UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

PRACOVNÁ PLOCHA VÝSKUMNÍKA

2011

Katarína Matysová

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

Evidenčné číslo: 184a2bad-9ed2-4f0a-84bc-f6270b9cfc28

PRACOVNÁ PLOCHA VÝSKUMNÍKA

Bakalárska práca

Študijný program: Aplikovaná informatika
Študijný odbor: 9.2.9 Aplikovaná informatika
Školiace pracovisko: Katedra aplikovanej informatiky
Školiteľ: Mgr. Pavel Petrovič, PhD.

Bratislava, 2011

Katarína Matysová

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Katarína Matysová
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

Cieľ: Aby výskumný pracovník mohol vykonávať svoju prácu dostatočne kvalitne, porebuje pravidelne vykonávať rôzne činnosti. Napr. zaznamenávať si svoje nápady a myšlienky, postup, hypotézy, vytvárať si zbierku relevantných článkov a publikácií a písať si k nim svoje poznámky, vyhľadávať v nich efektívne, uchovávať všetky výstupy svojej práce - publikácie, prezentácie, mať ich kedykoľvek k dispozícii, sledovať zaujímavé konferencie, workshopy, stretnutia a odborné časopisy v prehľadnom zozname, atď. Úlohou je analyzovať prácu výskumníka, opísať jeho pravidelné činnosti a implementovať systém, ktorý je prístupný cez webový prehliadač, ktorý môže používať na to, aby udržiaval všetky potrebné informácie. Systém by mal umožňovať pravidelné zálohovanie, jednoduchú údržbu, mal by byť dobre zdokumentovaný a otvorený, aby sa ľahko menil v prípade, že to v budúcnosti bude potrebné.

Vedúci: Mgr. Pavel Petrovič, PhD.

Spôsob sprístupnenia elektronickej verzie práce:
bez obmedzenia

Dátum zadania: 01.10.2010

Dátum schválenia: 25.10.2010

doc. RNDr. Mária Markošová, PhD.
garant študijného programu

.....
študent

Pavel Petrovič
.....
vedúci

Čestne prehlasujem, že som túto bakalársku prácu vypracovala samostatne s použitím citovaných zdrojov.

.....

Katarína Matysová

Abstrakt

Táto práca sa venuje návrhu a vytvoreniu webovej aplikácie, ktorá je určená na uchovávanie štruktúrovaných údajov, ukladá ich do kategórií podľa typu a popri tom poskytuje základné funkcie organizéra. Aplikácia je svojím obsahom prispôsobená potrebám práce výskumného pracovníka. Práca obsahuje popis funkcií a vlastností navrhovaného systému a zaoberá sa jeho porovnaním s podobnými aplikáciami. Analyzuje a porovnáva nástroje, ktoré by mohli byť použité pri vývoji, venuje sa opisu ich vlastností a výberu najvhodnejšieho z nich. Ďalšou časťou je podrobnejšie priblíženie a charakteristika použitých technológií, opis návrhu aplikácie a samotnej implementácie. Jej výsledkom je systém, v ktorom je k dispozícii základná funkcionalita potrebná na jeho používanie, zároveň je jednoducho modifikovateľný, rozšíriteľný a slúži ako nástroj pre efektívne vyhľadávanie potrebných informácií a v konečnom dôsledku zjednodušenie a lepšiu organizáciu práce jej používateľa.

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

Abstract

The goal of this bachelor thesis is to design and implement a web application that stores structured data, divides them into categories according to their type and offers basic organizer functions. The content of the application is suited for needs of the researcher's work. The paper contains a description of functions and features of the system and compares it with similar applications. Thesis analyzes and compares the tools that could be used during the development, it describes their features and chooses the most suitable tool. Next part provides more detailed characteristics of the technology, describes the application design and the implementation process. The result is a system which provides a basic functionality needed for its use, is easily customizable, extensible and serves as a tool for effective search of the necessary information and ultimately simplifies and organizes the work of its users.

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

Obsah

1. Úvod

1.1 Motivácia a základné charakteristiky aplikácie

Každodenná práca výskumníka si vyžaduje synchronizáciu množstva činností, ktoré musí vykonávať, efektívnu organizáciu pracovného času a prehľad o zdrojoch informácií, či už sú to výsledky vlastnej, alebo cudzej práce, knihy, časopisy a podobne. Táto organizácia, aby bola čo najlepšia, by mala byť sústredená na jednom mieste, ktoré je poruke a dostupné vždy, keď je to potrebné.

Pracovná plocha výskumníka je webová aplikácia, ktorá umožní výskumníkovi zhromažďovať informácie rôzneho druhu v štruktúrovanej podobe (napr. publikácie, súbory, projekty, softvér, konferencie, atď.), poskytuje základné funkcie organizéra, ako je pridávanie udalostí do kalendára, vytváranie zoznamu úloh, kontaktov a poznámok. Aplikácia nemá byť len štandardným programom plniacim funkciu organizéra a zoznamu úloh, naopak, je špecifická tým, že umožňuje ukladať záznamy do kategórií s rôznou štruktúrou údajov, vytvárať medzi nimi vzťahy a ukladať ku každému z nich poznámky, štítky (tagy) a súvisiace URL odkazy. Samotné vytváranie vzťahov má používateľovi zjednodušiť a zrýchliť vyhľadávanie tým, že sa k informáciám bude vedieť dostať z viacerých strán. Okrem toho, zoznam kategórií, ktoré bude program obsahovať nie je konečný a pevne daný, je možné ich v prípade potreby rozširovať o ďalšie, resp. odstrániť tie, ktoré nie sú potrebné.

1.2 Existujúce programy

Je veľmi ťažké nájsť existujúcu aplikáciu, ktorá sa funkcionalitou približuje Pracovnej ploche výskumníka. Na internete je obrovské množstvo rôznych organizérov, ktoré poskytujú štandardné funkcie, ako kalendár, ukladanie udalostí, kontaktov, úloh a podobne. Tie sa sčasti podobajú na moju prácu, na druhej strane však obsahujú aj funkcie, ktoré v nej zahrnuté nie sú. Sú to funkcie ako napríklad pripomienky udalostí, alarmy alebo synchronizácia s e-mailovým klientom. Tieto programy a im podobné zároveň nemajú možnosť vkladať iné typy údajov, vytvárať vzťahy ani ukladať súbory.

Programov, ktoré sa primárne zameriavajú na vkladanie záznamov s používateľom definovanou štruktúrou a vytváranie vzťahov medzi nimi, je oveľa menej ako plánovačov a organizérov. Aplikácie z tejto skupiny zasa neobsahujú funkcie kalendára a vkladania

úloh a práca so vzájomnými odkazmi je často ťažkopádna alebo poskytuje iba obmedzenú funkcionalitu.

Vyššie uvedené dôvody sú hlavnou motiváciou k tomu, prečo je potrebné vytvoriť aplikáciu, akou je Pracovná plocha výskumníka.

1.3 Cieľ práce

Cieľom mojej bakalárskej práce teda bude zanalyzovať a porovnať možné prístupy k realizácii takejto aplikácie, porovnať ju s existujúcimi aplikáciami podobného typu a vytvoriť základ a hlavné kategórie údajov potrebné k jej používaniu. Zároveň bude dôležité navrhnúť systém tak, aby ho bolo možné neskôr čo najjednoduchšie rozširovať a upravovať.

1.4 Štruktúra práce

V druhej kapitole budem podrobnejšie opisovať požiadavky na aplikáciu, ktoré zahŕňajú jej funkcionalitu a systémové požiadavky.

Tretia kapitola sa venuje prehľadu technológií, ktoré môžu byť pri implementácii použité, opisuje ich vlastnosti, porovnáva ich a zdôvodňuje vhodnosť, resp. nevhodnosť ich použitia vzhľadom na stanovené požiadavky.

V štvrtej kapitole je opísaný samotný návrh aplikácie, obsahuje návrh databázy, architektúry systému a návrh používateľského rozhrania.

Predposledná, piata kapitola opisuje implementačnú časť, použité nástroje a knižnice a vysvetlenie princípu fungovania niektorých častí aplikácie.

V šiestej kapitole je popísaný krátky návod na inštaláciu a spustenie vytvoreného programu.

2. Požiadavky na systém

2.1 Užívatelia aplikácie

Aplikácia je určená pre ľudí, ktorí potrebujú zhromažďovať a organizovať veľké množstvo informácií rôzneho druhu, potrebujú v nich efektívne vyhľadávať a popri tom mať prehľad o všetkých dôležitých aktivitách, udalostiach a naplánovaných úlohách. Samozrejme, cieľová skupina používateľov závisí aj od kategórií údajov, ktoré program obsahuje. V mojej práci sa budem zameriavať na skupinu používateľov, ktorých práca má blízko k práci výskumníka, čomu zodpovedá aj funkcionalita programu a druhy údajov, s ktorými bude aplikácia narábať.

2.2 Požiadavky na funkčnosť

Aplikácia bude pozostávať z viacerých sekcií (kategórií), z ktorých každá obsahuje záznamy s vopred definovanou štruktúrou. Táto štruktúra je špecifická pre každú sekciu, s výnimkou údajov, ktoré sú spoločné pre každý jeden záznam. Do tejto skupiny patria poznámky, štítky, zoznam odkazov na iné (súvisiace) záznamy v aplikácii a zoznam pomenovaných URL adries.

Poznámky je možné každému záznamu vpísať do jednoduchého textového poľa.

V zozname odkazov sú súvisiace záznamy rozdelené na kategórie, do ktorých v programe patria. Pri každom z nich sa zobrazuje jeho názov a tlačidlo odkazujúce na okno s podrobnými informáciami.

Zoznam URL adries tvoria dvojice *Názov – Adresa*, po kliknutí na pole s adresou sa v prehliadači otvorí daná webová stránka.

2.2.1 Štítky

Zoznam štítkov je použitý pre všetky kategórie záznamov v aplikácii a vytvára si ho sám používateľ. Pri označovaní záznamu vyberie z vytvoreného zoznamu štítkov len tú podmnožinu, ktorá charakterizuje vkladaný obsah.

Štítky tvoria stromovú štruktúru. Je vhodná pri ich zadelení do skupín a podskupín najmä tým, že umožní lepšie a jednoznačnejšie vyjadriť význam jednotlivých štítkov a zároveň tým zjednodušiť prehľadávanie záznamov. Preto je použitie stromu v tomto prípade výhodnejšie ako jednoduchá lineárna štruktúra.

2.2.2 Zoznam kategórií údajov

Používateľ môže do programu zadávať údaje rozdelené do nasledujúcich kategórií:

- Publikácie
- Časopisy
- Vlastné výstupy
- Konferencie
- Posudky
- Akcie, podujatia
- Pracovné cesty
- Softvér
- Projekty
- Evidencia majetku
- Foto a video dokumentácia
- Zoznam emailov
- Kontakty
- Kalendár
- Zoznam úloh
- Zápisník

2.2.3 Publikácie

Táto kategória je určená na vkladanie záznamov, z ktorých každý popisuje jednu konkrétnu publikáciu. Všetky záznamy sú exportovateľné vo formáte BibTeX, ktorý je určený na ukladanie informácií o citáciách a zdrojoch. Údaje ukladané ku každému záznamu sú vybrané zo štandardnej množiny údajov definovaných v BibTeX formáte, patrí medzi ne typ záznamu (článok, kniha, manuál, posudok, atď.), autor, názov, vydavateľ, rok vydania, editor, číslo vydania, kapitola/diel, séria vydania, adresa vydavateľa, edícia, mesiac vydania, kľúčové slová, vydanie v elektronickej podobe, URL adresa zdroja.

Okrem týchto údajov má používateľ možnosť ku každému záznamu nahrať súbor obsahujúci danú publikáciu v elektronickej podobe (napr. vo formáte .pdf), označiť publikáciu ako vlastnú alebo cudziu a označiť, či má, alebo nemá k dispozícii fyzický výtlačok. V prípade potreby vie takisto označiť knihu ako požičanú a uložiť k záznamu dátum, dokedy je potrebné knihu vrátiť. Ak knihu požičal niekomu inému, môže pridať meno a kontakt na tohto človeka, poprípade vloží iba odkaz na údaj zo zoznamu kontaktov, ak sa tam daná osoba nachádza.

2.2.4 Časopisy

V tejto časti si môže používateľ vkladať záznamy o časopisoch, denníkoch alebo iných pravidelne vydávaných zdrojoch. Vie si k nim uložiť informáciu o názve časopisu, vydavateľovi, čísle vydania, sídle vydavateľa, roku vydania prvého čísla a názve spoločnosti, ktorá časopis vydáva.

2.2.5 Vlastné výstupy

Táto kategória je určená na ukladanie súborov a výstupov, ktoré výskumník počas svojej práce vytvára. Výstupy sú rozdelené do štyroch skupín, sú to:

- Dokumenty
- Prezentácie
- Dátové súbory
- Programy

Každý záznam z tejto časti obsahuje názov, dátum vytvorenia, súbor alebo URL súboru, označenie verzie a dátum, dokedy majú byť dokončené (ak je to potrebné). Výstupy sa teda buď nahrávajú na server, alebo sa uloží iba odkaz na miesto, kde sa nachádzajú. Všetky súbory takto nahraté na server sú používateľovi dostupné nie len cez rozhranie aplikácie, ale môže k nim pristupovať aj cez zabezpečené FTP. To platí aj pre iné kategórie, ktoré ku svojim údajom ukladajú súbory.

2.2.6 Konferencie

Návštevy rôznych konferencií sú neoddeliteľnou súčasťou práce výskumníka, preto je potrebné mať informácie o nich a o všetkom, čo s nimi súvisí, dobre zorganizované. Táto sekcia programu obsahuje údaje o plánovaných alebo v minulosti absolvovaných konferenciách, spolu s údajmi ako sú skratka, plný názov, miesto konania konferencie, dátum konania.

Okrem nich môže každý záznam ukladať niekoľko súvisiacich termínov, tie (ich názvy, kvôli opakovanému použitiu) si zadefinuje používateľ vopred a pri vytváraní nového záznamu o konferencii už iba vyberá z tohto vytvoreného zoznamu tie, ktoré sú potrebné a k nim vloží príslušný dátum. Na začiatku obsahuje aplikácia zoznam s preddefinovanými typmi termínov, patrí tam napr. odoslanie abstraktu, potvrdenie abstraktu, odoslanie článku na posúdenie, odpovede na článok s pripomienkami, odoslanie výslednej verzie a odoslanie posterov. Tento zoznam je samozrejme možné ľubovoľne upravovať.

Ďalšími ukladanými údajmi sú súbor (alebo odkaz na súbor) súvisiaci s udalosťou a text pozvánky pre autorov – ten sa nahrá buď vo forme súboru, priamo vloženého textu alebo ako odkaz na konkrétny email zo zoznamu v časti Emailové konto

2.2.7 Posudky

Táto sekcia obsahuje informácie o článkoch, na ktoré má výskumník napísať posudok. Záznamy obsahujú názov udalosti, pre ktorú je posudok určený, pozvánku na spravenie posudku vo forme buď súboru, alebo ako odkaz na email v časti Emailové konto, kópiu článku/práce, ktorú treba posudzovať, názov článku, meno autora a dátum odovzdania. Je možné pripojiť aj zoznam súvisiacich dokumentov a prístupové údaje k systému, kde sa má hotový posudok nahrať.

2.2.8 Akcie, podujatia

Kategória uchovávajúca informácie o akciách a podujatiach je jedna z menej rozsiahlych, ale o to dôležitejších častí programu, ktorá výrazne pomáha organizovať a plánovať pracovný čas. Pri každom podujatí si používateľ ukladá jeho názov, textový popis, typ a termín (dátum a čas) začiatku a konca podujatia.

2.2.9 Pracovné cesty

Každý údaj v tejto kategórii obsahuje miesto konania pracovnej cesty, dátum začiatku a dátum ukončenia. Používateľ môže nahrať do systému dokument s príkazom na pracovnú cestu a dokument obsahujúci správu z cesty. Je tu priestor pre vloženie poznámok, ktoré sa týkajú napr. priebehu cesty alebo spôsobu cestovania. Taktiež sa tu dajú vložiť súbory s elektronickými lístkami potrebnými počas cesty.

2.2.10 Softvér

Táto sekcia je určená na ukladanie informácií o softvéri, ktorý výskumník používa, alebo si plánuje v budúcnosti zaobstarať. Záznamy obsahujú názov produktu, označenie verzie, meno resp. názov dodávateľskej firmy, cenu a URL adresu zdroja, kde je daný softvér k dispozícii na siahnutie.

2.2.11 Projekty

Projekty umožňujú okrem zadefinovania vlastných údajov mnohonásobné odkazy na ostatné časti systému, čím zhromažďujú záznamy ľubovoľného typu (kategórie) do jedného súvisiaceho celku = Projekt. Každý projekt obsahuje názov, popis alebo špecifikáciu zadania, dobu trvania (dátum začatia a ukončenia), kategóriu – jedna z možností minulé/aktívne/plánované. Ďalej sú to odkazy na ostatné záznamy v systéme

(publikácie, časopisy, výstupy, skupiny v zozname úloh, ...). Ich pridávanie funguje podobne ako pridávanie odkazov pri hociktorom inom zázname.

Súčasťou je zoznam termínov súvisiacich s projektom. Jeho funkcionalita je podobná ako pri zozname termínov konferencií.

Zoznam riešiteľov projektu je vytváraný tak, že sa buď vyberie osoba z časti Kontakty, alebo ak sa tam nenachádza, vloží sa iba jej meno. Ku každej osobe z tohto zoznamu je potom možnosť vkladať komentáre. Tie sú špecifické pre každý projekt aj v prípade, že riešitelia sú vo viacerých projektoch tí istí.

Projekt ďalej obsahuje údaje o zdroji jeho financovania a o rozpočte. Tam patrí celkový rozpočet pre projekt, zostávajúci rozpočet a zoznam jednotlivých transakcií a výdavkov.

2.2.12 Evidencia majetku

Sekcia Evidencia majetku je určená na zhromažďovanie informácií o rôznych pomôckach a zariadeniach používaných pri práci, vrátane tých, ktoré má výskumník v pláne zaobstarať si v budúcnosti. Záznamy obsahujú názov, miesto umiestnenia, cenu, zdroj financovania – buď v textovej forme, alebo ako komentovaný odkaz na projekt v systéme, dátum nadobudnutia a dokumenty s objednávkou a faktúrou.

2.2.13 Foto a video dokumentácia

Umožňuje nahrávanie súborov do používateľom definovanej štruktúry priečinkov. Je možné súbory nahrávať jednotlivo cez formulár priamo v systéme, alebo hromadne. napr. cez FTP klienta. Namiesto súboru sa dá uložiť aj odkaz na miesto, kde je daný súbor uložený.

2.2.14 Zoznam emailov

Používateľ si môže do systému vložiť prístupové údaje k emailovému kontu, z ktorého sa pravidelne vyberajú všetky emaily a v textovej podobe sa ukladajú do systému. V prípade, že email obsahuje prílohy, tie sa uložia do používateľovho priečinka a odkazy na tieto prílohy sa zapamätajú spolu s textom emailu. S týmito emailami sa potom dajú vykonávať operácie ako s ktorýmkoľvek iným záznamom, t.j. zoraďovanie, vyhľadávanie, pridávanie poznámok, priradenie odkazov na iné záznamy a pod.

2.2.15 Kontakty

Zoznam kontaktov na osoby, s ktorými výskumník spolupracuje je nevyhnutnou súčasťou tejto aplikácie. Vzhľadom na to, že je ťažké vopred definovať, aké typy kontaktných údajov bude potrebné pri každej z osôb ukladať, používateľ si ich môže v programe zadefinovať sám. To znamená, že globálne pre celú aplikáciu bude vedieť vytvoriť zoznam všetkých druhov kontaktných informácii, ktoré sa budú dať k osobám uložiť (napr. adresa, mobilný telefón, telefón domov, telefón do práce, email, ...). Následne pri vkladaní kontaktu na osobu vyberie zo vytvoreného zoznamu tú podmnožinu údajov, ktorú potrebuje (napr. iba adresu a email) a tie položky vyplní. Na začiatku sa už v zozname nachádzajú niektoré štandardné typy kontaktných údajov.

2.2.16 Kalendár

Umožňuje prehľadné zobrazenie udalostí zadaných v systéme (napr. termíny konania konferencií, akcií, odovzdávania posudkov, ...), udalosti v kalendári patria do kategórií vytvorených podľa jednotlivých sekcií programu (publikácie, výstupy, konferencie, ...). Na zobrazovanie udalostí je možné nastaviť filter, teda určiť, ktoré kategórie sa majú/nemajú zobraziť. Po kliknutí na udalosť je možnosť ju prezrieť, editovať alebo zmazať. Ak je naviazaná na nejaký konkrétny záznam (termín), môže sa úplne zmazať spolu s príslušným záznamom, alebo sa len prestane zobrazovať v kalendári. Zobraziť sa dá prehľad udalostí v aktuálnom mesiaci, týždni alebo dni.

Okrem hlavného kalendára sa bude zobrazovať jeho zmenšená a zjednodušená verzia na úvodnej stránke programu. V tejto verzii bude každý deň, v ktorom je vytvorená nejaká udalosť, farebne označený (v závislosti od kategórie udalosti), pričom krátky popis udalostí sa zobrazí napr. kliknutím na daný deň. Tento zjednodušený kalendár neposkytuje funkcionalitu hlavného kalendára, slúži iba na to, aby mal používateľ základné informácie z kalendára stále poruke.

2.2.17 Zoznam úloh

Obsahuje skupiny (zoznamy) krátkych poznámok, z ktorých každá popisuje nejakú činnosť, ktorú je potrebné vykonať v zadefinovanom dátumovom intervale. Poznámky teda ukladajú text, dátum začiatku a konca činnosti (v prípade jednodňovej udalosti sú dátumy rovnaké) a prioritu v rámci skupiny.

Používateľ si môže zadefinovať ľubovoľné množstvo skupín a v každej skupine ľubovoľne veľa poznámok. Každá zo skupín môže byť priradená ako odkaz niektorému

záznamu v aplikácii a zároveň skupina môže obsahovať odkazy na iné záznamy. Poznámky v skupine sa dajú zoraďovať buď ručne, alebo podľa ich priority. Ku každej poznámke sa automaticky v kalendári vytvorí príslušná udalosť. Je zobrazená ako postupnosť dní, v ktorých sa má činnosť vykonávať, s jasne zvýrazneným začiatočným a koncovým dátumom.

2.2.18 Zápisník

Štruktúra zápisníka je podobná ako štruktúra jednoduchého blogu – tvorí ho zoznam článkov/poznámok, ktoré v sebe obsahujú predmet (názov), text poznámky, dátum a kľúčové slová. Poznámky sa dajú zoraďovať podľa dátumu alebo názvu. Funguje filtrovanie podľa kľúčových slov, dátumu (zobrazenie iba poznámok od – do nejakého dátumu) a klasické vyhľadávanie.

2.3 Ostatné charakteristiky aplikácie

2.3.1 Prístup cez webový prehliadač, efektivita a rozšíriteľnosť

Aplikácia Pracovná plocha výskumníka je určená na to, aby zjednodušila, zorganizovala a urýchlila prácu používateľovi. Preto je dôležité, aby bola kdekoľvek k dispozícii. To sa dá dosiahnuť tým, že bude vytvorená ako webová aplikácia a nie ako desktopová, aby sa predišlo potrebe inštalovať ju na každý počítač, na ktorom potrebuje používateľ pracovať. Zároveň sa kladie dôraz na efektivitu často vykonávaných operácii, čo je okrem iného možné zabezpečiť aj tým, že sa minimalizuje komunikácia klientskej časti aplikácie so serverom, aby používateľ mohol plnohodnotne pracovať aj pri pomalších pripojeniach na internet.

Ďalšou požiadavkou je čo jednoduchá rozšíriteľnosť aplikácie. Mala by byť teda navrhnutá tak, aby bolo možné pridávať nové (resp. odstrániť nepotrebné) kategórie záznamov a druhy údajov ukladaných pri záznamoch v jednotlivých kategóriách. Nie je vyžadované to, aby tieto zmeny vedel priamo za behu urobiť používateľ aplikácie, ale mali by sa dať vykonať s čo najmenšími zásahmi do samotného programu.

2.3.2 Vytváranie vzťahov medzi záznamami

Medzi dvoma ľubovoľnými záznamami v systéme je možné vytvárať vzťahy tak, že pri vkladaní (editácii) sa označia súvisiace záznamy vybrané z ľubovoľnej kategórie v

programe. Znamená to, že pri prezeraní záznamu sa okrem informácií o ňom zobrazia aj všetky tie, ktoré sú s ním spojené.

2.3.3 Vyhľadávanie

Funguje globálne pre všetky časti systému, alebo je prístupné pre každú sekciu programu pri prezeraní jej záznamov.

V jednotlivých sekciách sa dá vyhľadávať pomocou filtrov aplikovaných na súhrnnú tabuľku so záznamami (v nej je zobrazených iba niekoľko hlavných polí záznamu). Filtre sú buď reťazcové (t.j. vyhľadajú sa všetky záznamy, ktorých zadané pole obsahuje nejakú postupnosť znakov), číselné, alebo dátumové (rozsah od – do zadaných čísel alebo dátumov).

Vyhľadávanie v celej aplikácii prebieha dvoma spôsobmi. Buď fulltextovo, alebo iba v spoločných poliach záznamov (štítky a poznámky). Pre hľadané slovo sa potom vyberie aj zoznam kategórií, v ktorých vyhľadávanie prebieha.

2.3.4 Automatické vytváranie udalostí v kalendári

Pri definovaní termínov a dátumov je používateľovi ponúknutá možnosť automaticky tento dátum vložiť do kalendára ako novú udalosť. Ak si túto možnosť zvolí, zadá kategóriu, názov a popis udalosti (ak je to možné, údaje sa vyberú z vkladaného záznamu) a tá sa vytvorí.

2.3.5 Automatická záloha a obnovenie systému

Záloha programu sa vytvára pravidelne v zadefinovanom časovom intervale, alebo sa vytvorí kedykoľvek, keď o to používateľ požiada. Záloha obsahuje kompletné informácie o systéme – údaje z databázy a všetky súbory, ktoré sú archivované v jednom *.zip* súbore. Vždy po vytvorení zálohy sa tento *.zip* súbor nahrá na používateľom definované miesto (externý server). Systém je možné z uloženej zálohy kedykoľvek obnoviť.

2.3.6 Vytváranie logu

Log je zoznam s akciami, ktoré používateľ na systéme vykonal aj s dátumom, kedy boli vykonané. Do logu sa zároveň ukladajú chybové správy a správy o činnosti systému, ktoré prebiehajú automaticky a nevykonal ich priamo používateľ (napr. vytvorenie zálohy). Log je kedykoľvek k dispozícii na prezretie.

2.3.7 Ukladanie súborov

Všetky súbory, ktoré sú do systému nahraté a ktoré sú v ňom vytvárané, sa ukladajú do adresárovej štruktúry, v ktorej sú súbory rozdelené do podadresárov podľa toho, do akej sekcie v systéme patria (t.j. adresárový strom obsahuje podadresáre *publikacie*, *vlastne_vytupy*, *konferencie*, ...). K súborom sa dá pristupovať nie len prostredníctvom programu, ale aj priamym prístupom cez zabezpečené FTP.

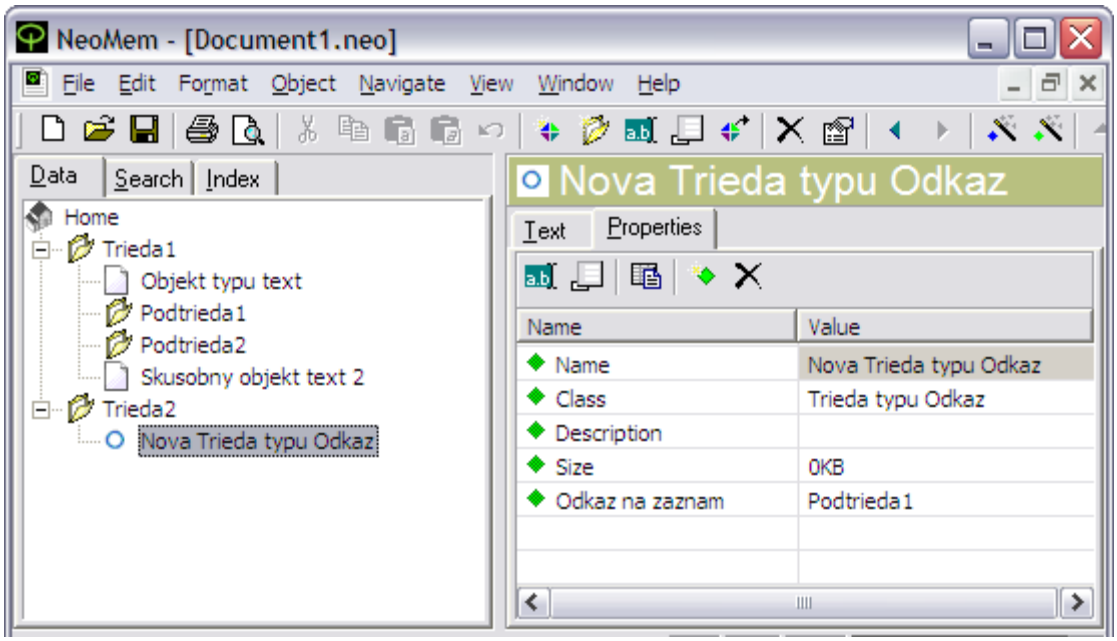
3. Analýza a prehľad technológií

3.1 Podobné programy

3.1.1 NeoMem

NeoMem je open-source desktopová aplikácia, dostupná pre OS Windows. Umožňuje vytvárať záznamy rôznych typov (tried) a organizovať ich v používateľom vytvorenej štruktúre adresárov. Triedy záznamov si definuje sám používateľ. Program obsahuje dve predvolené triedy – Adresár a Článok. Adresár je typ záznamu, ktorý v sebe obsahuje iné záznamy.

Každý záznam obsahuje textový popis, a podľa triedy, do ktorej patrí, rôzne druhy informácií (polí). Polia sa dajú zadefinovať pre každú triedu samostatne. Pri vytváraní polí je možné si vybrať zo základných dátových typov (reťazec, číslo, logická hodnota, dátum), alebo iných, napr. súbor, URL, odkaz na iný záznam v aplikácii a pod. V záznamoch sa dá vyhľadávať fulltextovo, buď vo všetkých, alebo iba v niektorých konkrétnych poliach.



Obrázok 3.1. Ukážka programu NeoMem

Medzi ďalšie funkcie patrí export všetkých záznamov vo formáte .csv, import z formátov .rtf alebo .txt a možnosť uložiť súbor chránený heslom a v šifrovanej podobe.

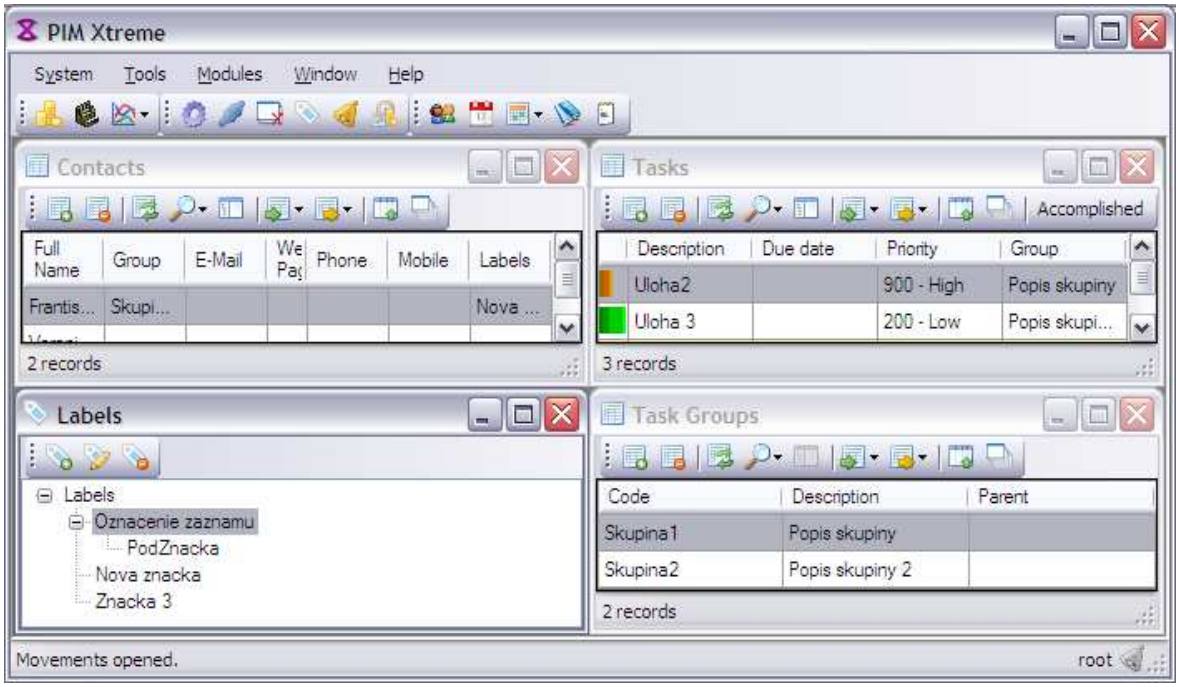
Program neobsahuje žiaden kalendár alebo organizér, je veľmi nepraktický v prípade, že chce používateľ vložiť k záznamu viacero odkazov alebo súborov, pretože je potrebné

mať pre každý odkaz/súbor zadefinované samostatné pole. Taktiež chýba označovanie záznamov napr. kľúčovými slovami alebo štítkami.

3.1.2 PIM Xtreme

Desktopová aplikácia PIM Xtreme patrí do početnej skupiny organizérov, ktoré uchovávajú úlohy, udalosti, kontakty a pod. Oproti väčšine iných programov tohto typu obsahuje navyše napríklad možnosť vkladať poznámky ku záznamom a odkazovať na iné kontakty alebo udalosti.

V programe je viacero modulov, sú to úlohy, kontakty, kalendár, budík (pripomienkovač), udalosti, a poznámky. Moduly zoskupujú všetky záznamy do skupín, dajú sa teda vytvoriť skupiny kontaktov, skupiny úloh, atď. Okrem týchto modulov sa tam nachádza ešte modul Štítky, kde sa vytvára stromová štruktúra označení použiteľných pre záznamy v celej aplikácii.



Obrázok 3.2. Ukážka programu Pim XTreme

K záznamom sa dajú vkladať odkazy na iné záznamy a poznámky, odkazy ale nie sú vo všetkých moduloch, iba pri niektorých (udalosti, kontakty a úlohy), okrem toho je možné uložiť iba 1 odkaz. Ďalším, podobným nedostatkom je to, že označovanie štítkami nefunguje v žiadnom module okrem kontaktov. Programu chýba aj globálne vyhľadávanie, obsahuje iba filter na údaje pre každý modul samostatne.

Každý modul sa otvára v samostatnom okne, čo umožňuje používateľovi uložiť si konfiguráciu pracovnej plochy a zároveň vidieť naraz všetky potrebné informácie. Všetky

moduly vedia svoje záznamy importovať/exportovať vo formáte XML alebo v textovom súbore. Celú databázu je možné zálohovať, alebo obnoviť dáta zo zálohy.

3.1.3 ContactOffice

ContactOffice je webová aplikácia, ktorá v sebe spája organizér, spravovanie súborov, vytváranie dokumentov prostredníctvom Wiki a množstvo ďalších funkcií, spolu s možnosťou vytvárať si skupiny kontaktov a nastavovať zdieľanie svojich údajov vzhľadom na tieto skupiny.

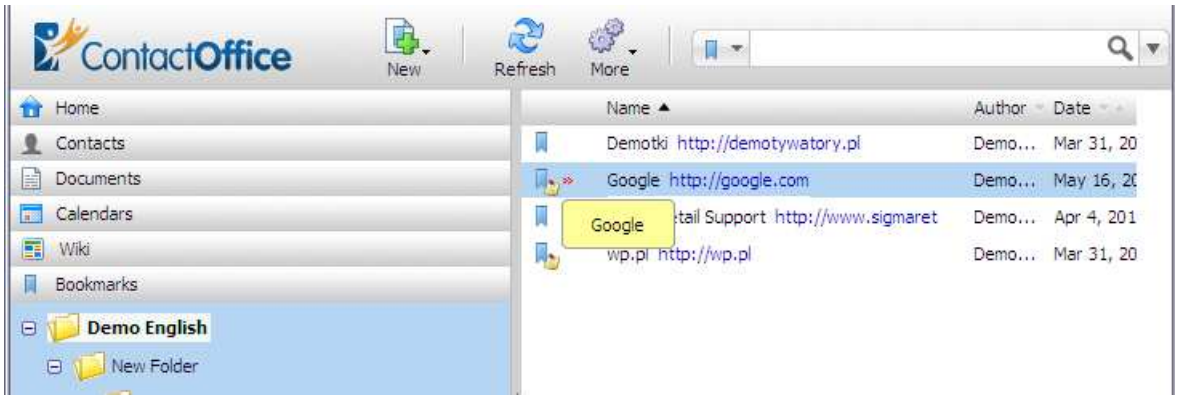
V časti Kontakty obsahuje každý záznam okrem základných kontaktných informácií aj zoznam príloh – súborov, ktoré boli do programu nahraté v kategórii Dokumenty.

V kalendári sú prehľadne zobrazené všetky vložené udalosti a úlohy, umožňuje prepínanie náhľadu medzi aktuálnym dňom, týždňom alebo mesiacom. Pri udalostiach je okrem štandardných údajov možné nastaviť aj SMS alebo e-mailové pripomienky, zoznam súvisiacich kontaktov a dokumentov.

Záložky obsahujú zoznam URL odkazov organizovaných v stromovej štruktúre adresárov. Kategória Úlohy je rozdelená na 2 podskupiny – úlohy priradené používateľovi a úlohy od používateľa pre ostatných.

Aplikácia obsahuje aj časť Emailové konto, z ktorej môže používateľ pristupovať ku svojim emailovým účtom.

Ku všetkým údajom sa dá pripojiť textový popis a zoznam štítkov, štítky potom slúžia pri vyhľadávaní. Údaje v jednotlivých kategóriách sú rozdelené na adresáre a podadresáre. Dáta sú exportovateľné vo viacerých formátoch, podľa typu dát a kategórie (vCard, csv, html súbor a pod.), export však nie je napríklad pri poznámkach alebo úlohách. V aplikácii chýba možnosť vytvárania vzájomných odkazov.



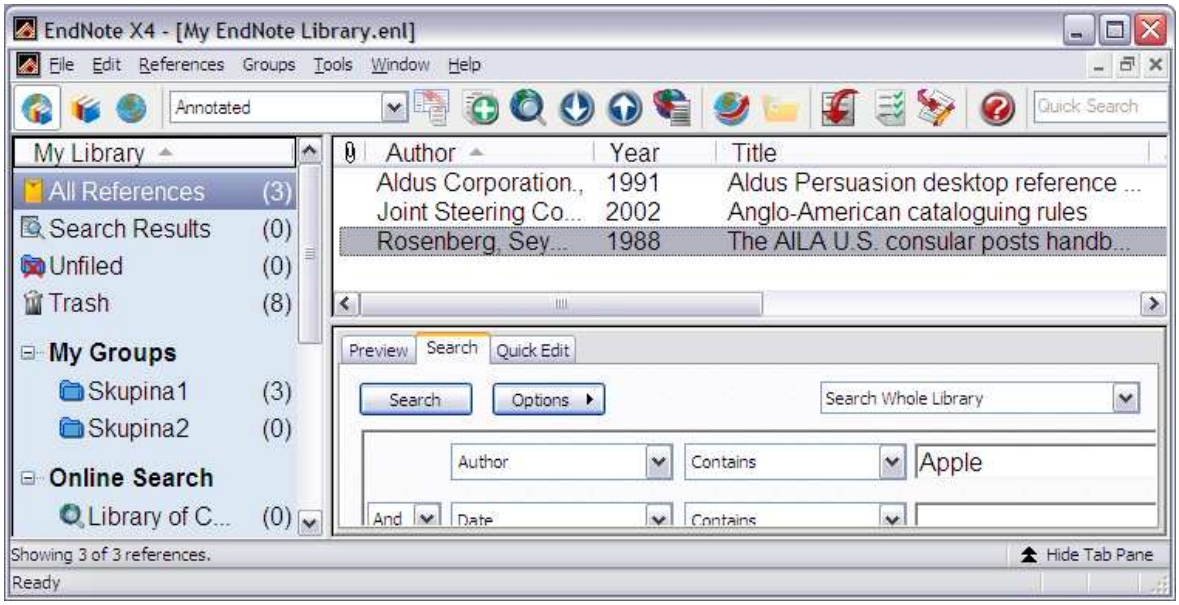
Obrázok 3.3. Ukážka programu ContactOffice

ContactOffice je dostupná zadarmo iba v obmedzenej verzii, kde nie sú k dispozícii niektoré funkcie, alebo je obmedzené množstvo uložených údajov. Chýba napríklad vyhľadávanie, export do niektorých formátov alebo možnosť pracovať s IMAP emailovými kontami.

3.1.4 EndNote

EndNote je desktopová aplikácia, ktorá je určená na organizovanie záznamov o publikáciách a článkoch, umožňuje vyhľadávanie v internetových databázach publikácií a vkladanie vyhľadaných záznamov do vlastných skupín. Aplikácia podporuje export a import všetkých záznamov, automatické vkladanie bibliografických údajov do textových editorov ako sú Microsoft Word a OpenOffice.org Writer a pod.

Množstvo funkcií tohto programu je značne odlišných od Pracovnej plochy výskumníka, preto nemá zmysel tieto dve aplikácie porovnávať vzhľadom na to, aké funkcie organizéra obsahuje EndNote alebo do akej miery Pracovná plocha výskumníka spolupracuje s textovým editorom. Vhodnejšie je porovnať možnosti súvisiace s vytváraním vzťahov, vlastnej štruktúry záznamov alebo vkladaním súborov.



Obrázok 3.4. Ukážka programu EndNote

Všetky bibliografické záznamy v aplikácii EndNote sa dajú rozdeliť do používateľom definovaných skupín. Každý záznam patrí pod jednu konkrétnu kategóriu (napr. knihy, slovníky, elektronické knihy, encyklopédie, ...), podľa toho je vybraná množina polí, ktoré záznam obsahuje. Túto množinu si môže používateľ pre každú kategóriu ľubovoľne zmeniť.

Záznamy obsahujú aj polia pre vkladanie URL odkazov, obrázkov a súvisiacich súborov. URL odkazy sa vkladajú iba v textovej forme, ale je možné vložiť iba jednu adresu, bez popisu. Počet súborov pri jednom zázname je obmedzený na 45 súborov. Chýba vytváranie vzájomných vzťahov medzi záznamami.

V programe sa dá vyhľadávať podľa vybraných polí (najviac tri naraz), buď v internetových databázach, alebo v lokálnej.

3.1.5 Zhrnutie

Všetky spomínané aplikácie majú časť funkcií spoločnú s Pracovnou plochou výskumníka, zároveň sa ale orientujú trochu iným smerom a niektoré dôležité funkcie sa v nich nenachádzajú. Tabuľka obsahuje stručný prehľad výhod a nevýhod jednotlivých programov.

	výhody	***nevýhody***
NeoMem	vzájomné odkazy, poznámky pri záznamoch, používateľom definovaná štruktúra údajov, záznamy v adresárovej štruktúre, export	chýba kalendár a zoznamy úloh, problém pri vkladaní viacerých odkazov/súborov k záznamu, chýbajú globálne štítky
PIM Xtreme	vzájomné odkazy, globálne štítky, poznámky pri záznamoch, ľubovoľné rozloženie modulov na pracovnej ploche, export	štítky a odkazy nie sú pri všetkých moduloch, vloženie iba 1 odkazu, chýba vkladanie súborov a globálne vyhľadávanie
ContactOffice	prepracovaný organizér, ukladanie súborov, poznámky pri záznamoch, globálne štítky, záznamy v adresárovej štruktúre, emailové konto, export	obmedzená bezplatná verzia, chýbajú vzájomné odkazy a ukladanie súborov pri všetkých kategóriách
EndNote	práca s bibliografickými údajmi, používateľom definovaná štruktúra údajov, ukladanie súborov k záznamom, záznamy v adresárovej štruktúre	chýbajú vzájomné odkazy a poznámky pri záznamoch, viac ako jeden URL odkaz pri zázname, globálne štítky

3.2 Prehľad frameworkov

Pri tvorbe rozsiahlejšej webovej aplikácie, ktorá obsahuje netriviálnu funkcionalitu a má byť navrhnutá tak, aby ju bolo možné ďalej rozširovať, je vhodné použiť niektorý z existujúcich webových frameworkov. Tie ušetria programátorovi prácu na tom, čo by inak musel vytvoriť sám od základu – štruktúru zdrojových kódov, návrh architektúry, zabezpečenie a používateľské roly, rozhranie na prácu s databázou, spracovanie chybových stavov a množstvo ďalších. Ďalšou výhodou je to, že poskytujú možnosť písať kód oveľa rýchlejšie a efektívnejšie vďaka jeho opakovanej použiteľnosti a je k nim dostupných množstvo knižníc s hotovými komponentmi. Navyše, pri použití kvalitného frameworku má programátor istotu, že aplikácia je postavená na stabilnom základe a spĺňa požiadavky pre efektivitu a bezpečnosť.

3.2.1 Základné požiadavky na použitý framework

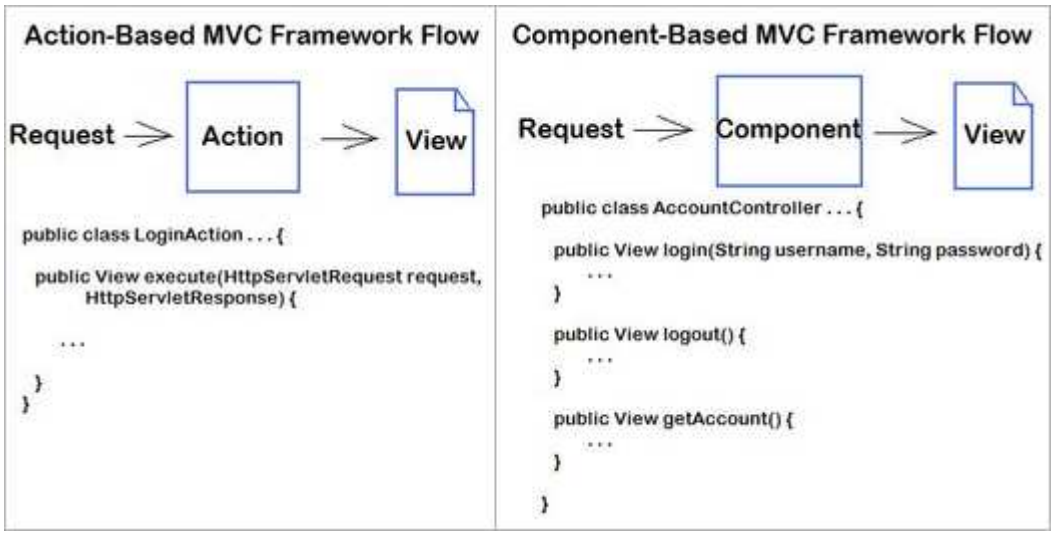
Nasledujúce vlastnosti, ktoré by mal spĺňať framework použitý pri vývoji aplikácie Pracovná plocha výskumníka, vyplývajú zo zoznamu funkčných požiadaviek popísaných v predchádzajúcej kapitole.

- *Model-View-Presenter (MVP) architektúra*: MVP (alebo jej podobná Model-View-Controller) je často používaná architektúra v interaktívnych webových aplikáciách, jej výhodou je oddelenie obsahu stránky (View), reprezentácie dát (Model) a časti, ktorá vykonáva operácie nad dátami na základe akcií používateľa (Presenter).
- *Podpora pre objektovo-relačné (OR) mapovanie objektov na databázu*: nástroje pre OR mapovanie uľahčujú písanie dotazov najmä tým, že prepájajú dve rozdielne reprezentácie dát – objektovú reprezentáciu a relačnú databázu. Ďalšou výhodou je to, že syntax dotazov je jednotná bez ohľadu na typ použitej databázy.
- *Podpora technológie Ajax*: Ajax umožňuje aktualizovať obsah webových stránok bez potreby ich znova načítať, čím sa zvýši interaktivita aplikácie a minimalizuje sa komunikácia klienta so serverom.
- *Čo najviac dokumentácie a dostupných tutoriálov*
- *Knižnice hotových komponentov:* medzi potrebné komponenty patria napríklad formuláre, kalendár, panely pre zobrazenie stromovej štruktúry alebo tabuľky.

3.2.2 Struts2

Struts2 je webový framework pre vývoj aplikácii v jazyku Java. Jeho priamy predchodca, Struts1, vznikol v roku 2000 a bol to prvý framework, ktorý používal

Model-View-Controller architektúru. Jeho dôležitou vlastnosťou je modulárnosť – je ľahko rozšíriteľný a existuje preňho množstvo zásuvných modulov vo forme JAR archívov. Patrí do skupiny tzv. „action based“ frameworkov, čo znamená, že obsah stránky je generovaný na základe URL adresy a HTTP odpovedí od servera - jeden typ používateľovej akcie spracúva v aplikácii jeden konkrétny objekt a ten je zodpovedný za generovanie výstupu. Sú teda vhodnejšie pre jednoduché typy aplikácií, kde nie je vykonávaných príliš veľa akcií na strane používateľa[8]. Pre porovnanie, druhá skupina sú tzv. „component based“ frameworky, ktoré zhromažďujú príbuzné akcie do jedného celku, každému takémuto celku je priradený jeden objekt (pri použití MVC architektúry je to časť „controller“) a ten sa stará o generovanie výstupu a spracovanie akcií. „Component based“ frameworky sú potom použiteľné v aplikácii, kde používateľ vykonáva množstvo akcií na jednej webovej stránke a URL adresa a odpoveď od servera nie sú jediné smerodajné údaje pre vytvorenie obsahu – dôležitý je aktuálny stav pozostávajúci z množstva parametrov ukladaných na serveri.



Obrázok 3.5. Schéma fungovania *„action based“* a *„component based“* frameworkov[8]

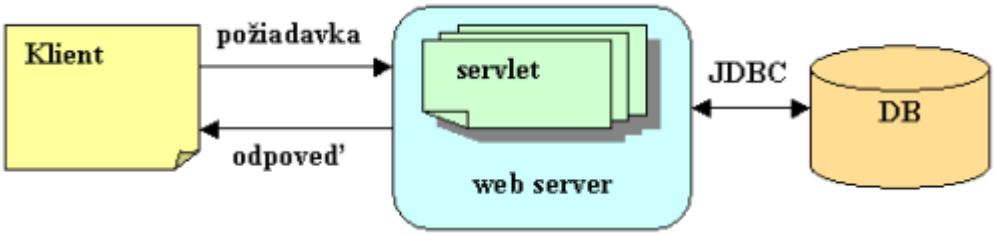
Struts2 je možné doplniť o Hibernate Plugin - zásuvný modul pre integráciu s Hibernate frameworkom, čo je nástroj pre objektovo-relačné mapovanie v jazyku Java. Medzi dostupnými zásuvnými modulmi sa nachádza aj Dojo Toolkit library, ktorý umožňuje pracovať s Ajax-om alebo knižnica komponentov založená na Yahoo! User Interface Library (YUI).

3.2.3 Java Server Faces

Java Server Faces (JSF) je framework, alebo lepšie povedané špecifikácia popisujúca framework pre vývoj webových aplikácií, ktorá má dve nezávislé implementácie (Mojarra

a MyFaces). JSF je postavený na architektúre Model-View-Controller. Patrí do skupiny „component based“ frameworkov, k dispozícii je množstvo znovu použiteľných komponentov, či už patriacich do základnej implementácie, alebo pridaných prostredníctvom niektorej z knižníc. Ajax je v JSF natívne podporovaný, ale zároveň je možné si doinštalovať niektorú z knižníc, ktorá obsahuje komponenty pre prácu s ním. JSF je súčasťou JavaEE edície, je zahrnutý v jej špecifikácii od verzie J2EE 5.0.

Používateľské rozhranie je prezentované prostredníctvom JSP stránok, z ktorých každá pozostáva z formulárov a komponentov. Pri každej vykonanej akcii (zmena hodnoty poľa, kliknutie na tlačidlo, a pod.) sa zavolá metóda, ktorá je naviazaná na konkrétnu akciu a tá je spracovaná špeciálnym servletom – FacesServlet. Servlet je programový komponent napísaný v jazyku Java, ktorý spolupracuje (nie nutne, ale vo väčšine prípadov) s http protokolom, na serveri spracúva požiadavky odoslané klientom a následne generuje webovú stránku[10]. FacesServlet je teda typ servletu určený pre použitie v JSF aplikáciách.



Obrázok 3.6. Schéma fungovania servletov[10]

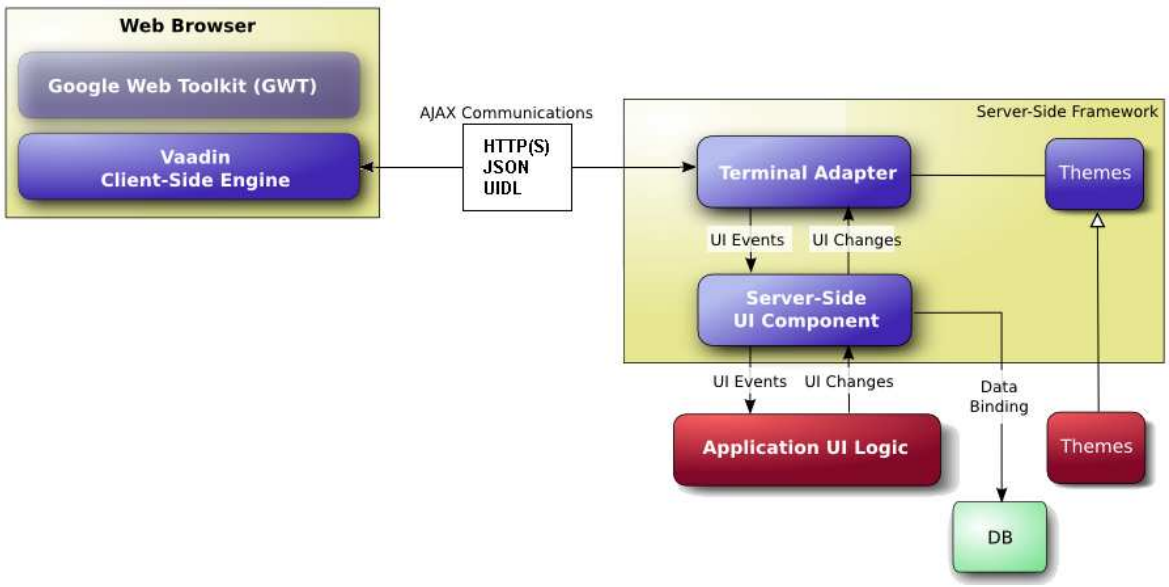
Rovnako ako Struts2, aj JSF je integrovateľný s Hibernate frameworkom pre objektovo-relačné mapovanie.

3.2.4 Vaadin

Vaadin, podobne ako ostatné frameworky popisované v tejto kapitole, je postavený na jazyku Java. Na strane klienta je použitá technológia Ajax a väčšina kódu aplikácie je potom vykonávaná na serveri. Na generovanie stránky a používateľského rozhrania používa Google Web Toolkit (GWT) framework, čím je zabezpečená kompatibilita s prehliadačmi pri použití JavaScript-u. Pre Vaadin je k dispozícii množstvo knižníc komponentov a je možné ho dopĺňať aj o komponenty pôvodne určené pre GWT.

Na serveri sa nachádza modul nazývaný *terminálový adaptér*, ktorý reaguje na akcie používateľa, na základe nich zistí, aká udalosť nastala a ktorému komponentu patrí. Udalosť konkrétneho komponentu je následne vykonaná a klientovi sa pošle odpoveď vo

forme zmien, ktoré nastali na webovej stránke[11]. Za spracovanie a odoslanie tejto odpovede je znova zodpovedný terminálový adaptér.



Obrázok 3.7. Schéma architektúry Vaadin frameworku[11]

Pre oddelenie vzhľadu používateľského rozhrania od samotného kódu a aplikačnej logiky sa používajú šablóny. Šablóny obsahujú súbory s CSS štýlmi, obrázky a grafiku a rozloženie stránok popísané v HTML dokumentoch. Vaadin poskytuje niekoľko preddefinovaných šablón, ale je možné si vytvárať aj vlastné. Celý vzhľad rozhrania sa dá meniť aj za behu programu jednoduchou zmenou šablóny.

Vaadin sám o sebe neobsahuje nástroje podporujúce vytváranie aplikácii s MVC architektúrou, je to ale možné použitím *IT Mill Toolkit*. Okrem jedného tutoriálu s názvom *MVC Basics in Vaadin*, ktorý sa nachádza na oficiálnej stránke[12], je ťažké nájsť k tejto téme potrebné informácie.

3.2.5 Google Web Toolkit

Google Web Toolkit (GWT) je framework pre vytváranie webových aplikácií, ktoré na komunikáciu so serverom využívajú technológiu Ajax. GWT kompiluje celý kód bežiaci u klienta do optimalizovaného JavaScript-u, aby zabezpečil jeho čo najefektívnejšie a najrýchlejšie vykonávanie. Optimalizácia prebieha napríklad tak, že sa z kódu odstraňujú nepoužívané triedy a premenné alebo sa volania metód nahrádzajú inline metódami. Vygenerovaný JavaScript je potom kompatibilný so všetkými podporovanými prehliadačmi.

Pri ladení sa aplikácia spúšťa v tzv. „development“ (vývojovom) móde, pri ktorom ešte neprebieha kompilácia do JavaScript-u, ale je vykonávaný pôvodný kód napísaný

v jazyku Java. Pre účel zobrazenia aplikácie bežiacej vo vývojovom móde je preto pre prehliadače k dispozícii zásuvný modul - GWT Developer Plugin.

GWT poskytuje dva mechanizmy na komunikáciu klientskej aplikácie so serverom - Remote Procedure Calls (RPC) framework a triedy pre posielanie http požiadaviek. Vlastné http požiadavky je vhodné využiť v prípade, že na strane servera nie je použitý Java servlet. Vtedy http triedy umožňujú komunikovať cez JSON alebo XML. RPC framework je navrhnutý na spoluprácu s Java servletmi, stará sa o serializáciu objektov a ich posielanie medzi serverom a klientom. Pri použití jedného zo spomínaných mechanizmov je možné väčšinu kódu vykonávať u klienta v prehliadači a so serverom komunikovať iba v prípade, že je to naozaj potrebné, čím sa minimalizuje množstvo prenášaných dát a zaťaženie servera.

GWT podporuje vytváranie aplikácií s Model-View-Presenter (MVP) architektúrou. Jednotlivé „presentery“ používajú na vzájomnú komunikáciu triedu HandlerManager, ktorá slúži ako zbernica udalostí – zhromažďuje informácie o vykonaných udalostiach a registruje, ktorým „presenterom“ je potrebné posielať ktoré zo zaregistrovaných udalostí.

3.2.6 Zhrnutie – výber vhodného frameworku

Na porovnanie som si vybrala niekoľko frameworkov, ktoré používajú jazyk Java. Hlavná nevýhoda prvého zo spomínaných, Sruts2, je to, že je vhodný na použitie skôr pre aplikácie menšieho rozsahu, kvôli „action based“ spôsobu generovania stránok. Vaadin sa ukázal ako menej vhodné riešenie kvôli nedostatku materiálov týkajúcich sa MVC architektúry, aj napriek natívnej podpore Ajax-u, použitiu šablón a veľkému počtu knižníc komponentov. Google Web Toolkit má oproti ostatným výhodu najmä v tom, že je vykonávaný takmer výlučne na strane klienta, komunikácia so serverom je minimálna a je k nemu dostupných množstvo materiálov a dokumentácie, preto som sa rozhodla vývoj mojej aplikácie postaviť práve na ňom.

3.3 Použité technológie

3.3.1 XHML

XHTML (*Extensible Hypertext Markup Language*) je značkovací jazyk pre vytváranie webových dokumentov. Je nasledovníkom staršieho HTML 4 a je založený na XML – to v tomto prípade umožňuje integráciu XHTML s inými technológiami založenými na XML, ako sú napríklad MathML (značkovací jazyk pre popisovanie matematických vzorcov)

alebo SVG (reprezentácia dvojrozmerných obrázkov alebo animácií). Ďalšou výhodou je to, že XHTML dokumenty sa bez problémov zobrazujú na rôznych typoch zariadení, ktoré spolupracujú s XML (najmä mobilné zariadenia).

3.3.2 CSS

CSS (*Cascading Style Sheets*) je jazyk, ktorý umožňuje popísať štýl (farby, font, pozadie, veľkosť, atď.) jednotlivých elementov v HTML alebo XML dokumentoch. Jeho hlavným cieľom je oddelenie obsahu od definície vzhľadu, čím sa zvýši prehľadnosť samotného dokumentu. Potreba vzniku kaskádových štýlov sa objavila v čase, keď bol do HTML špecifikácie pridaný element <*font*> určený na zmenu vzhľadu textu. Pri rozsiahlejších webových stránkach bolo opakované použitie <*font*> nepraktické a bolo ho nutné mnohonásobne kopírovať všade, kde bol vyžadovaný rovnaký formát písma. CSS vyriešilo tento problém tým, že štýly vložené do samostatného dokumentu môžu byť použité opakovane pre rôzne stránky.

Od verzie CSS 2.1 je možné vytvárať nezávislé kaskádové štýly pre rôzne zobrazovacie zariadenia – obrazovka počítača, tlačené dokumenty, mobilné telefóny a podobne.

3.3.3 JavaScript

JavaScript je objektovo orientovaný skriptovací jazyk, používaný najmä pri vytváraní webových stránok. JavaScript je pri vložení do stránky schopný pristupovať k jej elementom, upravovať ich alebo nad nimi vykonávať rôzne akcie. Príkladom použitia je napríklad zmena obsahu elementov, validácia formulárov na strane klienta, dynamické otváranie okien a vytváranie obsahu v nich a množstvo ďalších. JavaScript je využívaný pri vývoji interaktívnych webových aplikácií, kedy nie je vhodné, aby používateľ po každej vykonanej akcii čakal na obnovenie stránky, namiesto toho sa používajú skripty na klientskej strane pre zvýšenie interaktivity.

Alternatívou pre použitie skriptov na strane klienta je *Server-Side JavaScript*, ktorý beží na serveri, kde vie napr. komunikovať s databázou alebo pristupovať k súborom.

3.3.4 Ajax

Ajax (*Asynchronous JavaScript + XML*) je zoskupenie viacerých technológií (XHTML, XML, JavaScript, XMLHttpRequest) a používa sa pri vytváraní interaktívnych webových stránok. Vždy, keď používateľ vykoná akciu a je potrebné získať dáta zo

servera, napríklad pre prístup k databáze, kvôli validácii formulárov alebo zmene obsahu na stránke, namiesto odoslania http požiadavky a následného obnovenia stránky Ajax vygeneruje asynchrónnu požiadavku použitím XML a XMLHttpRequest. Tú server spracuje, pošle späť požadované dáta a na základe nich sa použitím JavaScriptu modifikuje obsah stránky prostredníctvom Document Object Model-u, čo je rozhranie umožňujúce dynamicky pristupovať k obsahu stránky a meniť ho.

Ďalším prínosom tejto technológie je to, že od klienta na server sa prenáša niekoľkonásobne menšie množstvo údajov. Pri použití klasického prístupu a obnove stránky po každej používateľovej akcii sa musia znova načítať nie len požadované údaje, ale aj celá stránka, spolu s grafikou, rozložením a všetkým obsahom, ktorý sa vlastne vôbec nezmenil. Pri odosielaní požiadaviek cez Ajax sa prenáša iba to, čo je naozaj potrebné na stránke aktualizovať.

3.3.5 Java

Java je objektovo orientovaný programovací jazyk, dostupný takmer na všetkých platformách – vďaka tomu, že Java kód je namiesto do strojového kódu kompilovaný do inštrukcií v binárnej forme (nazývané *Java bytecode*) pre vykonávanie v Java Virtual Machine (JVM). Dostupnosť jazyka Java pre konkrétnu platformu je potom závislá iba na dostupnosti JVM, čo charakterizuje aj slogan *„write once, run anywhere“*. Java v súčasnosti beží na osobných počítačoch, mobilných telefónoch, Blu-ray prehrávačoch, navigačných systémoch, tlačiarňach a množstve ďalších.

Pre spúšťanie programov vytvorených v Java-e je potrebný Java Runtime Environment (JRE), pre vývoj aplikácii je to Java Development Kit (JDK), ktorý poskytuje potrebné nástroje[20]. K dispozícii je viacero edícií, podľa typu vytváraných aplikácií:

- Java SE – pre vytváranie Java appletov (aplikácie bežiace v prehliadači) a desktopových aplikácií
- Java EE – pre rozsiahle a webové aplikácie
- Java ME – pre aplikácie pre mobilné zariadenia a jednočipové mikropočítače

3.3.6 MySQL

MySQL je voľne šíriteľný viacužívateľský relačný databázový systém, vydaný pod GPL licenciou a je dostupný takmer na všetkých platformách. Pre prácu s databázou používa jazyk SQL (*Structured Query Language*). Knižnice pre pripojenie k MySQL

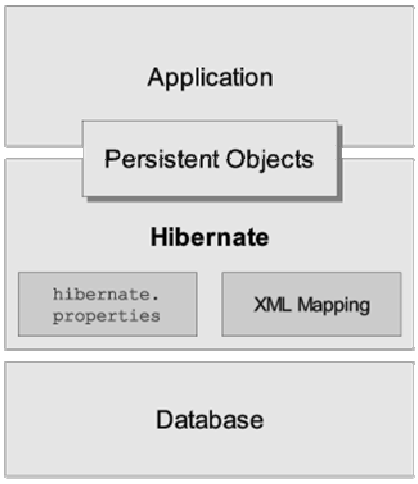
databáze sú dostupné pre množstvo programovacích jazykov, patria tam napr. PHP, C#, Java, Delphi, Python, Visual Basic a iné.

3.3.7 Hibernate

Hibernate je framework určený na objektovo-relačné mapovanie (ORM) v jazyku Java. ORM je mechanizmus používaný v objektovo orientovanom programovaní v systémoch, ktoré používajú na permanentné ukladanie dát relačné databázy. V takýchto aplikáciách sa komplikuje manipulácia s dátami, ktoré sú na jednej strane definované pomocou štruktúry objektov a na druhej strane sú ukladané do relačnej databázy vo forme tabuliek. Problémy vnikajú napríklad vtedy, keď je potrebné výsledok dotazu na databázu namapovať na premenné objektu a naopak – vzniká množstvo opakujúceho sa kódu, ktorý musí byť vždy aktualizovaný pri zmene štruktúry tabuľky alebo objektových premenných. Podobná situácia nastáva aj pri písaní všetkých dotazov zo skupiny tzv. *„CRUD operations“*, čiže operácií *create, read, update*, a *delete*. Navyše, databázu je potrebné ručne aktualizovať zakaždým, keď sa zmenia vlastnosti alebo premenné objektov.

Neprehľadný a duplicitný kód nie je jedinou nevýhodou pri práci s objektmi a relačnou databázou. Pri písaní dotazov v syntaxi konkrétneho typu databázy nastane problém v prípade, že bude potrebné zmeniť použitý databázový systém – ORM frameworky používajú na písanie dotazov jednotný jazyk, ktorý je neskôr prekladaný do jazyka špecifického pre danú databázu. Ďalšou výhodou, ktorú poskytuje ORM, je to, že pri vyberaní objektu z databázy nepotrebuje programátor ručne načítavať všetky objekty s ním súvisiace (ak sú napr. vo vzťahu *one-to-many*, *one-to-one* a pod.), pretože tie sú mu vrátené automaticky. Pri týchto vzťahoch medzi objektmi je možné povoliť kaskádové vykonávanie operácií ukladania, editácie a vymazávania – vtedy sa operácia zavolaná na objekt vykoná zároveň na všetkých jemu podradených objektoch.

K dispozícii sú dva spôsoby mapovania objektových premenných na stĺpce v relačnej databáze. Prvým je použitie anotácií – pri nich sa mapovanie definuje pomocou tzv. *meta-tagov* vkladaných priamo do kódu tried, ktoré sa vzťahujú na konkrétne premenné, resp. metódy. Druhý spôsob je vytvorenie samostatného XML súboru pre každú triedu, ktorá bude obsahovať všetky informácie pre korektné namapovanie na databázu. V obidvoch prípadoch sú definované stĺpce tabuľky, do ktorých sa ukladajú jednotlivé premenné, ktorá premenná slúži ako jednoznačný identifikátor objektu, aké sú vzťahy s ostatnými objektmi (tabuľkami) a iné doplňujúce údaje.



Obrázok 3.8. Schéma architektúry Hibernate frameworku[21]

Hibernate pracuje s tzv. *„Persistent classes“* (perzistentné triedy), čo sú triedy pri ktorých je možné ich namapovanie na databázu. Na perzistentné triedy nie sú kladené takmer žiadne obmedzenia. Hlavnou podmienkou je, že musia mať konštruktor bez argumentov a musia patriť do skupiny POJO-tried. POJO je skratka z *Plain Old Java Object*, aj keď tento pojem nie je celkom presne definovaný, je to štandardná trieda, ktorá by nemala implementovať žiadne špeciálne rozhrania (napr. Enterprise Java Bean, čo je serverový komponent používaný v sieťových klient/server aplikáciách).

Pre písanie dotazov na databázu je k dispozícii Hibernate Query Language (HQL). HQL je jazyk podobný SQL, ale na rozdiel od SQL je objektovo orientovaný – v dotazoch sa nepoužívajú samotné tabuľky a stĺpce, ale objekty a ich premenné. Podporuje dedičnosť, polymorfizmus, asociácie objektov a takmer všetky pokročilé funkcie z SQL.

3.3.8 Spring Security

Spring Security je framework určený pre použitie v rozsiahlych webových aplikáciách vytvorených v jazyku Java, ktorý poskytuje pokročilé funkcie súvisiace s autentifikáciou a autorizáciou používateľov. Je jednoducho konfigurovateľný prostredníctvom XML súboru, ktorý umožňuje nastavenie všetkých potrebných parametrov a vlastností.

V konfiguračnom súbore môže byť definovaný pevný zoznam používateľov, ich prístupové údaje a skupiny, do ktorých patria, alebo môže byť autentifikáciou poverený objekt patriaci aplikácii, ktorý (štandardne napojením sa na databázu) overí získané prihlasovacie údaje. V XML súbore sa dá nastaviť povolenie alebo zakázanie prístupu skupín používateľov ku konkrétnym URL adresám v aplikácii, prístup k stránkam prostredníctvom HTTPS protokolu alebo detekcia neplatných Session ID premenných.

Spring Security je integrovateľný s technológiami ako je OpenID (použitie jednej identity pre prihlásenie do všetkých podporovaných aplikácií), JAAS (*Java Authentication and Authorization Service* – autentifikácia a autorizácia používateľa a potvrdenie, že používateľ má právo vykonávať vyžiadané funkcie) a množstvom ďalších.

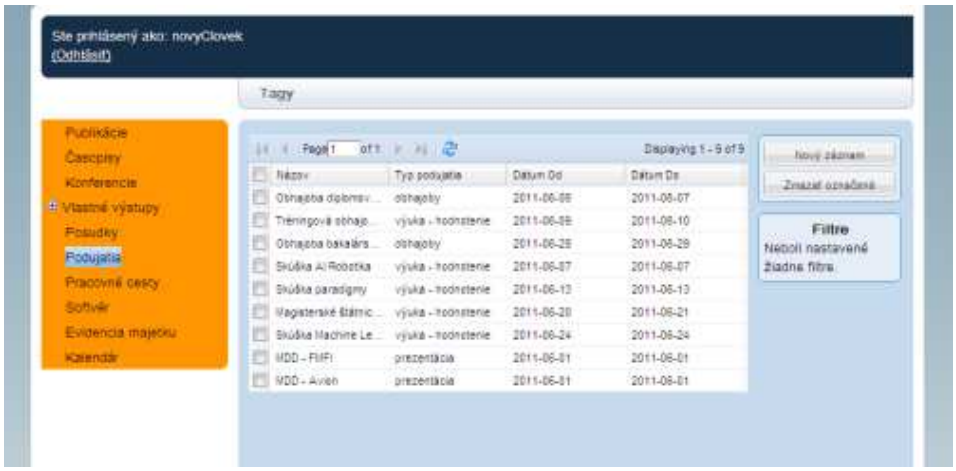
Aplikácie sú pri použití tohto frameworku automaticky chránené proti viacerým bezpečnostným rizikám. Medzi ne patrí napríklad tzv. *„Session fixation attack“*, pri ktorom útočník pošle obeti ním vytvorený identifikátor session a ak sa prostredníctvom neho obeť prihlási do účtu, útočník k nemu získa plný prístup. Takáto situácia je ošetrená vytvorením vždy nového identifikátora pri každom prihlásení.

Ďalším zo spôsobov ochrany je ochrana samotných metód objektov. Pomocou anotácii zahrnutých priamo v kóde aplikácie je ľubovoľnú metódu možné označiť ako zabezpečenú a povoliť jej vykonávanie iba pre vybrané skupiny používateľov.

4. Návrh riešenia

4.1 Návrh používateľského rozhrania

Okno aplikácie je rozdelené na štyri hlavné časti: panel s informáciami, hlavné menu, menu so zoznamom kategórií a panel pre zobrazovanie obsahu.



Obrázok 4.1. Rozloženie hlavných častí používateľského rozhrania

Panel s informáciami slúži sa zobrazovanie údajov, ako sú napríklad informácie o používateľovi, tlačidlo pre odhlásenie z aplikácie alebo chybové správy.

Hlavné menu sa nachádza pod informačným panelom a obsahuje položky pre prístup k nastaveniam, vytváraniu zoznamu globálnych štítkov a ostatným funkciám, ktoré nie sú priamo naviazané na žiadnu konkrétnu kategóriu záznamov v programe.

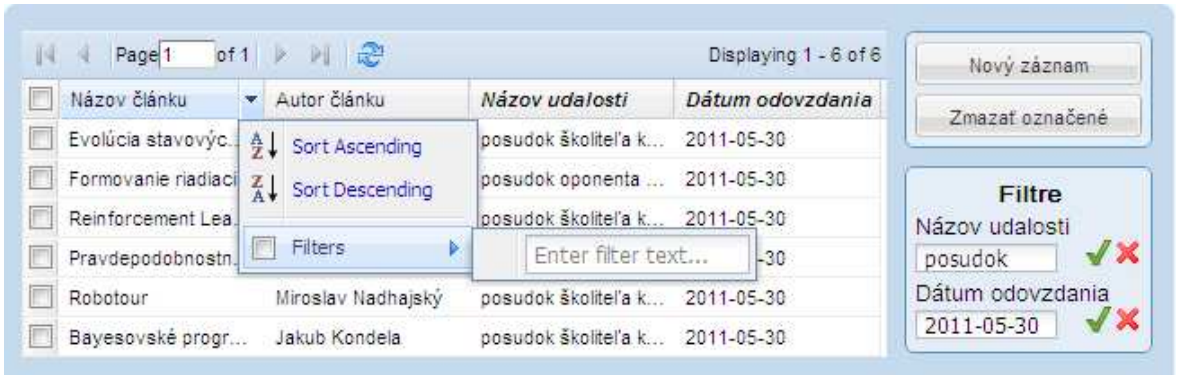
Menu pre výber kategórie je umiestnené v ľavej časti okna. Nachádza sa v ňom zoznam všetkých sekcií kategórií záznamov, ktoré aplikácia obsahuje. Po výbere niektorej z položiek sa jej obsah, resp. zoznam údajov zobrazí v strednej časti s obsahom.

Panel pre zobrazovanie obsahu slúži na zobrazenie záznamov z jednotlivých kategórií a manipuláciu s nimi. Samotný panel je ďalej rozdelený na niekoľko menších častí v závislosti od zobrazovanej kategórie.

4.1.1 Zobrazenie štandardných kategórií záznamov

Štandardné kategórie záznamov sú tie, ktoré sa od seba odlišujú iba štruktúrou ukladaných údajov a neposkytujú žiadnu špeciálnu funkcionalitu. Patria tam napríklad Publikácie, Časopisy, Konferencie, Posudky, Podujatia a pod. Pri zobrazení údajov týchto kategórií je panel s obsahom rozdelený na tri časti:

– Súhrnná tabuľka – obsahuje zoznam všetkých záznamov kategórie a niektoré ich najpodstatnejšie údaje. Tabuľka podporuje stránkovanie a zoraďovanie záznamov podľa abecedy pri jednotlivých stĺpcoch. Po kliknutí na niektorý zo záznamov sa otvorí nové okno s podrobnosťami o zázname, kde je všetky údaje možné upravovať.
– Panel s tlačidlami – obsahuje tlačidlá pre manipuláciu so záznamami, patria tam tlačidlá pre vytvorenie nového záznamu, vymazanie označených záznamov alebo ďalšie, v prípade potreby.
– Panel s aktívnymi filtrami – pri prezeraní tabuľky je možné nastaviť filtrovanie údajov v jednotlivých stĺpcoch – v tabuľke sa potom zobrazia iba tie záznamy, ktoré vyhovujú filtru. Filter môže byť reťazcový (údaj v stĺpci musí obsahovať zadaný podreťazec), číselný alebo dátumový (údaj musí byť väčší/menší ako daná hodnota). Nastavené filtre sa potom zobrazujú v bočnom paneli, kde je možné meniť ich hodnoty, alebo odstrániť.



Obrázok 4.2. Rozhranie pre zobrazenie štandardných kategórií záznamov

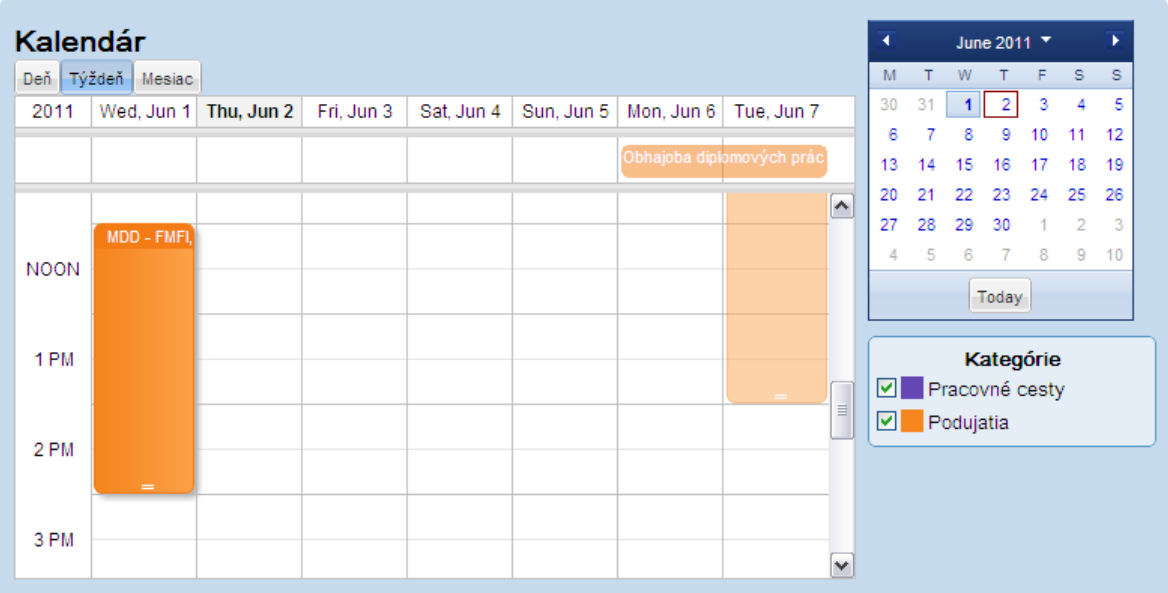
4.1.2 Zobrazovanie kategórií so špecifickou funkcionalitou

Medzi tieto kategórie patria napríklad foto a video dokumentácia, zápisník alebo kalendár. Tu musí byť rozhranie prispôsobené, pretože obsahuje funkcionalitu, ktorá nie je potrebná v žiadnej inej kategórii.

Pri foto a video dokumentácii musí panel s obsahom namiesto súhrnnej tabuľky obsahovať komponent pre prehliadanie stromovej štruktúry súborov a adresárov a pole pre nahrávanie nových súborov. Záznamy v zápisníku sa zobrazujú vo forme odstavcov pozostávajúcich z dátumu, názvu a textu poznámok.

Najviac sa od ostatných kategórií odlišuje kalendár. Kalendár obsahuje komponent, v ktorom je možné prepínať medzi troma zobrazeniami: iba aktuálny deň, najbližších 7 dní alebo aktuálny mesiac. v kalendári sa zobrazuje názov a popis udalostí, po kliknutí na niektorú z nich je možné prezrieť a editovať ostatné údaje. V pravej časti panela sa

nachádza malý kalendár, pomocou ktorého používateľ prepína aktuálny deň. Okrem neho je tam panel so zoznamom kategórií, do ktorých sú zaradené udalosti kalendára. Každá kategória má nastaviteľný štýl (vzhľad) jej patriacich udalostí. V zozname kategórií je tiež možné určiť, ktoré z nich sa v kalendári zobrazujú a ktoré nie.

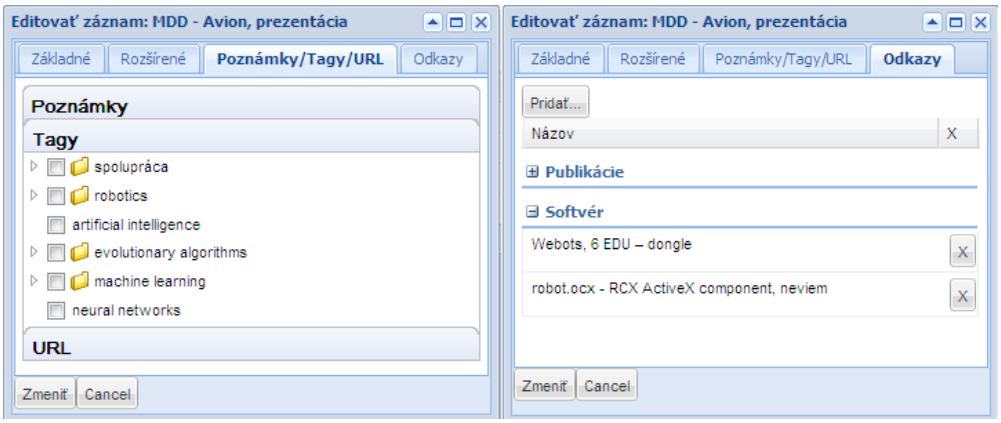


Obrázok 4.3. Rozhranie kalendára

4.1.3 Okno pre editáciu a vytváranie nových záznamov

Pri vytvorení nového, resp. editovaní existujúceho záznamu sa údaje zobrazujú v novom, kvôli možnosti pracovať naraz s viacerými záznamami. Okno je zložené s viacerých záložiek:

- Základné údaje, Rozširujúce údaje - obsahujú zoznam polí ukladaných pri zázname, polia sú rozdielne v každej kategórii
- Poznámky/Štítky/URL – tam sa nachádza textové pole pre písanie poznámok, zoznam zadefinovaných globálnych štítkov a tabuľka pre upravovanie zoznamu pomenovaných URL adries patriacich záznamu. Táto a nasledujúca záložka je spoločná pre všetky kategórie v aplikácii.
- Odkazy – obsahuje zoznam súvisiacich záznamov z aplikácie rozdelených do kategórií



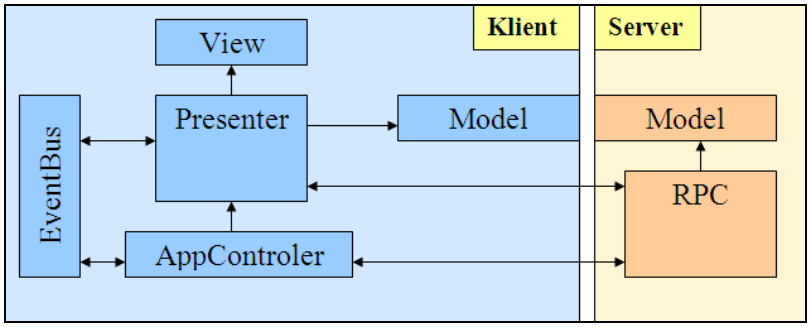
Obrázok 4.4. Ukážka editovacieho okna

4.2 Architektúra aplikácie

Aplikácia Pracovná plocha výskumníka používa Model View Presenter (MVP) architektúru, tá je v Google Web Toolkit frameworku priamo podporovaná a sú na jej implementáciu dostupné všetky potrebné nástroje. MVP architektúru je vhodné použiť pri vytváraní rozsiahlych aplikácií, aby bolo možné oddeliť od seba vrstvy zodpovedné za:

- zobrazovanie a používateľské rozhranie (View), ktoré nie je priamo naviazané na model, t.j. táto vrstva nevie o jeho existencii a komunikuje výhradne s príslušným presenterom
- reprezentáciu dát (Model), pričom tie sú uchovávané v MySQL databáze a pomocou objektovo relačného mapovania spojené s objektmi, s ktorými neskôr aplikácia pracuje
- operácie vykonávané nad dátami, ich aktualizáciu a spracúvanie akcií používateľa (Presenter)

Tým, že sú tieto vrstvy od seba oddelené, poskytuje niekoľko výhod. Nezávislosť časti zodpovednej za zobrazovanie od modelu umožňuje znovu použiť jej kód v inom kontexte. To, že presenter nie je zodpovedný za vytváranie používateľského rozhrania a nie je od neho nijakým spôsobom závislý, uľahčuje jeho testovanie a tiež možnosť použiť ho spolu s iným view modulom.



Obrázok 4.5. Schéma architektúry aplikácie s použitím MVP vzoru

Ako vidno na obrázku 4.5, väčšia časť aplikácie je sústredená u klienta, teda v internetovom prehliadači. Na serveri sa nachádza iba model a RPC (Remote Procedure Calls) modul, teda rozhranie, ktoré umožňuje asynchrónnu komunikáciu klienta so serverom. Model (triedy reprezentujúce jednotlivé tabuľky v databáze) má k dispozícii na používanie aj časť Presenter, preto je model súčasťou kódu na strane klienta.

4.2.1 Remote Procedure Calls

RPC volanie sa vykonáva vždy, keď je potrebné pristupovať k dátam zo servera. Vtedy sa vygeneruje http požiadavka a prebieha automatická serializácia ojektov, ktoré je potrebné pri požiadavke preniesť na server a potom následne spať ku klientovi pri generovaní odpovede na požiadavku. Implementácia RPC mechanizmu má tri časti:

1. rozhranie rozširujúce `RemoteService` rozhranie (`CommService.java`) – obsahuje zoznam deklarácií všetkých metód, ktoré je možné cez RPC mechanizmus zavolať
2. trieda rozširujúca triedu `RemoteServiceServlet`, ktorá zároveň implementuje rozhranie `CommService.java` (`CommServiceImpl.java`) – obsahuje implementáciu všetkých metód deklarovaných v `CommService.java`. Táto trieda je umiestnená na strane servera.
3. asynchrónne rozhranie pre `CommService.java` (`CommServiceAsync.java`) – bude automaticky priradené k jeho neasynchrónnej verzii. Obsahuje rovnaké metódy s tým rozdielom, že všetky majú návratový typ `void` a k zoznamu argumentov je pridaný argument typu `AsyncCallback` parametrizovaný pôvodným návratovým typom metódy, prostredníctvom ktorého bude klient notifikovaný o vykonaní metódy. Pri volaní zo strany sa potom použije asynchrónna metóda. Ukážka použitia:

```
public interface CommService extends RemoteService {
  public String myMethod(String s);
}
```

```
interface CommServiceAsync {
  public void myMethod(String s, AsyncCallback<String> callback);
}
```

4.2.2 View a Presenter

Ako už bolo spomínané, view je zodpovedný za vytvorenie používateľského rozhrania a presenter pracuje s dátami a spracúva akcie používateľa. Aby sa dosiahlo spracovanie súvisiacich udalostí na jednom mieste, je dobré vytvoriť samostatnú dvojicu view a presentera pre každý súvislý celok v aplikácii (napr. pre každú stránku, alebo pre časť

stránky, ktorá sa vyskytuje na viacerých miestach). Každý view má teda priradený presenter (nie nutne iba jeden), ktorý aktualizuje zobrazované údaje a reaguje na udalosti spojené s príslušným view. Ich prepojenie je realizované prostredníctvom rozhrania, ktoré je definované v rámci každého presentera. View potom implementuje toto rozhranie. Štandardne sa v ňom nachádzajú dva typy metód:

- metódy, pomocou ktorých vie presenter zmeniť obsah dát zobrazovaných vo view
- metódy, cez ktoré presenter pristupuje ku komponentom, interakcia s ktorými vyvoláva nejakú akciu (napr. tlačidlá, položky menu pod.). Presenter k nim musí mať prístup, pretože on je zodpovedný za vykonávanie takýchto akcií.

V mojej aplikácii majú dvojice view a presenterov nasledujúce úlohy:

- InfoTextPanelView/Presenter – zobrazovanie panela s informáciami o používateľovi
- MainMenuView/Presenter – podobne ako ľavé menu, operácie s menu s kategóriami
- LeftMenuView/Presenter – zobrazenie ľavého menu a vyvolanie akcií podľa vybratej položky
- TagsWindowView/Presenter – okno pre vytváranie a editáciu globálnych štítkov
- CenterPartView/Presenter – súhrnná tabuľka so záznamami v strednom paneli s obsahom
- EditRecordView/Presenter – okno pre editáciu záznamu
- NewRecordView/Presenter - okno pre pridávanie nového záznamu
- RecordListView/Presenter – pri vkladaní odkazov na súvisiace záznamy v aplikácii – zobrazuje okno so zoznamom všetkých existujúcich záznamov
- RecordRef1View/Presenter – záložka v okne pri vkladaní alebo editovaní záznamov, ktorá obsahuje formulár pre vloženie poznámok, štítkov a URL adries
- RecordRef2View/Presenter – podobne ako v predchádzajúcom prípade, záložka pre manipuláciu so súvisiacimi záznamami
- Okrem vyššie spomínaných tam ešte patria všetky tie, ktoré sú zodpovedné za zobrazovanie obsahu v kategóriách so špecifickou funkcionalitou (kalendár, zápisník, foto a video dokumentácia a pod.)

4.2.3 AppController

Trieda `AppController.java` je vo všeobecnosti zodpovedná za to, čo nie je špecifické pre jeden presenter, ale týka sa behu celej aplikácie. Vytvára sa v nej rozloženie

jednotlivých častí používateľského rozhrania a vytvorenie presenterov a im prislúchajúcich view, ktoré budú v týchto častiach zobrazené (napr. panel s hlavným menu patrí MainMenuPresenter-u, v strednej časti je to CenterPartPartPresenter, a pod.). S tým súvisí aj vytváranie histórie v prehliadači na základe toho, ako sa mení obsah stránky – kvôli tomu, že aplikácia beží stále na jednej stránke a o zmenu obsahu na nej sa stará AJAX, je potrebné v prehliadači „umelo“ vytvoriť históriu navštívených stránok. Tento mechanizmus pridáva k URL adrese stránky záložku, ktorá identifikuje aktuálne zobrazovaný obsah. Adresa stránky má potom nasledovný tvar:

http://www.aplikacia.com/index.htm#view1

4.2.4 EventBus

EventBus (premenná typu `HandlerManager`) v aplikácii slúži ako „zbernica“ udalostí, prostredníctvom ktorej medzi sebou komunikujú AppController a jednotlivé presentery. Je potrebný preto, lebo presentery nemajú žiadne informácie o existencii iných častí aplikácie a vedia pristupovať iba k im prislúchajúcim view, zbernici udalostí a RPC rozhraniu. Potrebujú však byť informované o udalostiach, ktoré v aplikácii nastali. Posielanie informácií o udalostiach cez zbernicu funguje nasledovne:

- Pri štarte aplikácie si trieda `AppController.java` vytvorí objekt typu `HandlerManager` (EventBus) a referenciu na tento objekt posiela jednotlivým presenterom pri ich vytvorení
- AppController a každý z presenterov sa „zaregistruje“ pre odoberanie udalostí, o ktoré má záujem, teda oznámi zbernici, požadovaný typ udalosti a akciu, ktorá má byť vykonaná v prípade, že udalosť nastane. Nasledujúci kód je ukážkou tejto situácie.

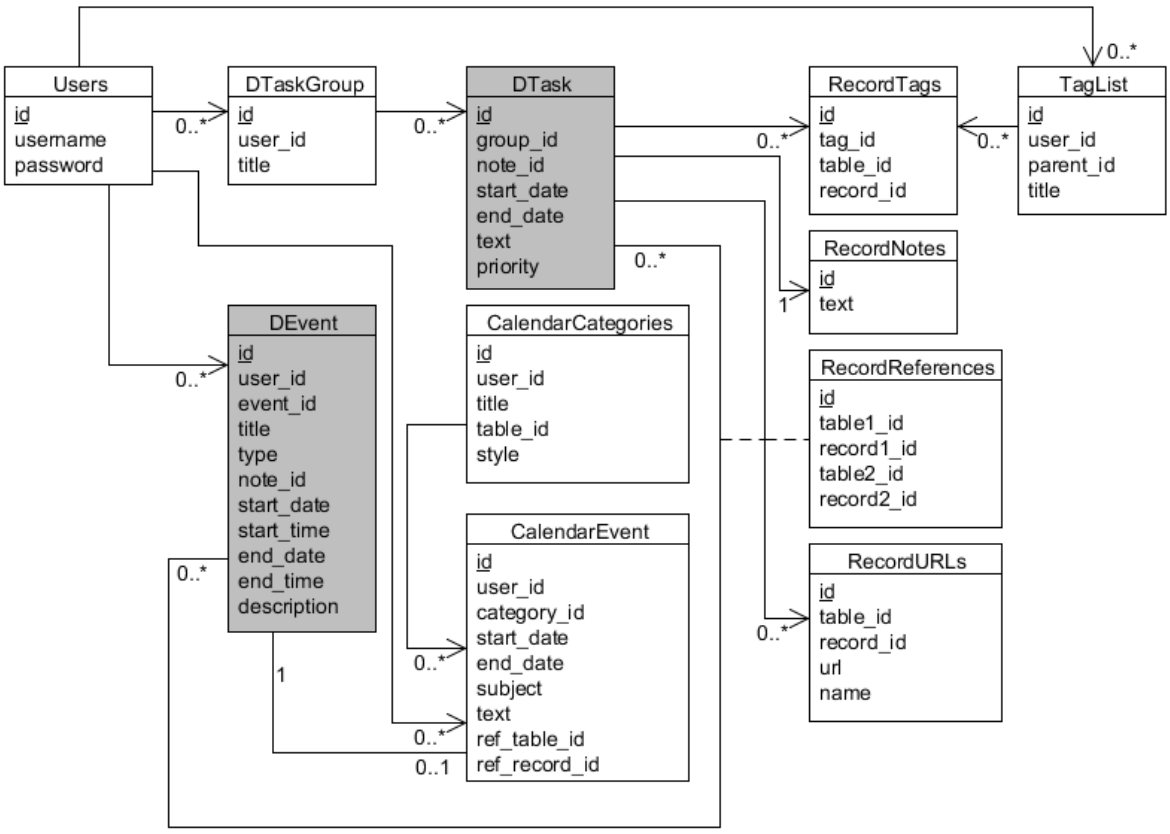
```
eventBus.addHandler(SomeEvent.TYPE,
      new SomeEventHandler() {
        public void onDoSomething(SomeEvent event) {
          doProcessEventInformation(event.getProperty());
        }
      });
```

- V prípade, že nastane nejaká udalosť, presenter zodpovedný za príslušnú časť aplikácie oznámi zbernici, že udalosť nastala a zbernica o tom následne informuje všetky presentery (a AppController).

Čím je aplikácia rozsiahlejšia, tým väčší počet udalostí je cez zbernicu potrebné posielať, preto sa odporúča, aby bola zbernica informovaná iba o udalostiach, ktoré majú význam pre celú aplikáciu, nie iba jeden konkrétny presenter.

4.3 Návrh databázy

Na obrázku 4.6 je znázornená časť tabuliek v databáze a vzťahy medzi nimi. Kvôli zachovaniu prehľadnosti sa v ňom nenachádzajú všetky tabuľky, ktoré databáza obsahuje. Na ukážku vzájomných vzťahov som vybrala tabuľky, ktoré obsahujú záznamy z kategórie Podujatia (*DEvent*) a Zoznam úloh (*DTask*). Tabuľky pariace ostatným kategóriám neuvádzam, pretože znázornenie vzťahov by v tomto prípade by bolo takmer identické (okrem niektorých prípadov) ako pri vybraných ukážkových tabuľkách.



Obrázok 4.6. Diagram zobrazujúci vzťahy medzi tabuľkami databázy

V tabuľke *Users* sa nachádza zoznam všetkých používateľov aplikácie. Keďže informácie o používateľovi nie sú pre aplikáciu dôležité, ukladajú sa iba najzákladnejšie údaje – prihlasovacie meno a heslo. Každý používateľ si definuje zoznam globálnych štítkov, ktoré obsahuje tabuľka *TagList*. Každý štítok má svoj názov a odkaz na nadradený štítok v rámci hierarchickej štruktúry.

Záznamy z kategórie Poznámky sa nachádzajú v tabuľke *DTask*. Každá poznámka obsahuje informáciu, do ktorej skupiny patrí (tabuľka *DTaskGroup*), prioritu, text a dátum začiatku a ukončenia.

Podujatia (*DEvent*) v tejto schéme reprezentujú typ záznamov, ku ktorým je možné priradiť udalosti v kalendári (*CalendarEvent*). Ak nejaké podujatie má priradenú udalosť,

je medzi nimi vzťah typu *one-to-one*, teda každá existujúca udalosť má priradený nejaký záznam. Tabuľka *CalendarCategories* obsahuje zoznam všetkých kategórií programu, do ktorých tieto záznamy spadajú.

Každému záznamu v programe, bez ohľadu na kategóriu, je možné priradiť textové poznámky (*RecordNotes*), zoznam štítkov (*RecordTags*), pomenovaných URL adries (*RecordURLs*) a odkazov na iné záznamy (*RecordReferences*). Všetky spomínané tabuľky okrem *RecordNotes* majú dva údaje, podľa ktorých jednoznačne rozoznávajú záznam v rámci aplikácie – *table_id* a *record_id*. Prvý z uvedených je krátky reťazec identifikujúci jednu kategóriu záznamov, a teda aj jednu tabuľku databázy. Tento identifikátor je definovaný ako statická premenná v triede, ktorá reprezentuje záznam z danej kategórie. Druhý, *record_id*, je identifikátor záznamu v rámci tabuľky. Tieto dva údaje sa používajú všade, kde je potrebné odkazovať na záznam z vopred neurčenej kategórie, ako napríklad v tabuľke *CalendarEvent*.

5. Implementácia

5.1 Použité nástroje a knižnice

Ako už bolo spomínané v kapitole 3 v časti Prehľad frameworkov, na vývoj aplikácie bol použitý Google Web Toolkit framework. Súčasťou samotného GWT je knižnica, ktorá obsahuje niekoľko komponentov pre použitie pri vytváraní užívateľského rozhrania. Tieto komponenty však poskytujú iba základnú funkcionalitu, chýbajú medzi nimi napríklad nástroje pre vytváranie zložitejších tabuliek s podporou stránkovania, filtrovania a zoraďovania údajov, nástroje pre prácu so zložitejšími formulármi alebo stromovou štruktúrou. Táto knižnica je preto v mojej aplikácii využívaná v menšej miere, namiesto nej používam knižnicu *Ext GWT*[23], ktorá obsahuje všetky vyššie spomenuté komponenty a množstvo ďalších. Neobsahuje však žiaden zložitejší kalendár, takže na ten bolo znova potrebné použiť iný nástroj. V tomto prípade ním bol *Gwt-cal*[24], čo je open source kalendár pre GWT podobný napríklad tomu, s ktorým pracuje aplikácia *Google Calendar*.

Ďalšími použitými nástrojmi sú Spring Security framework, ktorý je zodpovedný za autentifikáciu a autorizáciu používateľov a Hibernate framework pre objektovo-relačné mapovanie objektov na databázu. Oba sú bližšie popísané v kapitole 3 v časti Použité technológie.

5.2 Štruktúra zdrojových kódov

Zdrojový kód aplikácie je rozčlenený na niekoľko častí. Na najvyššej úrovni je to rozdelenie na balíčky (*packages*) *client* a *server*, teda na kód, ktorý je vykonávaný u klienta a na serveri. Obidva balíčky sú ešte rozdelené na ďalšie časti.

- ***server:*** obsahuje iba jednu triedu, `CommServiceImpl.java`, čo je implementácia triedy pre komunikáciu klienta so serverom (kapitola 4.2.1 – Remote Procedure Calls)
- ***server.auth*:** triedy, ktorých implementáciu vyžaduje Spring Security framework, zabezpečujú autentifikáciu používateľa
- ***client***: tu sa nachádza trieda `AppController.java` (kapitola 4.2.3 – AppController), rozhranie pre RPC mechanizmus (`CommService.java` a `CommServiceAsync.java`) a trieda, ktorá je „*vstupným bodom*“ pre aplikáciu – `PracPlochaVyskumnika.java`. Táto trieda obsahuje metódu

`onModuleLoad()`, ktorá sa vykoná ako prvá a je zodpovedná za vytvorenie AppController-a, zbernice udalostí a RPC rozhrania.

- ***client.event***: udalosti, ktoré sa v aplikácii posielajú prostredníctvom zbernice *eventBus* (kapitola 4.2.4 - EventBus) medzi presentermi a AppComtroller-om.
- ***client.localevent***: udalosti, ktoré sú posielané iba v rámci jedného presentera a o ich vykonaní nie sú informované ostatné časti aplikácie
- ***client.fieldinfo***: triedy, ktoré obsahujú informácie o poliach nachádzajúcich sa vo formulári pre pridávanie a editáciu záznamov. Ich úloha je podrobnejšie popísaná v podkapitole 5.4 – Okno pre vytvorenie a editáciu záznamu.
- ***client.model***: triedy reprezentujúce dáta, s ktorými narába aplikácia
- ***client.shared***: tu sú zhromaždené všetky ostatné triedy, ktoré aplikácia využíva. Patria tam napríklad pomocné triedy pre jednotlivé view, ktoré sú potrebné pre zobrazenie dát v niektorých komponentoch, ako tabuľky a panely pre zobrazenie stromovej štruktúry.
- ***client.view, client.presenter***: triedy reprezentujúce view presentery (kapitola 4.2.2 – View a Presenter)

5.3 Načítanie existujúcich kategórií

Zoznam všetkých kategórií, ktoré aplikácia obsahuje, sa nachádza v XML súbore `menuItems.xml`. Na základe tohto dokumentu sa zároveň generuje ľavé menu s kategóriami. Má nasledujúcu štruktúru:

```
<menu>
    <item dataClass="DPublication">
        <text>Publikácie</text>
    </item>

    <item dataClass="DJournal">
        <text>Časopisy</text>
    </item>

    ...

    <item category="CALENDAR">
       <text>Kalendár</text>
    </item>
</menu>
```

Koreňový element `<menu>` obsahuje elementy `<item>`, z ktorých každý popisuje jednu kategóriu záznamov. Každý z nich obsahuje jeden z dvoch parametrov:

- *dataClass* – názov triedy, ktorá reprezentuje záznam z niektorej zo štandardných kategórií

– *category* – identifikátor (nie nutne názov triedy) kategórie, na základe neho sa aplikácia pri kliknutí na položku v menu rozhodne, ktorý view (a jemu príslušný presenter) vygeneruje obsah stránky

`<item>` obsahuje vnorený element `<text>`, ktorého obsah je samotný text položky v menu a v prípade potreby viacúrovňového menu sa v ňom môžu nachádzať aj ďalšie `<item>` elementy.

Po tom, čo užívateľ klikne na niektorú položku menu, sa vykonajú nasledujúce akcie:

– Na základe parametra v elemente `<item>` sa zistí, či je potrebné načítať dáta štandardnej kategórie alebo niektorej špeciálnej.
– Ak sa načítava špeciálna kategória, podľa hodnoty parametra *category* sa zavolá príslušný presenter
– Ak sa načítava štandardná kategória, o vytvorenie obsahu sa stará `CenterPartPresenter.java`. Ten má za úlohu vytvoriť tabuľku zo súhrnným zoznamom všetkých záznamov v kategórii, ktorá zobrazuje vybranú podmnožinu z údajov ukladaných pri kategórii. Na to potrebuje:
    1. počet stĺpcov tabuľky a ich názvy
    2. názvy metód patriacich triede reprezentujúcej aktuálnu kategóriu (názov triedy = parameter *dataClass*), ktoré vrátia hodnotu pre konkrétny stĺpec tabuľky. Príklad: ak sa majú v tabuľke zobraziť údaje Názov, Dátum a Typ, názvy metód majú hodnoty *getTitle*, *getDate*, a *getType*.
    3. samotné dáta zobrazované v tabuľke

Dôvod, prečo je potrebné poznať názvy vyššie spomínaných metód, je nasledovný: aplikácia potrebuje od triedy reprezentujúcej kategóriu dostať údaje, ktoré vopred vôbec nepozná. Navyše, vopred nepozná ani samotnú triedu, s ktorou chce pracovať (keďže počet takýchto tried (a kategórií) je teoreticky neobmedzený). Pravdepodobne jediný spôsob, ako takéto údaje získať, je použiť knižnicu Java Reflection, ktorá vie vyvolať metódu objektu, pričom má k dispozícii reťazec obsahujúci názov triedy a názov metódy. Nasledujúci kód je ukážkou tejto situácie:

```
String className = “someName”;
String methodName = “getSomeData”;
Class c = Class.forName(className);
//druhy parameter metody getMethod je zoznam typov argumentov
Method m = c.getMethod(methodName, null);
Constructor ct = c.getConstructor(null);
Object o = ct.newInstance(null);
//druhy parameter metody invoke je zoznam argumentov
Object methodResult = m.invoke(o, null);
```

Na to, aby bolo možné vyššie spomínané informácie získať, všetky triedy reprezentujúce štandardnú kategóriu musia mať metódy:

– `ArrayList<String> getTableHeaders()` – pre získanie názvov stĺpcov
– `ArrayList<String> getMethodsNames()`– pre získanie názvov metód

Patria tam aj niektoré ďalšie, ale tie pre vysvetlenie princípu fungovania načítania kategórií nie sú dôležité. Každá takáto trieda teda musí byť teda dedičom triedy `GeneralDataClass.java`, kde sú tieto *„povinné“* metódy deklarované ako abstraktné metódy a musia byť následne v triede, ktorá od nej dedí, implementované.

5.4 Okno pre vytvorenie a editáciu záznamu

Pri vkladaní (pri editácii je postup rovnaký) záznamu patriacemu štandardnej kategórii je za vytvorenie formulára zodpovedný `NewRecordPresenter.java`. Ten potrebuje poznať zoznam polí, ktoré formulár obsahuje a ich typ (napr. jednoduchý textový vstup, textové pole, zaškrtávacie políčko, dátum a pod), pričom tie sú pri každej kategórii odlišné. Keďže formulár je rozdelený na 2 časti – základné a rozširujúce polia, každá trieda reprezentujúca štandardnú kategóriu musí aplikácii poskytnúť informáciu o zozname polí, ktoré sa tam nachádzajú. Na to slúžia dve metódy (deklarované v triede `GeneralDataClass.java`):

– `ArrayList<FieldInfo> getBasicFormFields()`
– `ArrayList<FieldInfo> getAdditionalFormFields()`
– Metódy teda vrátia zoznam objektov typu `FieldInfo`, ktorý obsahuje všetky potrebné informácie pre `NewRecordPresenter.java`, aby vedel vygenerovať formulár, spracovať hodnoty vložené užívateľom a vložiť ich do databázy. `FieldInfo.java` je abstraktná trieda, pre popísanie vlastností poľa je potrebné použiť niektorú z tried, ktoré sú jej dedičmi. Tieto triedy (takmer) bez výnimky obsahujú:
  – názov poľa vo formulári,
  – informáciu o tom, či je pole povinné
  – hodnotu poľa (potrebnú hlavne pri editácii), jej typ závisí od konkrétnej triedy (napr. v triede `StringFieldInfo` = String, `FloatFieldInfo` = Float, a pod.)

  Okrem týchto informácií môže navyše podľa potreby každá z tried obsahovať niektoré ďalšie.

6. Inštalácia a spustenie

Aplikácia vyžaduje na spustenie server Apache Tomcat 6. Použitá je MySQL databáza, ale po úprave nastavení je možné použiť aj inú.

V priloženom CD sa nachádzajú dva súbory:

- PracPlochaVyskumnika.sql – obsahuje vyexportovanú databázu, tento súbor je potrebné importovať do databázy, ktorú bude program používať. Názov databázy a prístupové údaje k nej je potrebné nastaviť v konfiguračnom súbore (popísané nižšie)
- PracPlochaVyskumnika.war – tento súbor sa umiestni na Apache Tomcat server. Je potrebné vykonať nasledujúce kroky:
    1. prihlásiť sa do Tomcat administrácie – na hlavnej stránke cez odkaz v časti *Administration -> Tomcat Manager*
    2. v administrácii v časti *Deploy -> WAR file to deploy* vybrať súbor *PracPlochaVyskumnika.war* a potvrdiť stlačením tlačidla *Deploy*. Po úspešnom vykonaní operácie by sa mala Pracovná plocha výskumníka objaviť v zozname nainštalovaných aplikácií (*Applications*).

Pred spustením je potrebné upraviť konfiguračný súbor pre prístup k databáze – hibernate.cfg.xml. nachádza sa v priečinku

Tomcat 6.0/webapps/PracPlochaVyskumnika/WEB-INF/classes

V ňom sa zmenia nastavenia v časti *Database connection settings*:

```
<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>
```

- databaseName: názov databázy
- databaseUser: užívateľ databázy
- password: heslo pre užívateľa

Aplikácia sa spúšťa znova v Tomcat administrácii v zozname aplikácií po kliknutí na *Start*. Nachádza sa potom na adrese <Tomcat-url>/PracPlochaVyskumnika, kde <Tomcat-url> je adresa Apache Tomcat servera.

7. Záver

Cieľom mojej bakalárskej práce bolo analyzovať a porovnať technológie a možné prístupy k vytvoreniu pomerne rozsiahlej aplikácie, ktorá je určená na uchovávanie štruktúrovaných údajov rôzneho typu, ich efektívne vyhľadávanie a zároveň poskytuje základné funkcie organizéra.

V práci som porovnala niekoľko existujúcich aplikácií, ktoré sú svojou funkcionalitou čiastočne podobné Pracovnej ploche výskumníka, ukázalo sa však, že je nesmierne ťažké nájsť takú, ktorá by mala potrebné funkcie a spĺňala základné požiadavky, ktoré boli pre mnou vytváranú aplikáciu stanovené.

Pri vývoji bol použitý framework Google Web Toolkit, ktorý oproti iným dostupným nástrojom poskytuje niekoľko výhod. Príkladom je podpora vytvárania rozsiahlych aplikácií podľa architektúry MVP a použitie technológie Ajax pre komunikáciu so serverom. Väčšia časť kódu písaného v jazyku Java je kompilovaná do optimalizovaného JavaScript-u, ktorý je vykonávaný na strane klienta, pričom je zabezpečená jeho kompatibilita s prehliadačmi.

Podarilo sa mi vytvoriť funkčnú aplikáciu, v ktorej sú implementované základné funkcie potrebné pre jej používanie, medzi nimi pridávanie poznámok, odkazov a globálnych štítkov k záznamom, vytváranie vzájomných vzťahov, kalendár a funkcie pre filtrovanie záznamov podľa viacerých kritérií. Aplikácia obsahuje niekoľko kategórií, do ktorých je možné záznamy rozdeliť. Zoznam týchto kategórií nie je konečný, keďže je systém navrhnutý tak, aby ho bolo možné jednoducho rozširovať o nové.

Aplikácia, ktorá je výsledkom mojej práce, neobsahuje všetku funkcionalitu popísanú v zozname požiadaviek, preto by bolo vhodné v jej vývoji ďalej pokračovať a rozširovať ju, aby mohla byť plnohodnotným nástrojom pre správu informácií a údajov rôzneho typu, efektívnu organizáciu pracovného času a v konečnom dôsledku byť veľkým prínosom pre jej používateľa.

8. Zoznam použitej literatúry

[1] NeoMem. [online]
<*http://www.neomem.org/*>

[2] 2008. PIM Xtreme – About. [online]
<*http://dgtalize.com/products/pimx*>

[3] ContactOffice – Virtual office. [online]
<*http://www.contactoffice.com/*>

[4] EndNote. [online]
<*http://www.endnote.com/*>

[5] Model-View-Presenter Pattern. [online]
<*http://msdn.microsoft.com/en-us/library/ff647543.aspx*>

[6] RAIBLE, Matt. 2010. Comparing JVM web frameworks. [online]
<*http://static.raibledesigns.com/repository/presentations/Comparing_JVM_Web_Frameworks_33rdDegree.pdf*>

[7] ROUGHLEY, Ian. 2006. Starting Struts2. [online]
<*http://www.infoq.com/minibooks/starting-struts2*>

[8] CHILDERS, Shaun. 2008. Action-based or Component-based MVC? [online]
<*http://shaunchilders.com/node/8*>

[9] 2007. What is Java Server Faces? [online]
<*http://www.roseindia.net/jsf/whatisjsf.shtml*>

[10] BRANICKÝ, Marek. 2003. Java Servlets - predstavenie technológie. [online]
<*http://interval.cz/clanky/java-servlets-predstavenie-technologie/*>

[11] Book of Vaadin. [online]
<*http://vaadin.com/download/current/docs/book-of-vaadin.pdf*>

[12] MVC Basics in Vaadin. [online]
<*http://vaadin.com/wiki/-/wiki/Main/MVC%20Basics%20in%20Vaadin?p_r_p_185834411_title=MVC%20Basics%20in%20Vaadin*>

[13] RAMSDALE, Chris. 2010. Large scale application development and MVP. [online]
<*http://code.google.com/intl/sk-SK/webtoolkit/articles/mvp-architecture.html*>

[14] GWT - Communicate with a Server. [online]
<*http://code.google.com/intl/sk-SK/webtoolkit/doc/latest/DevGuideServerCommunication.html*>

[15] 2002. XHTML 1.0 The Extensible HyperText Markup Language (Second Edition). [online]
<http://www.w3.org/TR/xhtml1/>

[16] 2011. CSS 2.1 Specification. [online]
<http://www.w3.org/TR/CSS21/>

[17] 2010. JavaScript overview. [online]
<https://developer.mozilla.org/en/JavaScript/Guide/JavaScript_Overview>

[18] GARETT, Jesse James. 2005. Ajax – a new approach to web applications. [online]
<http://www.adaptivepath.com/ideas/e000385>

[19] The Java programming language. [online]
<http://groups.engin.umd.umich.edu/CIS/course.des/cis400/java/java.html>

[20] LEAHY, Paul. Java platforms. [online]
<http://java.about.com/od/gettingstarted/a/javatechnologies.htm>

[21] KING, G. – BAUER, C. – ANDERSERN, M. R. - BERNARD, E. – EBERSOLE. S. – FERENTSCHIK, H. Hibernate Reference Documentation. [online]
<http://docs.jboss.org/hibernate/core/3.6/reference/en-US/pdf/hibernate_reference.pdf>

[22] ALEX, Ben – TAYLOR, Luke. Spring Security - Reference Documentation. [online]
<http://static.springsource.org/spring-security/site/docs/3.1.x/reference/springsecurity-single.html>

[23] Ext GWT - Internet Application Framework for GWT. [online]
<http://www.sencha.com/products/extgwt/>

[24] gwt-cal - an open-source calendar widget. [online]
<http://code.google.com/p/gwt-cal/>

9. Prílohy

Prílohy sa nachádzajú na priloženom CD, ktoré obsahuje:

- Zdrojové kódy aplikácie
- Export databázy (SQL)
- Súbor pre inštaláciu aplikácie na server Apache Tomcat 6 (.war)