart je ďalší v poradí veľmi oceňovaných projektových manažérov. Podobne ako MS Project, má viac verzií. Základnú, ktorá postačuje iba na jednoduché projekty, a pokročílú, ktorá dokáže uspokojiť aj mierne náročnejšieho používateľa. Obe verzie programu sú veľmi intuitívne, dokážu zvládať ukladanie projektov a zvládajú základné funkcie organizéra. Taktiež ako jeden z mála programov poskytuje bezplatnú podporu.

Najväčšie nedostatky programu sú v podstate spôsobené jeho prednosťami. Tým že je program veľmi jednoduchý, tak nedokáže podporovať príliš veľké projekty. Taktiež, rovnako ako MS Projekt, nie je multiplatformový, teda podporuje iba opračný systém Windows.



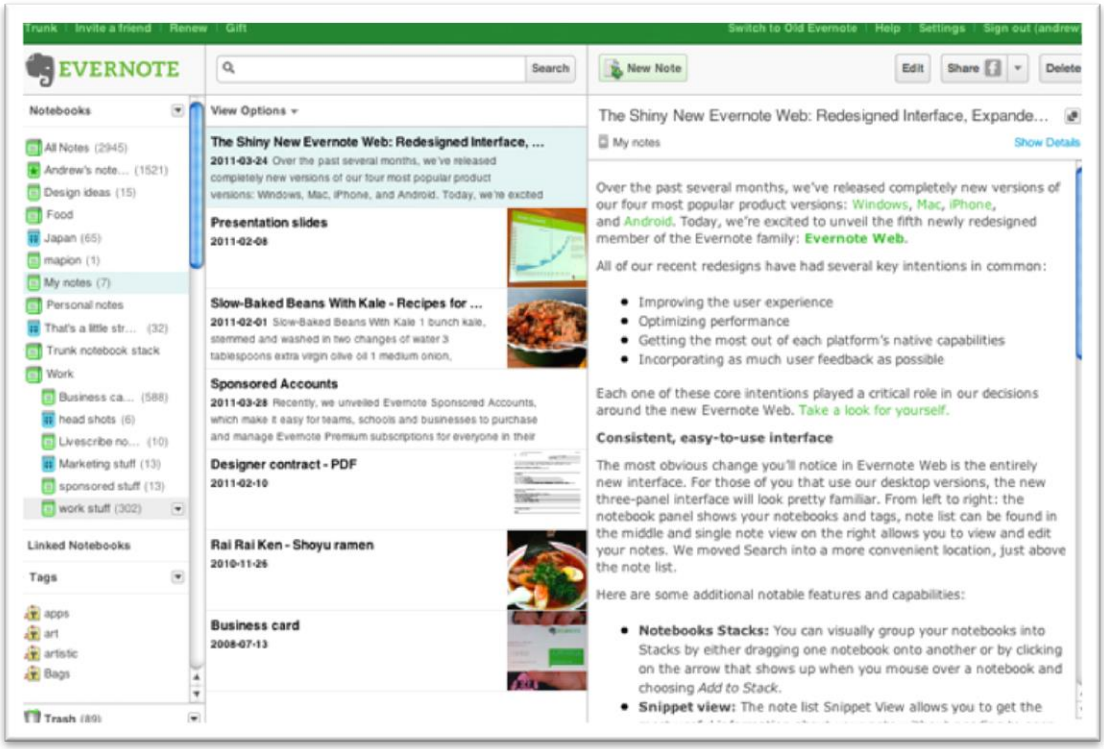
Obrázok 3. Ukážka programu KiskStart

2.2.3 EVERNOTE

EverNote patrí k webovým aplikáciám, ktoré sa zaraďujú medzi personálne organizéry. Radí sa medzi najlepšie aplikácie z tejto kategórie. Má niekoľko verzií, medzi pre nás najdôležitejšie ale patrí hlavne webová verzia, ktorá je dostupná priamo cez webový prehliadač, pretože sa najviac podobá na naše požiadavky. Aplikácia dokáže zaznamenávať a ukladať akékoľvek dokumenty, poznámky alebo webové dokumenty. Podporuje širokú škálu bežne dostupných formátov a umožnuje ich vytvárať a upravovať aj priamo v programe.

Dokáže organizovať dokumenty a poznámky podla tém a vkladať ich do zápisníkov. Dokumenty sa dajú zdieľať medzi ľuďmi a možu ich upravovať viacerí používatelia naraz.

Medzi hlavné nevýhody patrí najmä nemožnosť udržiavať kontakty a kalendár. Takiež to, že aplikácia je pre neplatiacich užívateľov výrazne obmedzená a k jej plnému potenciálu sa dá dostať až zakúpením si prémiového účtu.



Obrázok 4. Ukážka programu Evernote

2.3 VÝSLEDOK POROVNANIA PODOBNÝCH PROGRAMOV

Z vyššie uvedených dôvodov som dospel k záveru, že žiaden z dostupných programov nespĺňa naše požiadavky. Z tohoto dôvodu som sa rozhodol naprogramovať aplikáciu Pracovná Plocha Výskumníka.

3. PREHĽAD POUŽITÝCH TECHNOLÓGIÍ

3.1 ZÁKLADNÉ WEBOVÉ TECHNOLÓGIE

Pri programovaní aplikácie používam základné webové technológie, ktoré netreba vysvetľovať, pretože čitateľ je s nimi už oboznámený. Sú to technológie XHTML, CSS a JavaScript.

3.2 AJAX

Ajax (*Asynchronous JavaScript + XML*) je súhrnné označenie pre viacero programovacích jazykov, používaných pre vývoj asynchrónnych interaktívnych webových aplikácií, ktoré umožňujú meniť obsah webovej stránky bez opätovného načítania kompletnej stránky, za pomoci XML a XMLHttpRequest. Pri potrebe zmeniť obsah stránky sa pošle konkrétna požiadavka na server, server ju spracuje a odošle naspäť iba údaje ktoré si stránka vyžiadala. Tie sa následne spracujú pomocou JavaScriptu a na stránke nastane požadovaná zmena.

Ďalším pozitívom tejto technológie je, že v prípade dobrého implementovania, výrazne redukuje objem dátových tokov. Taktiež výrazne odlahčí zaťaženie serverov, keďže pri každej požiadavke server nemusí posielať celý webový dokument, tak ako pri klasickom prístupe ku web stránke, ale stačí poslať iba potrebnú časť. Server teda nemusí vyhľadávať všetky súčasti web stránky v pamäti ale nájde iba požadovanú časť a odošle ju.

3.3 JAVA

Java je objektovo orientovaný programovací jazyk, ktorý je momentálne jedným z najpoužívanejších programovacích jazykov na svete. Je to najmä vďaka tomu, že Java je multiplatformová, teda dá sa spustiť takmer na akomkoľvek operačnom systéme. Táto jej unikátna vlastnosť je spôsobená unikátnym spôsobom kompilácie, kde namiesto toho aby sa zdrojový kód prekladal do strojového kódu, prekladá sa do inštrukcií v binárnej podobe (*Java bite code*), ktoré su interpretované na virtuálnom stroji (*Java Virtual Machine*). Vďaka tomuto je teda Java dostupná na ktorejkoľvek platforme, kde je dostupná Java Virtual Machine.

Pre vývoj aplikácií v Jave je potrebný Java Development Kit (JDK), ktorý poskytuje potrebné nástroje a knižnice. Na vývoj je k dispozícii niekoľko edícií. Najpoužívanejšie sú Java SE, pre vytváranie bežných Java appletov a desktopových aplikácií, Java EE, pre rozsiahle webové aplikácie (táto verzia sa používa aj pri tejto práci), a Java ME, pre mobilné

aplikácie. Existujú ešte aj ďalšie menej používané verzie, ako napríklad Java Card, ktorá sa používa v inteligentných čipových kartách.

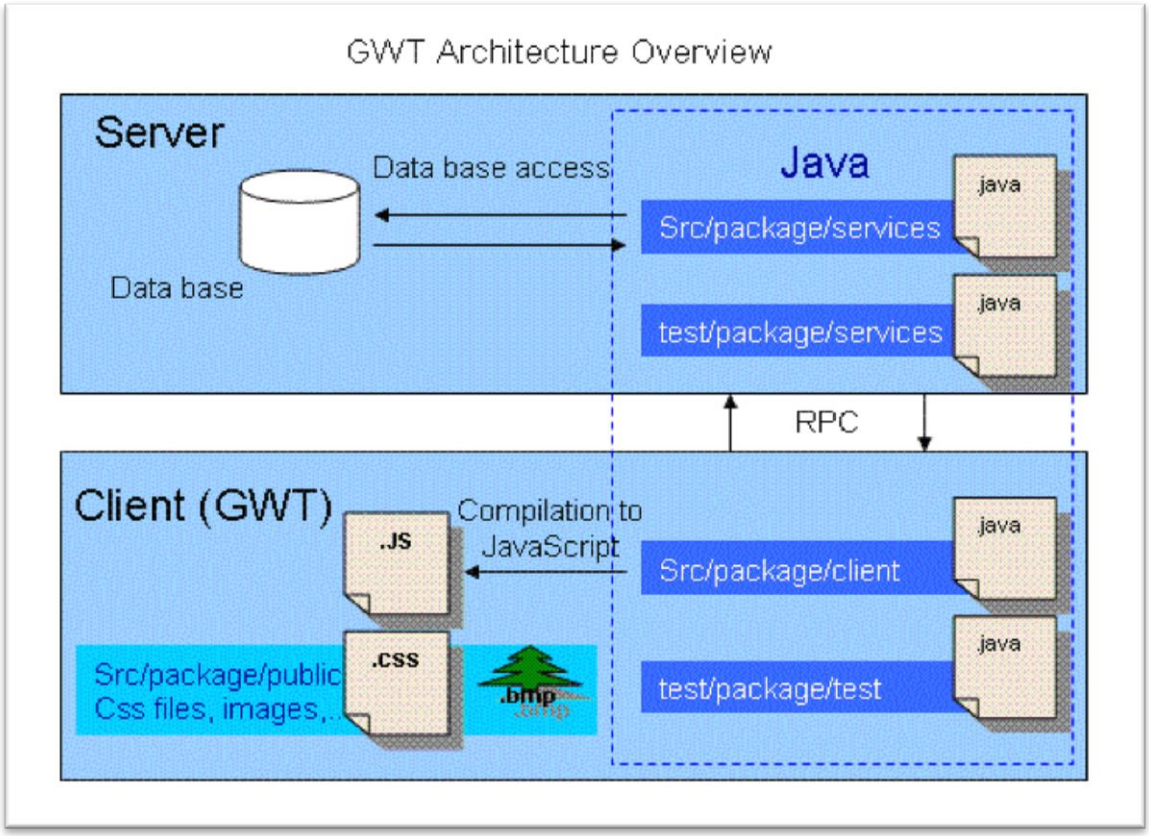
Medzi hlavné výhody Javy patrí objektová orientovanosť (všetky dátové typy okrem základných, sú objektové), interpretovanosť (Java bite code), distribuovanosť (je navrhnutý na podporu sieťových aplikácií), generačná správa pamäti (o pamäť sa stará Garbage collector), robustnosť, bezpečnosť a jednoduchosť.

3.4 MySQL

MySQL je slobodný a otvorený viacvláknový, viacužívateľský SQL relačný databázový systém. Je multiplatformový a je implementovaný vo viacerých jazykoch. Je relačného typu DBMS (database management system). Každá databáza je tvorená jednou alebo viacerými tabuľkami, ktoré môžu byť medzi sebou prepojené. Tabuľky sú reprezentované pomocou riadkov a stĺpcov kde stĺpce určujú dátový typ danej časti záznamu a riadky určujú jednotlivé záznamy. Prístup k dátam je zabezpečený pomocou dotazov, ktoré sú v programovacom jazyku SQL.

3.5 Google Web Toolkit

Google Web Toolkit (GWT) je open source framework pre Javu, ktorý umožňuje vytvárať a udržiavať zložité webové aplikácie komunikujúce so serverom pomocou technológie Ajax. GWT prekompilováva v Jave naprogramované aplikácie do JavaScriptu, čím umožňuje programátorom s existujúcimi vedomosťami v Jave programovať webové aplikácie. GWT pri kompilácií kód aj optimalizuje, čím zabezpečuje jeho najefektívnejšie vykonávanie. Optimalizácia prebieha napríklad odstraňovaním nepotrebných tried a premenných, riadkovaním metód a optimalizovaním stringov. Použitím rozdeľovacích bodov dokáže GWT rozdeliť jednotný kód do viacerých skriptov a tým zrýchliť nábeh webových aplikácií. Vygenerovaný JavaSript je potom kompatibilný so všetkými podporovanými prehliadačmi.



Obrázok 5. Grafické znázornenie frameworku Google Web Toolkit

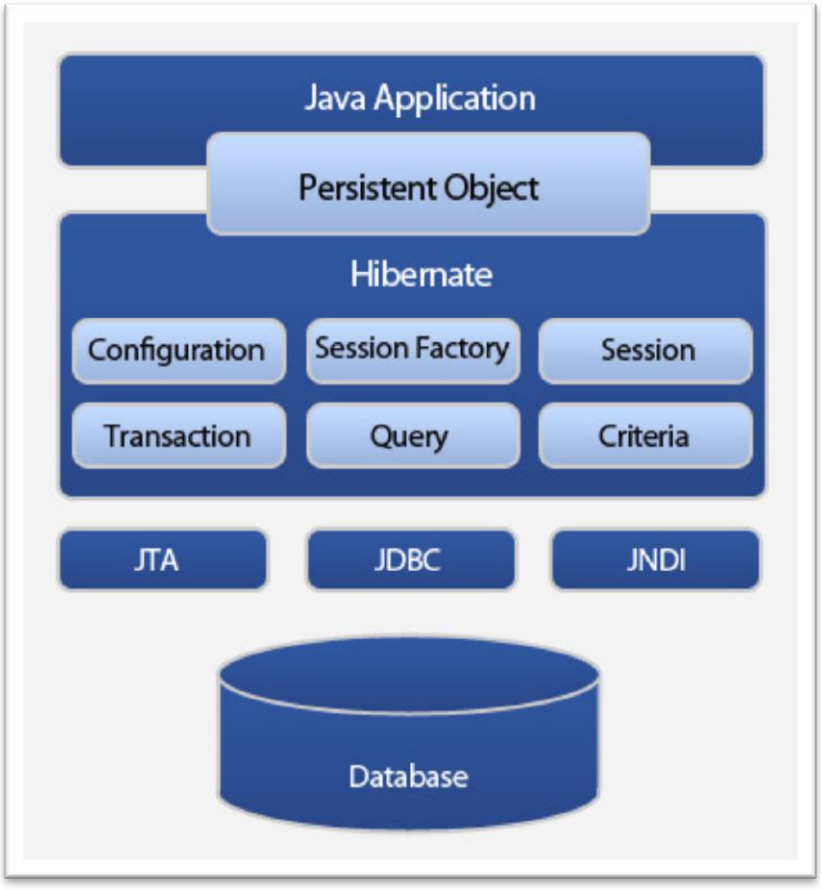
Samotná aplikácia vyvíjaná v GWT má dve verzie, prvá je vo vývojovom móde (*development mode*), kde sa celá aplikácia programuje. V tomto móde sa ešte žiaden kód do JavaScriptu neprekladá, ale webová aplikácia sa priamo zobrazuje v Jave. Vývojár si tak môže upravovať aplikáciu bez toho, aby musel neustále prekompilovávať kód do JavaScriptu, čo mu výrazne uľahčí prácu a čas. Až ked je vývojár so svojím dielom hotový, aplikácia sa presunie do druhej, finálnej fázy. V tejto fáze sa vygeneruje samostatný JavaScript ktorý dokáže obhospodarovať celý web na používateľskej strane.

GWT má aj dve rôzne verzie komunikácie medzi klientskou aplikáciou a serverom. V prvej verzii sa používa technológia Remote Procedure Calls (RPC). Pri tejto verzii sa požaduje, aby na strane servera bežal Java servlet, ktorý vykonáva väčšinu úkonov a do klientskej aplikácie posiela už hotové objekty. Technológia RPC je navrhnutá na spoluprácu s Java servletmi a stará sa o serializáciu tried a objektov a ich posielanie klientskej časti aplikácie. Pri druhej verzii sa údaje posielajú cez XML. V tejto verzii sa väčšina úkonov vykonáva na klientskej strane a so serverom sa komunikuje len pri výmene dát. Táto verzia nepotrebuje na serverovej strane Java servlet ale je náročnejšia na počítač koncového užívateľa.

3.6 HIBERNATE

Hibernate je objektovo-relačný mapovací (ORM) framework pre programovací jazyk Java. Poskytuje programátorovi spôsob, pomocou ktorého sa dá zachovať stav objektov medzi dvoma spusteniami aplikácie, teda udržuje dáta perzistentné. Táto schopnosť sa dosahuje pomocou ORM, čo znamená, že mapuje Javovské objekty na entity vhodné na zapísanie do relačných tabuliek (MySQL). Mapovanie prebieha na základe anotácií alebo mapovacích súborov. V nich je napísané, ako sa majú jednotlivé objekty transformovať do databázy a potom zase z databázy vybrať.

Táto schopnosť dokáže programátorovi ušetriť veľké problémy pri návrhu databázy a návrhu objektov, nakoľko sa nemusí starať o problémy spojené s mapovaním obsahu databázy na premenné v objekte a pod.



Obrázok 6. Grafické znázornenie štruktúry frameworku Hibernate

Ďaľšou výhodou je, že pri výbere objektu z databázy nepotrebuje programátor ručne vyberať všetky objekty, ktoré su niečim spojené s vyberaným objektom, ale Hibernate automaticky zistí, ktoré objekty vybrať treba a programátorovi ich vráti všetky. Toto má

výhodu najmä ak používame objekty, ktoré su vo vzťahu s inými objektami v schéme one-to-many, one-to-one alebo many-to-one. Je taktiež možné nastaviť závislosti objektov a ich poradie, v akom sa majú vyberať, upravovať alebo vkladať do databázy. Na výber z databázy je implementovaný jazyk Hibernate Query Language (HQL), ktorý je odvodený z SQL a je upravený na prácu s objektami v databáze.

Jedinou nevýhodou je, že je nutné používať iba klasické objekty, tzv. POJO (Plain Old Java Object).

3.7 SPRING SECURITY

Spring Security je framework pre Java EE, ktorý umožnuje autentifikáciu a autorizáciu užívateľov. Medzi jeho hlavné výhody patria najmä jednoduchá nakonfigurovateľnosť cez XML a jeho pokročilé bezpečnostné prvky. Framework sa dokáže brániť voči všetkým bežne používaným útokom a vďaka neustálym aktualizáciám dokáže odolľávať aj novým hrozbám.

4. FUNKČNÉ POŽIADAVKY NA APLIKÁCIU

4.1 VÝCHODISKOVÝ STAV APLIKÁCIE

Kedže aplikácia je už rozpracovaná v inej bakalárskej práci, treba si najprv zadefinovať čo z pôvodného plánu už je zrealizované.

Moja predchodkyňa vytvorila základné užívateľské rozhranie so základnou funkcionalitou pre evidenciu projektov, publikácií, konferencií, pracovných ciest a projektov. Tieto kategórie sú prepájateľné a ku každému záznamu sa dá pridať URL adresa, štítok alebo poznámka.

V aplikácii je tiež implementovaný kalendár, ktorý spĺňa základnú funkcionalitu, ale chýba mu prepojenie s internetovou službou Google calendar.

4.2 POŽIADAVKY NA APLIKÁCIU

V aplikácii je toho už vela spraveného, ale chýba niekoľko zložitejších funkcií ktoré sú pre plnohodnotné používanie aplikácie potrebné.

Ako prvé je potrebné spraviť databázu kontaktov a možnosť linkovať ich na ktorúkoľvek existujúcu kategóriu.

Ako druhé je treba spraviť e-mailový systém schopný pripájať sa na e-mailovú schránku, načítať z nej e-maily a analyzovať, či sa jedná o pozvánku na udalosť, call for paper alebo iný, z nášho pohladu nepotrebný mail. Toto je dôležité z toho dôvodu, aby užívateľ nemusel každý e-mail analyzovať samostatne, ale aplikácia to za neho spraví sama.

Ako tretie je potrebné prepojiť existujúci kalendár s službou Google calendar.

A napokon je potrebné dorobiť manažér projektov, ktorý bude fungovať ako nadstavba ostatných častí. Bude ich vedieť spájať do jednotlivých projektov a vytvárať medzi nimi väzby.

4.2.1 KONTAKTY

Kategória kontakty umožnuje ukladanie, manažovanie a upravovanie kontaktov a všetkých informácií o nich, ktoré by používateľ mohol potrebovať. Vie si k nim uložiť osobné informácie ako napr. meno, priezvisko, adresu, prácu, ale aj iné, menej konkrétne informácie typu miesto stretnutia, prezývka, dôvod známosti a podobné údaje.

Taktiež je možné prepájať ktorýkoľvek iný záznam s kontaktami.

4.2.2 E-MAILOVÝ PRIJÍMAČ A PARSER

Tento nástroj, umožnuje prijímanie e-mailov, ich triedenie podla účelu a následné vkladanie udalostí do kalendára.

E-maily sú triedené podla troch kategórií v závislosti od obsahu. Prvá kategória je pozvanie na konferenciu. V tejto kategórii sa nachádzajú e-maily, v ktorých bolo určené miesto, časové rozpätie, názov a prípadne aj téma konferencie. Druhá kategória je call for paper, teda výzva na poslanie článkov. Táto kategória je špecifická tým, že je v nej dátum, do ktorého treba článok poslať, adresa, kam ho poslať a okruh tém, na ktoré by mal odborník písať. Tretia kategória je pre nás nepodstatná, kedže sa jedná o maily, ktoré neobsahujú žiadne pozvánky na konferencie ani výzvy na články.

Po analýze zistení všetkých potrebných údajov bude vytvorená udalosť v kalendári. Táto udalosť bude označená ako automaticky vytvorená udalosť, aby ju užívateľ mohol ľahko identifikovať a prípadne upraviť.

Maily si bude možné pozrieť aj samostatne a identifikovať tie, ktoré by mali spadať do jednej s vyššie uvedených kategórií, ale nachádzajú sa v kategórii ostatné.

4.2.3 PREPOJENIE KALENDÁRA S GOOGLE CALENDAR

Prepojenie interného kalendára so službou Google kalendár na princípe automatickej synchronizácie pri prihlásení sa užívateľa. Pri vytvorení novej udalosti v kalendári aplikácie by sa automaticky vytvorila nová udalosť aj v kalendári Google a vice versa

4.2.4 PROJEKTY

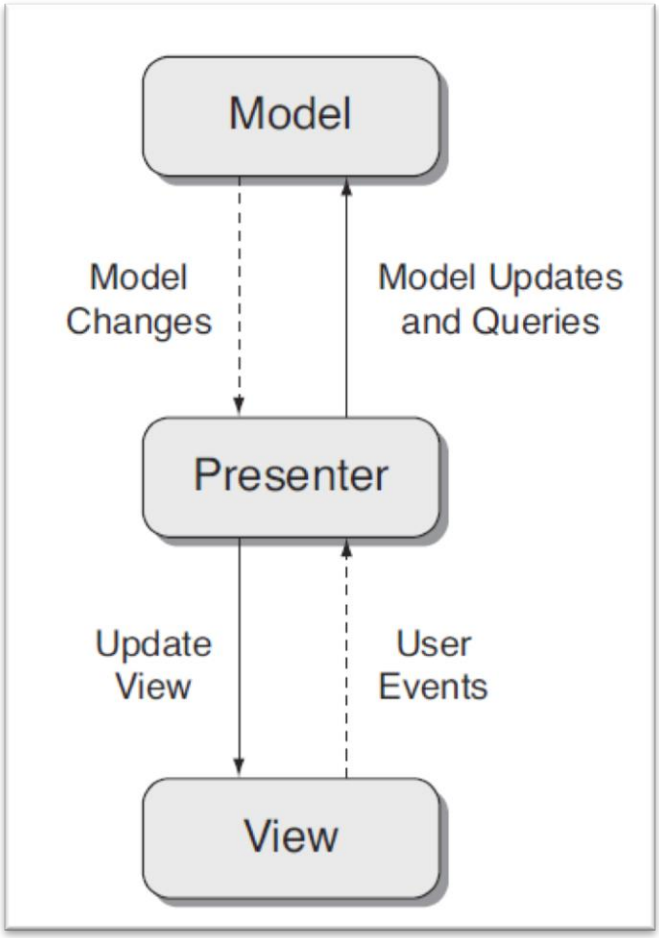
Kategória projekty umožnuje správu projektov, ich vytváranie, upravovanie a mazanie. Každý projekt má základné údaje o sebe a odkazy na ostatné položky aplikácie. Medzi základné údaje o projektoch patrí názov, rozpočet, už vyčerpané finančné prostriedky, dátum začiatku a plánovaný dátum konca. Odkazy na ostané údaje v aplikácií sú riešené spôsobom odkazu z daného záznamu na záznam projektu, pričom pri projekte si užívateľ môže prezerať všetky ostatné záznamy pridružené k projektu. Záznamy pri projekte budú dynamické a priamo z projektu ich bude môcť užívateľ editovať alebo mazať.

5. NÁVRH RIEŠENIA

5.1 ANALÝZA POUŽITÝCH TECHNOLÓGIÍ V PREDCHÁDZAJÚCEJ PRÁCI

Predchádzajúca bakalárska práca položila základ nielen vo funkcionalite, ale aj v technologickom vyhotovení, s ktorým treba pri návrhu tejto práce dopredu počítať kvôli úspešnej implementácii.

5.1.1 MODEL VIEW PRESENTER ARCHITEKTÚRA



Obrázok 7 Grafické znázornenie architekrúry Model View Presenter

Existujúca práca je vytvorená pomocou architektúry Model View Presenter (MVP), ktorá je priamo implementovaná v Google Web Toolkit frameworku (GWT). Architektúra MVP je postavená na nasledujúcich troch vrstvách:

- Dátový model (Model)

- Spracovávač údajov (Presenter)
- Uživateľom videné rozhranie (View)

Prvá časť (Model), je jediná, ktorá sa nachádza na serveri. Zabezpečuje kontakt s databázou, spravovanie dát a ich možnosť odoslania na presenter.

Druhá časť architektúry, presenter, sa stará o spracovanie údajov a udalostí medzi modelom a používateľským rozhraním. Výkonáva takmer všetky operácie nad dátami.

Tretia časť, jediná viditeľná uživateľom, sa stará o zobrazenie dát poskytnutých presenterom na uživateľské rozhranie.

5.1.2 Remote Procedure Calls

Remote procedure calls (RPC) sú zodpovedné za prenos údajov medzi modelom (server) a presenterom (klient). Tento prenos dát prebieha cez HTTP požiadavky, cez ktoré sa posielajú serializované objekty, ktoré je potrebné poslať na server aby sa následne iné, potrebné objekty, vrátili naspäť po spracovaní serverom. Delí sa na tri časti.

Prvá je rozhranie rozšírujúce RemoteService, CommService.java. Táto trieda obsahuje deklarácie všetkých metód ktoré sa dajú zavolať cez RPC mechanizmus.

Druhá trieda, CommServiceAsync.java, rozširuje RemoteServiceServlet a zároveň implementuje triedu CommService.java. Táto trieda obsahuje implementáciu všetkých procedúr ktoré sa volajú cez RPC.

Tretia trieda, časť RPC, je CommServiceAsync.java. Je to asynchrónna trieda ktorá je automaticky priradená ku svojej neasynchrónnej verzii (CommService.java). Táto trieda obsahuje rovnaké metódy ako neasynchrónna verzia, iba s tým rozdielom, že namiesto návratového typu je každá funkcia typu void a k zoznamu argumentov je pridaný argument typu AsyncCallback, ktorý je parametrizovaný pôvodným návratovým argumentom danej metódy a pomocou ktorého sa presenter dostane k návratovej hodnote.

5.1.3 AppController

AppController je trieda spravujúca takmer všetko, čo je rovnaké a nemení sa s každým presenterom a viewom. Vytvára základné uživateľské rozhranie, v ktorom vytvára jednotlivé presentery s príslušným view. Okrem tohto vytvára aj umelú históriu v prehliadači. Umelú preto, lebo aplikácia beží neustále na tej istej webovej adrese a všetky zmeny sa vytvárajú len pomocou AJAXu. Z pohladu používateľa sa ale stránka rozdeľuje na jednotlivé podstránky

a AppController teda vytvára v histórii „falošné“ záznamy pomocou pridávania k URL adresám krátke texty, vďaka ktorým aplikácia pochopí, že sa má zobraziť konkrétny presenter a view.

5.1.4 EVENTBUS

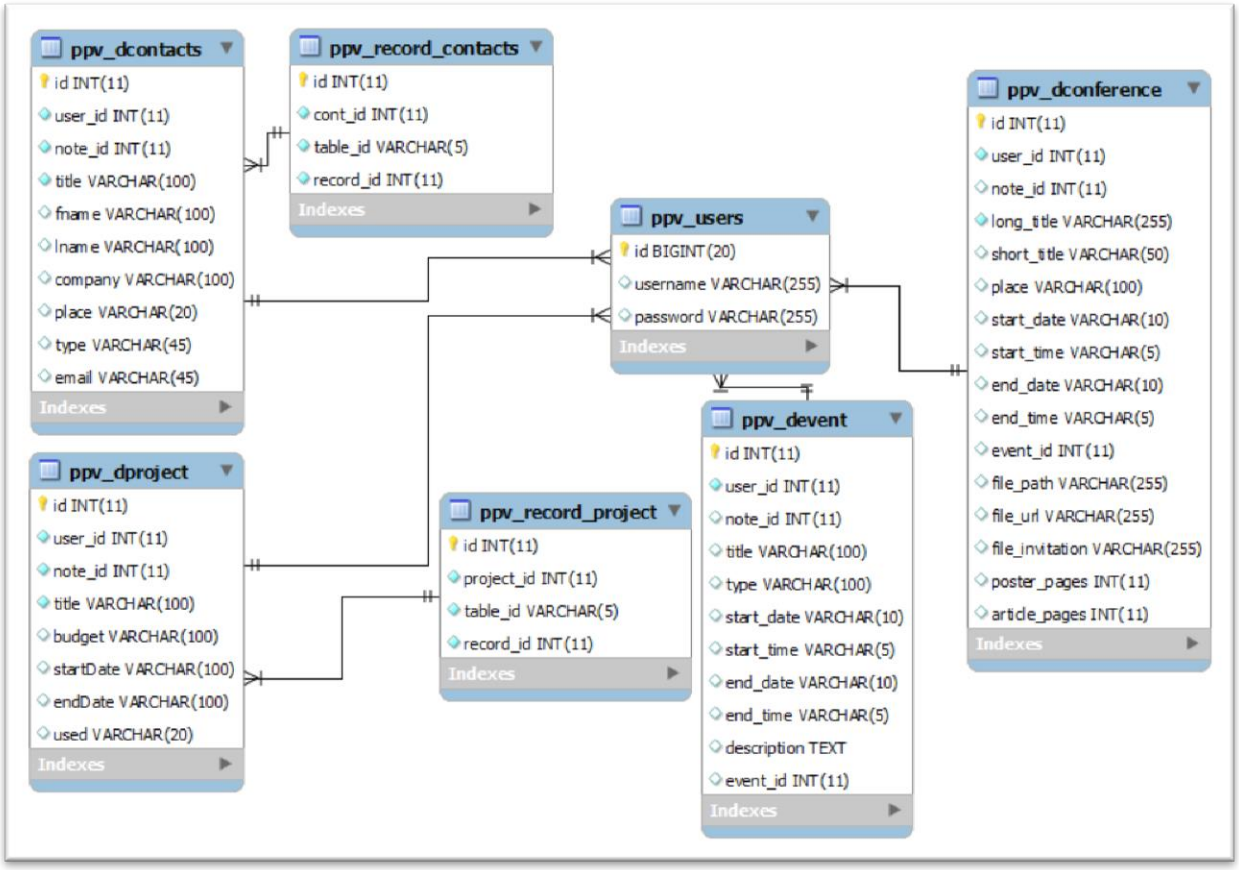
EventBus (premenná typu HandlerManager) je časť aplikácie slúžiaca ako zbernica udalostí. Pomocou tejto premennej spolu komunikuje AppController a jednotlivé presentery. Je potrebný z dôvodu, že prezentery navzájom nemajú informácie o iných častiach aplikácie a potrebujú mať informácie o udalostiach, ktoré v nich nastali.

EventBus funguje nasledovne:

- Pri štarte aplíkácie sa vytvorí inštancia triedy AppControler.java, ktorý následne vytvorí aj EventBus a tento objekt posiela aj ostatným presenterom pri ich vytvorení
- Každý presenter si do EventBusu zaregistruje požadovaný typ udalosti a požadovanú reakciu na danú udalosť
- V prípade nastania danej udalosti, zodpovedná časť aplikácie oznámi zbernici, že nastala udalosť a zbernica o tom informuje všetky prezentery, ktoré si zaregistrovali daný typ udalosti

5.1.5 DATABÁZA

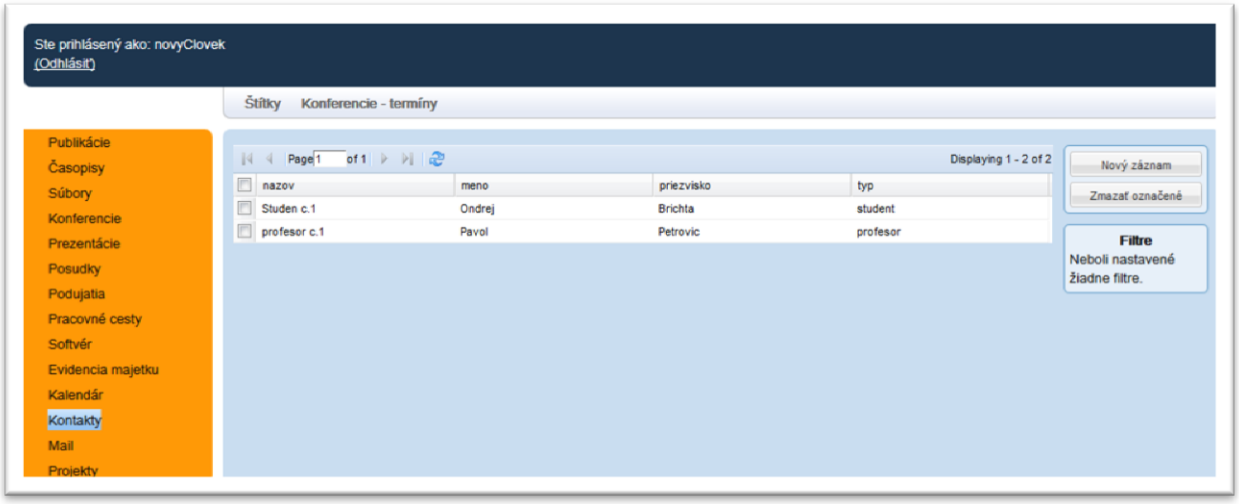
Na obrázku 7 je znázornená časť tabuliek databázy so vzťahmi medzi nimi. Kvôli zachovaniu prehľadnosti sa v ňom nenachádzajú všetky entity databázy, ale len tie, ktoré sú dôležité pre implementáciu funkcií vytvorených v tejto práci. V tabuľktabuľke users sú prihlasovacie údaje o všetkých užívateľoch. Táto tabuľka sa využíva iba pri prihlasovaní, po ktorom sa do globálnych premenných uloží referencia na ID používateľa. Všetky neskoršie údaje sa vyberajú pomocou tohto ID. V tabuľke DContacts sa nachádzajú všetky údaje o kontaktoch. Ku nej pridružená tabuľka, record_contacts, obsahuje všetky referencie na kontakty z iných tabuliek. Tabuľky project a k nej pridružená tabuľka record_project sú dátovou reprezentáciou projektov. Ich implementácia na úrovni databázy je takmer identická, iba majú rozdielny počet a názvy stĺpcov. Tabuľky DEvent a DConference sú reprezentáciami udalostí a konferencií, ktoré sa exportujú z e-mailov.



Obrázok 8. Ukážka vybraných tabuliek z databázy

5.2 KONTAKTY

Kontakty sa rozdeľujú na dve časti. Prvá časť je samotná údajová štruktúra kontaktov, ktorá sa stará o ich vytváranie, ukladanie, zobrazovanie, upravovanie a mazanie. Kontakty sa zobrazujú v jednej tabuľke ktorá má možnosť filtrovania podla ktorejkoľvek položky. V tejto tabuľke sa zobrazujú hlavné údaje o každom kontakte, ako meno, priezvisko, názov kontaktu a typ kontaktu. Zvyšné položky, ako práca, bydlisko, e-mailová adresa a podobne, sa zobrazujú pri rozkliknutí kontaktu, kde ich je možné aj editovať. V tejto časti sa dajú pridávať aj rôzne poznámky, štítky, URL adresy, iné kontakty alebo zaraďovať do projektov.



Obrázok 9. Ukážka kontaktov

Druhá časť, je možnosť pripojenia ktoréhokoľvek záznamu v aplikácii ku konkrétnemu kontaktu. Táto možnosť prepojenia je riešená cez vyššie spomenuté editovacie okno, ktoré je vytvorené pre každý prvok v databáze, teda ktorýkoľvek záznam sa môže prepojiť s ktorýmkoľvek kontaktom. Zároveň je aj možnosť zobraziť pri ktoromkoľvek kontakte všetky iné záznamy, ktoré su na tento kontakt prepojené.

5.3 E-MAILOVÝ KLIENT

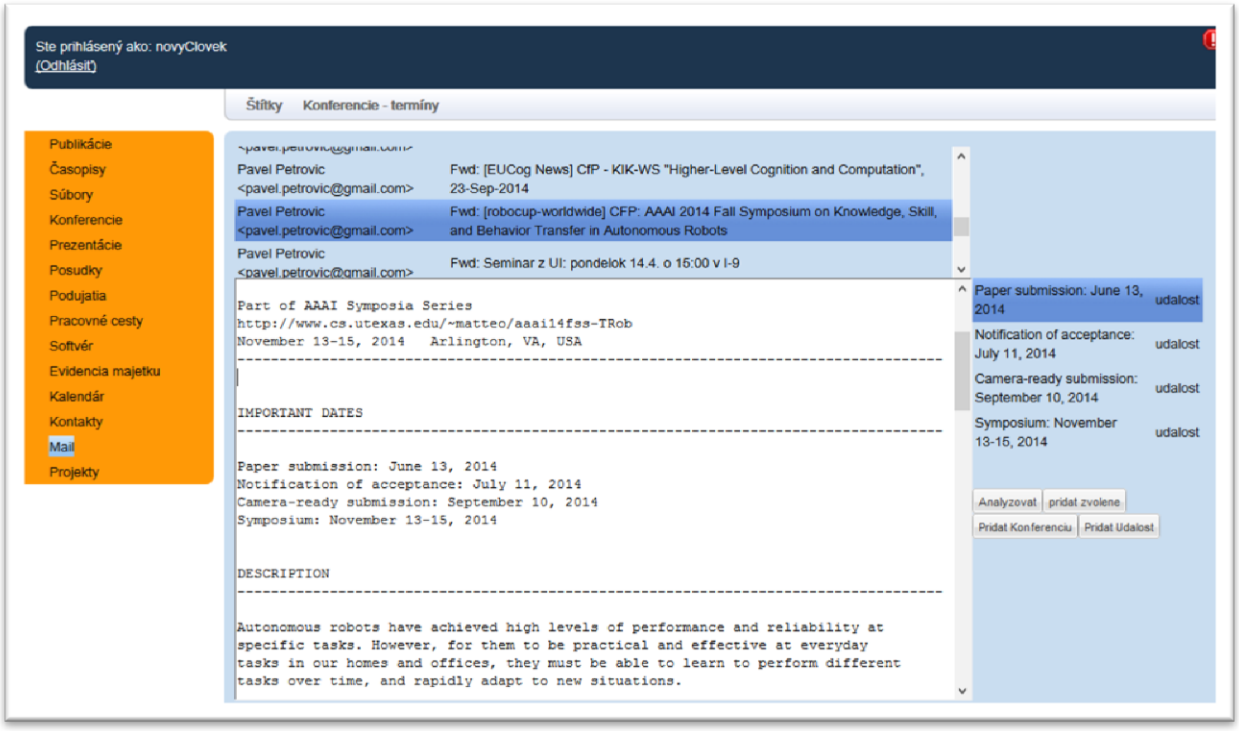
E-mailový klient sa pripája na e-mailovú službu g-mail, pomocou e-mailového konta zadefinovaného pri registrácii. Z e-mailového konta sa pri spustení aplikácie stiahnu všetky nové e-maily a pri otvorení kategórie e-mail sa užívateľovi zobrazia. Zobrazenie je rozdelené na 3 časti.

Prvá časť je tabuľka so zoznamom e-mailov, ktorá sa nachádza navrchu stránky. V tejto časti sa zobrazujú všetky nové e-maily, ktoré ešte neboli spracované.

Druhá časť, text e-mailu, sa zobrazuje pod zoznamom e-mailov, a vždy sa tu zobrazuje e-mail, ktorý si používateľ zvolil. V tejto časti je možné e-mail akokoľvek modifikovať, aby sa zlepšila jeho čitateľnosť alebo funkčnosť tretej časti.

Tretia časť, dynamické vytváranie podujatí a konferencií, sa nachádza v pravej časti stránky. V tejto časti sa nachádzajú tri hlavné prvky: panel s tlačidlami, zoznam analyzovaných udalostí a konferencií a vstupné parametre pre vytvorenie novej konferencie alebo udalosti. Panel s tlačidlami obsahuje tlačidlá na analyzovanie e-mailu, vytvorenie novej konferencie, vytvorenie novej udalosti a zmazanie e-mailu. Zoznam analyzovaných udalostí

je scrollovateľná tabuľka obsahujúca všetky analyzátorom navrhnuté udalosti k danému e-mailu a všetky užívateľom vytvorené udalosti k danému e-mailu. Tieto udalosti sa zobrazujú v tabuľke až kým ich uživateľ neodsúhlasí. Pri kliknutí na ktorýkoľvek prvok v tabuľke z analyzovaných udalostí sa táto vyplní do predpripravených koloniek na vytvorenie novej udalosti alebo konferencie. Po prípadnej úprave zo strany užívateľa sa vytvorí záznam s udalosťou alebo konferenciou, ktorý sa vloží do databázy.



Obrázok 10. Ukážka e-mailového klienta

V e-mailovom klientovi je aj jedna neviditelná časť a tou je automatické vyhľadávanie dátumov v e-maili a ich vloženie do zoznamu analyzovaných udalostí a konferencií. Táto časť sa spustí pri stlačení tlačidla analyzovať. Po spustení sa zoberie telo e-mailu a postupným prechádzaním e-mailu sa z neho vyexportujú dátumy s názvami a miestami udalostí.

V každom pozývacom e-maili sa nachádzajú obvykle dva možné zápisy dátumov s udalosťami. Prvý, jednoduchšie nájditeľý, je vo veľmi ľahko čitateľnej forme. Niekde v rámci e-mailu sa nachádza nadpis „Important Dates“ a pod ním sa nachádza niekolko riadkov. Každý z týchto riadkov reprezentuje zvyčajne jednu udalosť.

Obrázok 11. Ukážka zápisu udalostí v e-maile-maili

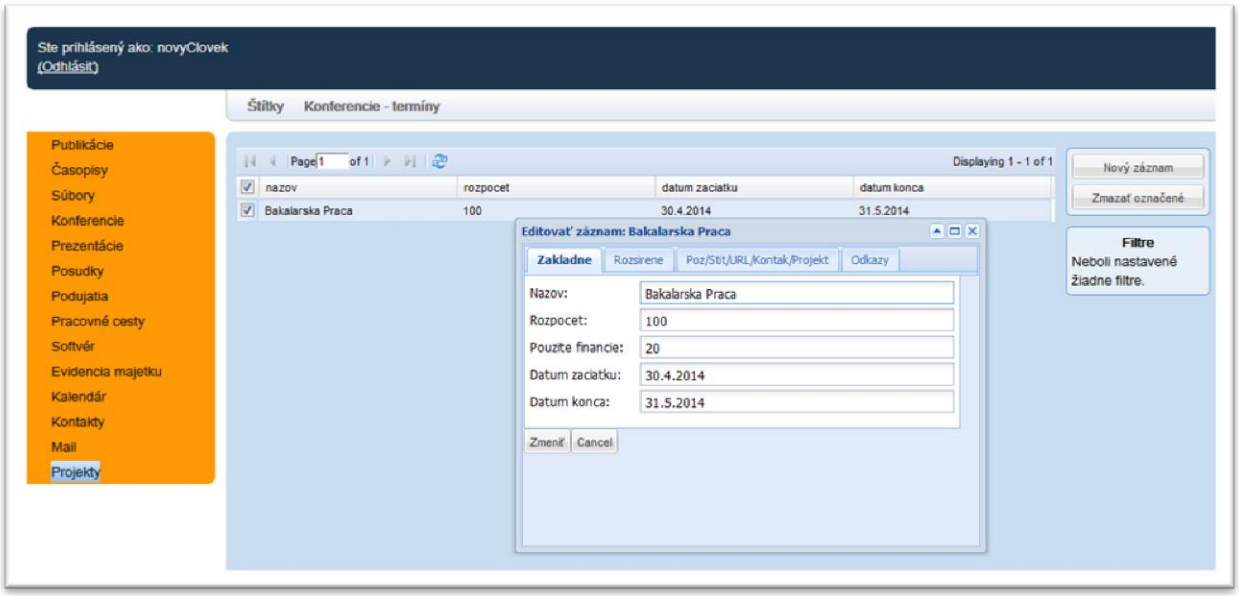
Druhá možnosť je, keď je dátum označený voľne v e-maili a zvyšné detaily o udalosti sú niekde v okolitých riadkoch. Táto možnosť, vzhľadom na náročnosť a nejdenoznačnosť textu, je zatiaľ vynechaná z automatického vyhľadávania dátumov.

5.4 PROJEKTY

Projekty sa rozdeľujú, podobne ako kontakty, na dve časti. Prva časť je o vytváraní, zobrazovaní a manažovaní kontaktov, druhá je o ich priraďovaní ku ktorýmkoľvek údajom v aplikácii.

V prvej časti sa projekty zobrazujú v tabuľke, kde sa zobrazujú ich hlavné časti, ako názov, rozpočet, čas začiatku a čas skončenia. V rozšírenej časti sa zobrazujú položky ako popis projektu, zodpovedná osoba a percentuálne zobrazenie štádia projektu.

Druhá časť umožnuje zaraďovanie ktoréhokoľvek údaju v aplikácii do konkrétneho projektu, prípadne aj do viacerých projektov naraz.



Obrázok 12. Ukážka projektov v móde editácie záznamu

6. IMPLEMENTÁCIA

6.1 ZÁVAŹNÉ PROBLÉMY PRI IMPLEMENTÁCII

Vzhľadom na to, že táto bakalárska práca je pokračovaním inej bakalárskej práce, ktorá bola vytvorená pred 3 rokmi, bolo potrebné najprv dôkladne preštudovať predchádzajúcu prácu a sfunkčniť samotnú aplikáciu.

Prvý problém, s ktorým ale bolo počítané, nastal pri prvom pokuse o sfunkčnenie aplikácie, keďže väčšina frameworkov na ktorých je aplikácia postavená, sa podstatne zmenila vo funkcionalite a len niektoré z nich podporujú aj štruktúry navrhnuté pre staré verzie. Tento problém sa ale podarilo eliminovať zohnaním starších verzií frameworkov alebo zmenením častí kodu.

Druhý problém nastal, keď z neznámeho dôvodu nebolo možné vybrať niektoré dáta z databázy počas vývojového režimu. Tento problém nastával vždy pri spustení aplikácie a pretrvával pri každom spustení inú dobu. Po niekoľkodňovom hladaní chyby, sa podarilo prísť na okľuku, ktorou pri vývojovom režime chyba nenastávala. Na túto okľuku je potrebné získať starú verziu internetového prehliadača Firefox, vo verzii 24, a nainštalovať do neho prvok GWT developmer plugin (2). Keď je táto podmienka splnená, treba spúšťať aplikáciu cez develpment mode a zamedziť prístup cez iné prehliadače.

6.2 KONTAKTY

Implementácia kontaktov, sa rovnako ako funkčnosť, delí na dve časti.

Prvá časť sú samotné kontakty, ich vytváranie, manažovanie a zobrazovanie v tabuľke ako samostatné prvky. Na dosiahnutie tohto treba vytvoriť samostatnú triedu DContacts.java so všetkými potrebnými metódami, k nej vytvoriť mapovací xml súbor DContacts.hbm.xml a entitu v databáze s požadovanými vlastnosťami. Po zaregistrovaní tohto dátového modelu do centrálneho xml súboru pre framework Hybernate a daľšieho navigačného xml súboru pre AppControler kvôli zaradeniu do navigácie aplikácie, sa o samotné vykreslenie tabuľky s údajmi, pridávanie, upravovanie a mazanie údajov stará centrálny presenter, ktorý na vytváranie volá NewRecordPresenter.java a na upravovanie EditRecordPresenter.java.

Druhá časť je pridávanie odkazov na kontakty z ľubovoľného záznamu v aplikácii. Táto časť je implementovaná pomocou vytvorenia dátového modelu pre prepojenie kontaktu s iným záznamom v databáze, rozsiahlych úprav v RecordRefIPresenter a k nemu pridruženému

view, kde bolo treba vytvoriť niekoľko desiatok procedúr, ďalej úpravou existujúcich a vytvorením niekoľkých nových procedúr v CommService a jeho pridruženej asynchrónnej verzii.

Vytváranie referencií na kontakty funguje nasledovne:

- Pri vytváraní nového alebo úprave ľubovolného záznamu v aplikácii sa v editore záznamu (RecordRefIPresenter + RecordRefIView) vytvorí záložka s možnosťou pridania jedného z existujúcich kontaktov
- Z databázy sa pomocou EventBus a RPC vyberú všetky existujúce údaje o prepojení daného záznamu s niektorým z kontaktov a zároveň všetky údaje o každom kontakte
- Užívateľovi sa zobrazí označovateľný zoznam všetkých kontaktov, kde sa načítavajú dáta z databázy kontaktov. Pri ukladaní referencie na kontakt sa pri zázname uloží iba ID kontaktu, aby sa v prípade zmeny kontaktu zmenila aj referencia
- Po pridaní alebo odobratí kontaktu sa cez EventBus a RPC vyšle príkaz o uložení celého záznamu, v CommServiceImpl sa zo všetkých požadovaných tabuliek v databáze najskôr odstránia všetky existujúce záznamy relevantné k danému záznamu a potom sa pridajú všetky nové aj existujúce záznamy, ktoré užívateľ nezmazal. Toto sa deje z dôvodou, aby v databáze nenastalo viacnásobné vloženie toho istého prvku pri jeho miernej modifikácii.
- Cez RPC sa vráti odpoveď o úspešnom vložení záznamu alebo o zlyhaní a uživatelská časť sa podla toho zachová. V prípade úspechu zobrazí modifikovaný záznam v tabuľke danej kategórie a v prípade neúspechu ostane otvorené modifikačné okno záznamu a užívateľ bude oboznámený s neúspechom ukladania záznamu.

```
public List<ContTableItem> getRecordConts(String className, Long recordId) throws Exception {
    Session session = beginSession();
    Query query = session.createQuery("from RecordContact where record_id=:record_id and table_id=:table_id");
    query.setParameter("record_id", recordId);
    query.setParameter("table_id", getTableId(className));
    List<RecordContact> list = query.list();
    List<ContTableItem> res = new ArrayList<ContTableItem>();
    for (RecordContact cont : list) {
        res.add(new ContTableItem(cont.getCont_id(), cont.getId()));
    }
    endSession(session);
    return res;
}
```

Obrázok 13. Ukážka vyberania odkazov na kontakty z databázy

6.3 E-MAILOVÝ KLIENT

Pri implementácii e-mailového klienta bolo potrebné vytvoriť nový presenter (MailPresenter.java) a k nemu aj potrebné view (MailView.java). Z dôvodu častej komunikácie medzi view a presenterom, je v presenteri vytvorený interface, ktorý je následne implementovaný vo view. Tým pádom máme zaručený jednoduchý prístup k potrebným dátam od užívateľa priamo z presentera.

Celý e-mailový klient je rozdelený na dve časti, prvá sa stará o načítanie e-mailov zo servera googlu a druhá sa stará o vytiahnutie potrebných dát z e-mailu a následné vloženie do databázy pod správnym typom záznamu.

6.3.1 NAČÍTANIE DÁT ZO VZDIALENÉHO SERVERA

Prvá časť, načítanie dát, sa deje pomocou knižnice JAVAMAIL API (3). Na načítanie e-mailov sa používa pokročilý protokol IMAP, a to z dôvodu, aby sa nemusel načítavať celý e-mail aj s prílohami, ale iba potrebné časti. Iné e-mailové protokoly toto neumožňujú. Pri implementácií tohto protokolu ale nastal závažný problém, keď sa zistilo že, JAVAMAIL API má problém s sťahovaním e-mailov z služby GMAIL v prípade, že sa zároveň používa framework SpringSecurity. Riešenie (4) z oficialnej web stránky podpory JAVAMAIL API ale funguje.

Po načítaní dát sa dáta pošlú do klientskej časti, kde sa spracujú a všetky e-maily sa zobrazia v tabuľke. Tabuľka je interaktívna a po kliknutí na ktorýkoľvek e-mail sa jeho obsah zobrazí v textovom poli pod tabuľkou. Vedľa obsahu e-mailu sa zobrazia možnosti pre užívateľa. Môže buď manuálne pridať udalosť, kde sa mu len otvorí nový formulár na vkladanie udalostí alebo konferencie, alebo využije možnosť automatického detekovania udalostí a konferencií v e-maili.

6.3.2 AUTOMATICKÁ DETEKCIA UDALOSTÍ A KONFERENCIÍ V TEXTE

Na vyhľadávanie a export udalostí a konferencií z textu e-mailu, je vytvorená podtrieda v triede MailView. Podtrieda bola použitá namiesto samostatnej triedy z dôvodu prehladnejšej štruktúry kódu a takisto z dôvodu jednoduchšieho pristupovania k tejto triede a jej objektom. Trieda má len veľmi málo verejne prístupných metód, a to z dôvodu prehľadnosti kódu a jednoduchšej vnútornej manipulácii s dátami. Zmyslom triedy je nájdenie, určenie, vytvorenie a vrátenie udalostí, ktoré sa nachádzajú voľne v texte e-mailu. Pre zjednodušenie, je zameraná hlavne na e-maily majúce jednoznačný zámer, a to pozvanie výskumníka na konferenciu alebo požiadanie o zaslanie článku do odborného časopisu.

Vzhľadom na tento fakt, je možné sa zamerať na presné formy textu, ktorý treba v e-maili hladať (viď obr. 8).

Postup vyhladávanie udalostí v texte:

- Vytvorí sa objekt triedy parser, so vstupným údajom text. Tento údaj sa uloží do objektovej premennej.
- Objektová premenná sa rozdeĺí na jednotlivé riadky, z dôvodu jednoduchšej manipulácie s textom.
- Zistí sa, či je e-mail preposlaný (čo zväčša je, pretože e-mailový klient sa zvyčajne nepripája priamo na e-mailový účet výskumníka, ale na samostatný účet vytvorený priamo pre účely aplikácie) alebo došiel priamo na túto e-mailovú adresu. Toto sa zisťuje podla typickej hlavičky, ktorá sa nachádza na začiatku. V prípade že existuje, sa táto hlavička odstráni, z toho dôvodu, že v sebe obsahuje dátum, ktorý by sa neskôr mohol nesprávne vybrať ako termín na dodržanie alebo dátum konferencie. Vzhľadom na to že, e-mail mohol prechádzať niekoľkými ľuďmi, táto procedúra odstráni všetky hlavičky svedčiace o preposlaní z celého e-mailu.
- Z e-mailu sa odstránia všetky nepotrebné riadky, v ktorých nie je žiaden text, alebo celé pozostávajú iba z jedného opakujúceho sa znaku. Vždy sa ale medzi odstavcami nechá jeden voľný riadok, na lepšiu identifikáciu.
- Vyhľadá sa textový reťazec „Important dates“. Vzhľadom na rôzne mutácie tohto slovného spojenia sa text vyhľadáva aj v mierných obmenách. Poloha riadku v ktorom sa našiel tento text sa uloží do premennej pre neskoršie využitie.
- Vyhľadá sa najbližší voľný riadok. Toto sa môže hladať z dôvodu, že po informácii o dôležitých dátumoch je vždy odsek, ktorý čitateľovi tieto dôležité dátumy opticky zviditeľní a spraví e-mail čitateľnejším. Poloha tohto riadku sa tiež vloží do premennej pre neskoršie použitie.
- Ak obe predchádzajúce čísla boli nájdené a nie je medzi nimi rozdiel rovný jeden, tým pádom sa nejedná o po sebe nasledujúce riadky, sa vytiahnu všetky riadky medzi týmito dvoma riadkami a pošlú sa ďalej. V prípade, že sa tieto riadky nachádzajú priamo pod sebou, hľadá sa daľší prázdny riadok dovtedy, kým sa nenájde aspoň jeden riadok s údajmi medzi týmito riadkami. To je

z dôvodu občasného zasadenia jedného alebo viacerých prázdnych riadkov medzi nadpis ohlasujúci dôležité dátumy a samotné záznamy o dátumoch.

- Po najdení týchto údajov, sa vytvorí prechodný záznam o udalosti, ktorý pri vytvorení dostane celý riadok, v ktorom sa nachádzajú detaily o udalosti.
- Prechodný záznam má päť premenných, názov, typ udalosti a začiatok a koniec udalosti. Do prvých štyroch premenných sa rozdelí celý riadok a v prípade že sa jedná o konferenciu, teda má v sebe aj názov miesta konania, sa vloží údaj aj do poslednej premennej.
- Tieto objekty sa vrátia naspäť do MailView na neskoršie spracovanie.

V prípade, že sa v texte e-mailu nachádzjú aj iné dátumy, tieto sú ponechané pre ručnú analýzu samotným užívateľom aplikácie, z dôvodu prílišnej zložitosti extrahovania informácií.

6.3.3 VYTVÁRANIE ZÁZNAMOV Z EXTRAHOVANÝCH ÚDAJOV

Pre vytvorenie udalostí alebo konferencií z extrahovaných údajov bolo potrebné vytvoriť niekoľko samostatných asynchrónnych procedúr a správne využiť niektoré z existujúcich.

Technický postup pri vytváraní záznamov:

- V tabuľkovom objekte typu CellList je zoznam vyexportovaných udalostí z parsera (viď ods. 6.3.2).
- Po vybratí niektorého z prvkov zoznamu a stlačení tlačidla „vložiť vybraný prvok“ sa vytvorí objekt DEvent alebo DConference odvodený od objektu GeneralDataClass.
- Pomocou údajov z tohto objeku sa vytvorí pole dvojíc textu, kde prvý prvok určuje v akom formáte a pod akým názvom sa má ukladať prvok druhý.
- Vytvorí sa asynchrónny dopyt na vytvorenie novej udalosti, do ktorého sa ako parameter posiela daný objekt spolu s celkovým textom e-mailu ako objekt typu poznámka.
- Na serveri sa tieto údaje spracujú a vložia do databázy.
- Cez asynchronnu komunikáciu sa vráti ID záznamu. Toto je zároveň známkou toho, že sa záznam úspešne uložil a zároveň je táto položka použitá na otvorenie editačného módu pre daný záznam.

- V okne pre editáciu si môže užívateľ akokoľvek upraviť daný záznam a znova ho uložiť. Pri ukladaní záznamu sa najprv starý záznam vymaže a následne sa vloží záznam nový.

```
view.GetAddEventBtn().addSelectionListener(new SelectionListener<ButtonEvent>() {
    public void componentSelected(ButtonEvent ce) {
        commService.getCategoryName("rd.client.model.DEvent",
                new AsyncCallback<String>() {
            public void onFailure(Throwable caught) {
                caught.printStackTrace();
                eventBus.fireEvent(
                        new ShowErrorEvent("Chyba: neidentifikovatelna kategoria."));
            }
            public void onSuccess(String result) {
                eventBus.fireEvent(new NewRecordWindowEvent("DEvent", result));
            }
        });
    }
});
view.getAddSelected().addSelectionListener(new SelectionListener<ButtonEvent>() {
    public void componentSelected(ButtonEvent ce) {
            commService.addNewRecord(view.getRecordToSave().getClass().getSimpleName(),
                    view.getRecordToSave().getClass().getName(),
                    getDataValuePairs(view.getRecordToSave()), view.getSelectedMailBody(),
                    new ArrayList<ModelData>(),new ArrayList<UrlTableItem>(),
                    new ArrayList<ContTableItem>(), new ArrayList<ProjectTableItem>(),
                    new ArrayList<QuickRecordInfo>(), new AsyncCallback<GeneralDataClass>() {
            public void onFailure(Throwable caught) {
                caught.printStackTrace();
                eventBus.fireEvent(new ShowErrorEvent("Chyba pri ukladani noveho zaznamu"));
            }
            public void onSuccess(GeneralDataClass result) {
                eventBus.fireEvent(new ShowInfoEvent("Zaznam bol pridany"));
                eventBus.fireEvent(new ShowEditWindowEvent(view.getRecordToSave().getClass()
                                                    .getSimpleName(), (long) result.getId()));
                }
        });
    }
});
```

Obrázok 14. Ukážka vyvolania asynchrónnej udalosti na vytváranie novej udalosti alebo konferencie.

6.4 PROJEKTY

Implementácia projektov je veľmi podobná implementácii kontaktov. Delí sa na dve časti, ktoré sú implementované takmer úplne nezávisle. Prvá časť je rovnako ako pri kontaktoch implementovaná pomocou už existujúcich funkcií a metód, ktoré stačí iba správne zavolať a nastaviť niekoľko xml dokumentov. Je ale potrebné vytvoriť dátový model pre projekt, čo je prvá vec kde sa projekty podstatnejšie líšia od kontaktov, pretože majú výrazne odlišnú štruktúru potrebných údajov. Druhá časť, je možnosť vkladať ktorékoľvek údaje do daného projektu. Táto časť je implementovaná pomocou rozsiahlych úprav v RecordRefIPresenter.java a k nemu pridruženému view RecordRefIView.java a taktiež bolo potrebné

vytvoriť a implementovať niekoľko funkcií v CommService a v jej asynchrónnej a implementačnej časti.

V RecordRefIPresenter bolo potrebné vytvoriť funkcie na vyberanie zoznamu projektov z databázy a vyberanie a vkladanie referenčných vzťahov zo serverovej časti aplikácie. Tieto funkcie volajú asynchrónne funkcie z CommService a ich výsledky spracovávajú a posielajú dalej do RecordRefIPresenter pomocou metód, ktoré sú implementované cez interface Display. V RecordRefIView sú tieto údaje vložené do grafického prostredia, kde sa zobrazia užívateľovi, ktorý ich môže modifikovať. Po modifikácii sa následne údaje uložia naspäť do databázy cez spoluprácu presentera a asynchrónnej komunikácie so serverom.

```
//tabulka project s naplenymi udajmi
public void createProject(List<ProjectTableItem> nameProjectIdList) {
    createProject();
    for (ProjectTableItem c:nameProjectIdList) {
        c.setName(getProjectById(c.getProject_id()).getTitle());
    }
    projectTable.getStore().removeAll();
    projectTable.getStore().add(nameProjectIdList);
}
private DProject getProjectById(Long id){
    for (DProject c:projects){
        if (c.getId()==id) return c;
    }
    return null;
}
@Override
public void createProjectsToSelect(ArrayList<GeneralDataClass> project) {
    projects = new ArrayList<DProject>();
    for (int i = 0; i < project.size(); i++) {
        projects.add((DProject)project.get(i));
    }
}
```

Obrázok 15. Ukážka procedúr zobrazujúcich projekty na výber

7 MOŽNOSTI MODIFIKÁCIE DO BUDÚCNOSTI

V tejto bakalárskej práci sa nepodarilo vytvoriť všetko, čo treba na to, aby aplikácia mohla fungovať plnohodnotne. Chýba niekoľko dôležitých prvkov. Pre informáciu čitateľa uvediem zopár najdôležitejších.

- Kalendár: Prepojenie kalendára so službou Google calendar. Tento prvok mal byť pôvodne súčastou tejto práce ale kvôli inicíálnemu zdržaniu a po zistení rozsiahlosti daného problému, sa zistilo, že vytvorenie tohto prepojenia je príliš časovo náročné, keďže by si vyžadovalo výrazné zásahy do existujúcej aplikácie.
- Súbory: možnosť pridávať súbory priamo k jednotlivým prvkom v aplikácii podobnou formou ako je teraz riešené pridávanie poznámok alebo URL adries.
- Logovací systém: vytvoriť logovací systém, kde by sa zaznamenávali všetky udalosti, ktoré sa udiali v aplikácii, pričom by sa dali v rámci aplikácie prezerať.
- Záloha a recovery: Pridať možnosť vytvorenia zálohy celej databázy a jej obnovu z rôznych časových bodov uloženia. Prípadne prepojiť s logom a vytvoriť možnosť vracať len vybrané zmeny z logu.
- Vyhľadávanie: Aplikácii momentálne chýba fulltextové vyhľadávanie cez všetky položky v databáze.
- Drobné úpravy existujúcej funkcionality: Aplikácia sa nachádza stale len v štádiu vývoja. Tým pádom zatial nie je veľmi používateľsky prívetivá, keďže sa jej vývoj zatiaľ zameriava na základnú funkcionalitu. Bolo by vhodné upraviť niektoré grafické prvky, možnosti zapamätania si rôznych lokálnych nastavení a pod.

8 INŠTALÁCIA A SPUSTENIE

Aplikácia na spustenie vyžaduje aplikačný server napríklad s GlassFish alebo Tomcat s databázou.

Na priloženom CD sa nachádzajú tri súbory:

- PracPlochaVyskumnika.sql: tento súbor obsahuje vyexportovanú databázu. Tento súbor je potrebné importovať na databázu, ku ktorej sa server vie pripojiť. Názov databázy a prístupové údaje je treba neskôr nastaviť v konfiguračnom súbore aplikácie.
- PracPlochaVyskumnika.zip: tento súbor obsahuje všetky zdrojové kody aplikácie. V prípade modifikácie je potrebné tento súbor rozbaliť a jeho obsah importovať do programu Eclipse alebo jeho eqvivalentu s nainštalovaným rozšírením Google web toolkit SDK (5). V ňom po úpravách treba vyexportovať inštalačný súbor pre server.
- PracPlochaVyskumnika.war: tento súbor je inštalačným súborom pre aplikáciu. Treba ho umiestniť na server a nainštalovať (Deploy).

Pred spustením aplikácie (po nainštalovaní) je potrebné upraviť konfiguračný súbor pre prístup k databáze, hibernate.cfg.xml. Tento súbor sa nachádza v priečinku {inštalačná adresa na serveri} /PracPlochaVyskumnika/WEB-INF/classes. V súbore treba zmeniť nasledujúce riadky:

```
<property name="connection.driver_class">org.gjt.mm.mysql.Driver</property>
<property name="connection.url">jdbc:mysql://localhost/databaseName</property>
<property name="connection.username">databaseUser</property>
<property name="connection.password">password</property>
```

- driver_class je typ databázy, na ktorú sa aplikácia pripája
- connection_url je adresa databázy
- username je uživateľské meno, pod ktorým sa na databázu pripája
- password je heslo, s ktorým sa na databázu pripája

Po tomto nastavení je nutné aplikáciu spustiť. Po spusteni je prístupná na adrese {adresa server}/PracPlochaVyskumnika.

9 ZÁVER

Cieľom tejto bakalárskej práce bolo preštudovať existujúcu prácu, analyzovať iné, podobné programy, porovnať ich a dorobiť do existujúcej aplikácie chýbajúcu funkcionalitu.

V práci som porovnával niekoľko existujúcich aplikácií, ktoré majú podobnú funkcionalitu ako Pracovná plocha výskumníka. Ako sa však ukázalo, ani jedna z týchto aplikácií nespĺňala požiadavky, ktoré boli pre moju aplikáciu stanovené.

Pred vývojom som musel dôkladne preštudovať existujúcu prácu s jej dokumentáciou a dokumentáciu niekoľkých frameworkov, ktoré boli v danej práci použité. Okrem samotných frameworkov bolo potrebné pochopiť, ako fungujú jednotlivé modely a postupy, ktoré sa v aplikácii nachádzali. Bolo potrebné sa najmä naučiť, ako funguje Google web toolkit, v ktorom bola celá aplikácia vytvorená a architektúru Model View Presenter, na ktorej aplikácia fungovala.

Podarilo sa mi vytvoriť funkčné a dôležité moduly do aplikácie, ktoré sú pre celkovú použiteľnosť vitálne. Pridal som možnosť vkladať kontakty a projekty a prepájať ich s ktorýmkoľvek prvkom v databáze. Vytvoril som e-mailový systém, ktorý automaticky analyzuje e-maily a vytvára udalosti a konferencie v aplikácii na základe údajov v týchto e-mailoch.

Aplikácia, ktorú som vyprodukoval síce nie je úplne hotová, ale už teraz sa dá plnohodnotne používať a je ľahko modifikovateľná. V prípade, že by sa niekto iný rozhodol pokračovať v aplikácii, spísal som väčšinu potrebných modifikácií na to, aby aplikácia bola kompletne dokončená. Open sourcová licencia aplikácie umožňuje zásahy do kódu podľa aktuálnych a meniacich sa požiadaviek.

Celkovo si teda myslím že sa mi podarilo vytvoriť aplikáciu, ktorá spĺňa väčšinu požiadaviek, ktoré som si zadal a nespochybniteľne bude veľmi dobre slúžiť výskumníkovi, ktorému ušetrí množstvo času a zlepší jeho organizáciu.

POUŽITÁ LITERATÚRA A ZDROJE

1. **Matysová, Katarína.** *Pracovná Plocha Výskumníka.* Bratislava : Fakulta Matematiky, Fyziky a Informatiky Univerzity Komenského, 2011.

2. GWT plugin. [Online] https://dl-ssl.google.com/gwt/plugins/firefox/gwt-dev-plugin.xpi.

3. JAVAMAIL API Documentation. [Online] https://javamail.java.net/docs/api/.

4. JAVAMAIL IMAP bug . [Online] http://www.oracle.com/technetwork/java/javamail/faq/index.html#imapserverbug.

5. Google web toolkit SDK. [Online] http://www.gwtproject.org/download.html.

6. Google Web Toolkit Documentation. [Online] http://www.gwtproject.org/doc/latest/DevGuide.html.

7. GWT Event system. [Online] https://code.google.com/p/google-web-toolkit-incubator/wiki/GwtEventSystem.

8. GWT developer blog. [Online] http://googlewebtoolkit.blogspot.sk/.

9. Spring Security. [Online] http://projects.spring.io/spring-security/.

10. EXT GWT. [Online] http://www.sencha.com/products/gxt/.

11. JavaScript. [Online] http://www.w3schools.com/js/DEFAULT.asp.

12. GWT Project. [Online] https://code.google.com/p/google-web-toolkit/.

13. **Boodhoo, Jean-Paul.** Model-View-Presenter. [Online] http://msdn.microsoft.com/en-us/magazine/cc188690.aspx.

14. **Steven Bird, Ewan Klein, Edward Loper.** Extracting Information from Text. [Online] 2009. http://www.nltk.org/book/ch07.html.

15. **King, G. Bauer, C. Andersen, M. R. Bernard,.** H. Hibernate Reference Documentation. [Online] https://docs.jboss.org/hibernate/core/3.6/reference/en-US/pdf/hibernate_reference.pdf.

16. **Ramsdale, Chris.** Large scale application development. [Online] 2010. http://www.gwtproject.org/articles/mvp-architecture.html.