Sem vložte zadanie Vašej práce.

České vysoké učení technické v Praze

Fakulta informačních technologií

Katedra počítačových systémů

Bakalárska práca

Uživatelské rozhraní rezervačního systému učeben FIT

Ivan Prokipčák

Vedúci práce: Ing. Ondřej Guth, Ph.D.

11. mája 2015

Poďakovanie

V prvom rade by som chcel poďakovať môjmu vedúcemu práce Ing. Ondřejovi Guthovi, Ph.D., za jeho odborné vedenie, cenné rady ale najmä ústretovosť a trpezlivosť počas celej doby návrhu či implementácie. Veľká vďaka patrí aj mojim rodičom a najbližším, ktorí pri mne stáli a podporovali ma počas celej doby štúdia. Napokon ďakujem aj mojim spolupracovníkom na tomto projekte.

Prehlásenie

Prehlasujem, že som predloženú prácu vypracoval(a) samostatne a že som uviedol(uviedla) všetky informačné zdroje v súlade s Metodickým pokynom o etickej príprave vysokoškolských záverečných prác.

Beriem na vedomie, že sa na moju prácu vzťahujú práva a povinnosti vyplývajúce zo zákona č. 121/2000 Sb., autorského zákona, v znení neskorších predpisov, a skutočnosť, že České vysoké učení technické v Praze má právo na uzavrenie licenčnej zmluvy o použití tejto práce ako školského diela podľa § 60 odst. 1 autorského zákona.

V Prahe 11. mája 2015 .....

Odkaz na túto prácu

Prokipčák, Ivan. *Uživatelské rozhraní rezervačního systému učeben FIT*. Bakalárska práca. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2015.

Abstrakt

Táto práca sa zaoberá návrhom a tvorbou užívateľského rozhrania rezervačného systému učební FIT, formou webovej aplikácie postavenej na platforme Java EE.

Klíčová slova rezervačný systém učební, Java EE, webová aplikácia, Java PrimeFaces, JSF

Abstract

This bachelor's thesis describes design and implementation of user interface of FIT schoolrooms reservation system as a web application built on the Java EE platform.

Keywords schoolroom reservation system, Java EE, web application, , Java PrimeFaces, JSF

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

Naša fakulta obsahuje veľké množstvo skvelých softvérov a podporných aplikácií zjednodušujúcich či spríjemňujúcich štúdium. Niektoré vypomáhajú pri výučbe študentov, ich testovaní alebo hodnotení iné zasa informujú či umožňujú študentom upravovať výučbu podľa ich predstáv. K fakulte s technickým zameraním tieto možnosti neodmysliteľne patria, a preto je dôležité ich neustále vytváranie, zlepšovanie a rozširovanie o mnohé ďalšie, ktoré doposiaľ chýbajú.

Škola má byť miesto, kde sa študenti radi vracajú, prípadne tam trávia väčšinu času vyhradeného pre štúdium. Z toho dôvodu je dobré poskytovať študentom priestory aj mimo výuky vo forme študovní či učební. Ak chceme pokračovať trendom automatizácie, tak zistíme, že takýto nástroj, vhodný pre správu či rezervácie priestorov školy fakulta doposiaľ nemá. Preto sme v rámci bakalárskej práce prijali projekt zhotovenia rezervačného systému učební pre Fakultu informačných technológií, realizovaného vo forme webovej aplikácie. Je predsa aj v našom záujme, ako študentov, dopomôcť k zlepšeniu infraštruktúry a prosperite našej fakulty.

Cieľ práce

Hlavným cieľom tohto projektu v rámci bakalárskej práce je vytvorenie rezervačného systému učební, ktorého účelom je zjednodušenie a zlepšenie využívania priestorov školy s technickým vybavením. Keďže komplexnosť práce presahuje množstvo potrebné na pokrytie bakalárskej práce bol projekt rozdelený na back-endovú a front-endovú časť. Mojou úlohou je návrh a implementácia front-endovej časti, čiže zhotovenie funkčného a vizuálne prívetivého užívateľského rozhrania. Do mojej časti taktiež spadá testovanie použiteľnosti či analýza aplikácie, potrebná pre jej realizáciu.

Spojením týchto dvoch častí vznikne funkčný prototyp, ktorý bude študentom či zamestnancom školy umožňovať a hlavne časovo efektívne využívať

učebne ako i rozvrhárom lepšie koordinovať obsadenosť priestorov školy.

Pevne verím, že tento projekt bude veľkým prínosom alebo aspoň malým podnetom k zhotoveniu podobného či lepšieho nástroja, ktorý by určite nemal chýbať v portfóliu fakulty technického zamerania.

Štruktúra práce

V rámci prvej kapitoly nazrieme do podrobnej analýzy aplikácie, zakončenej návrhmi užívateľského rozhrania prostredníctvom wireframov. Druhá kapitola v sebe popisuje náhľady aplikácie a implementáciu spolu s časťami kódu pre lepšie pochopenie úvah. Posledná kapitola sa zaoberá testovaním.

Kapitola 1

Analýza a návrh

Čim väčší projekt, tým viac času má byť vyhradeného pre podrobný rozbor a pochopenie problému, ktorý sa bude riešiť. Analýza je základným kameňom každého seriózneho projektu a preto by sa k nej malo pristupovať pomaly, s rozvahou a dostatočným časovým rozostupom medzi jednotlivými časťami. Mnoho projektov stroskotalo časom z dôvodu zlej alebo nie príliš rozdpovedne navrhnutej štruktúry, ktorá zdanlivo vyzerala jednoducho. Podobne som to videl na našom projekte, ktorý pre mnohých predstavuje jednoduchosť obyčajnej webovej aplikácie, no my sme aj napriek tomu zhotovili rozsiahlu analýzu, čo sa nám napokon vyplatilo.

Do analýzy sme okrem modelovania návrhových diagramov zahrnuli aj rozbor použitých technológií čo nás postupne priviedlo až k zhotoveniu samotných wireframov, o ktoré som sa opieral pri implementácií front-endovej časti aplikácie.

Na analýze sa pracovalo spoločne a v prípade, že to v práci nie je uvedené, ide o mnou získané poznatky či vypracované sekcie. Nepoužité spoločné materiály sú uvedené v sekcii príloha B).

1.1 Analýza požiadaviek

Ako pri každej správnej analýze je dobré si na začiatku zhrnúť súbor požiadaviek ako funkčných, ktoré popisujú samotnú funkcionalitu a teda aké služby má systém poskytovať, kde kladú dôraz na možnosti ich realizácie, tak i nefunkčných, ktoré sú väčšinou technického typu. V nich sa rozoberá najmä akým spôsobom má systém služby poskytovať. Vo väčšine prípadov ide o podmienky či obmedzenia. Keďže sme pracovali v týme, tento súhrn požiadaviek vznikol ako výsledok spoločnej práce.

Funkčné požiadavky:

1. Zobrazenie miestnosti

2. Pridanie miestnosti
3. Odstránenie miestnosti
4. Blokovanie miestnosti
5. Vyhľadávanie (filter) miestností
6. Vytvorenie rezervácie
7. Zrušenie rezervácie
8. Úprava rezervácie
9. Schvaľovanie rezervácie
10. Blokovanie užívateľov

Nefunkčné požiadavky:

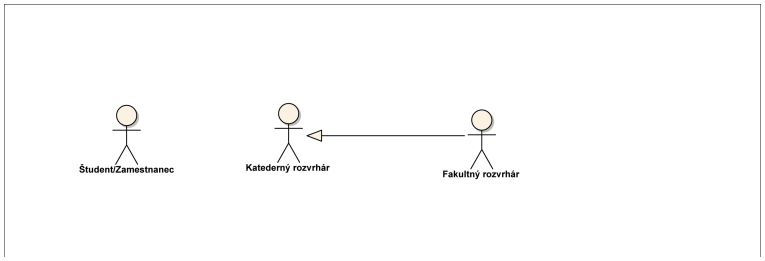
1. Dostupnosť na webe, systém je realizovaný ako webová služba
2. Platforma Java EE
3. Kompatibilita (zdroj dát) – Sirius, Google Calendar, KOSapi
4. Systém jednoducho rozšíriteľný pre ďalšie vstupné a výstupné zdroje
5. Užívateľsky prijateľné rozhranie

V nasledujúcich sekciách sa budem podrobne venovať typom a rozborom použitých diagramov, kde v sebe každý nesie iný pohľad na výslednú aplikáciu.

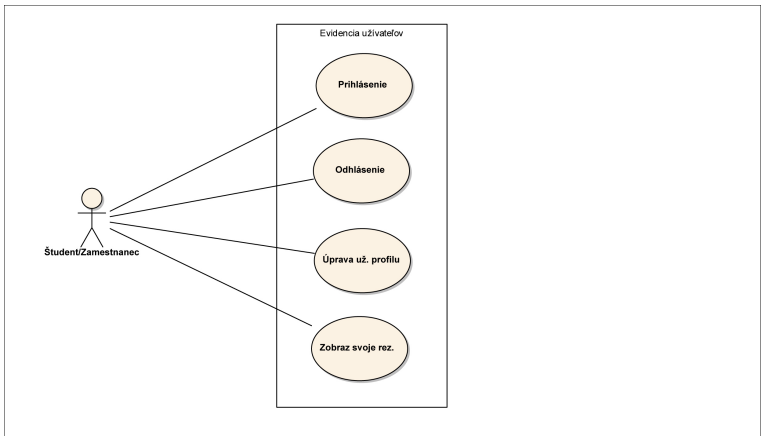
1.2 Use case diagramy

Use case diagram (digram použitia) zobrazuje správanie systému tak, ako ho vidí užívateľ. Účelom diagramu je opísať funkcionalitu systému, teda to čo od neho očakávame. Diagram hovorí o tom, čo má systém robiť, ale neopisuje ako to bude robiť. Preto je prvým druhom diagramov, ktorý sme v našej analýze použili. [1]

Prvý use case diagram nám zobrazuje vzťahy medzi užívateľmi systému. Najnižšiu rolu zastupuje študent/zamestnanec. Podrobnejší popis príslušných funkcionalít tejto role uvidíte na druhom diagrame. Fakultný – katederný rozvrhár je vo vzťahu dedenia, kde katederný rozvrhár dedí všetky vlastnosti a práva od fakultného rozvrhára. Žiadna ďalšia rola sa už v našom systéme nenachádza, aj keď sme mali v pláne ešte vytvoriť rolu administrátorskú, ktorá by mala na starosti práva či nastavania aplikácie. Zistili sme ale, že rozsah funkcionalít, ktorý by sme priradili tejto role by nebol postačujúci natoľko,



Obr. 1.1: Use case diagram rolí



Obr. 1.2: Use case diagram študent/zamestnanec

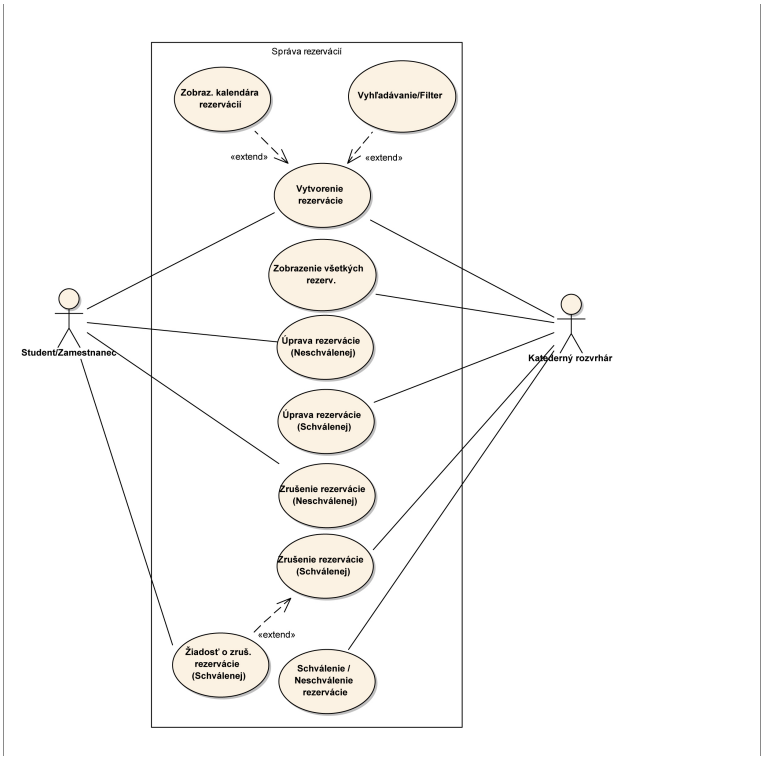
aby bola nutnosť zastúpenia individuálnou rolou. Z toho dôvodu sme tieto role spojili a preto fakultný rozvrhár je aj zároveň administrátorom správy aplikácie.

Následujúci diagram 1.2 ukazuje základnú funkcionalitu užívateľských rolí. Ktorýkoľvek užívateľ preto môže vykonávať bežné úkony ako sú napríklad zobrazenia či úpravy rezervácii.

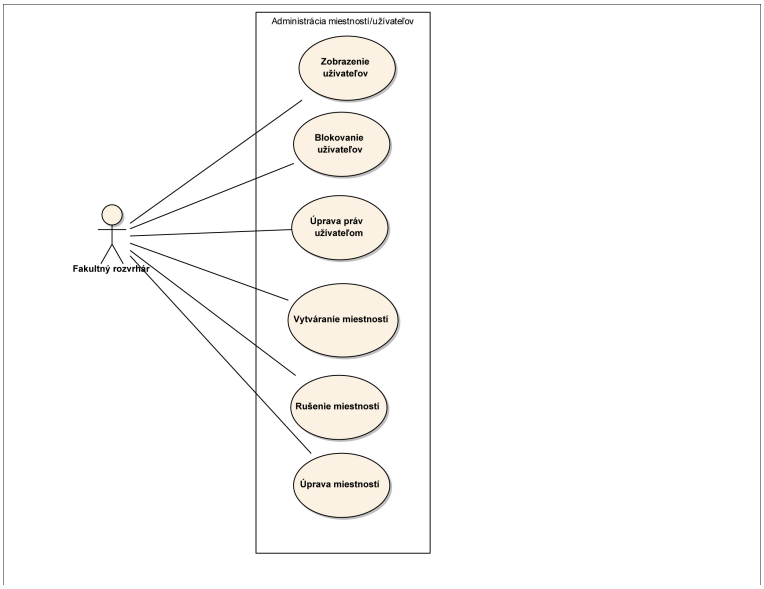
Tretí use case diagram, viď obrázok 1.3 nám dáva do porovnania rolu bez vyšších práv teda študent/zamestnanec a rolu fakultného rozvrhára, ktorý je v hierarchii rolí už v pozícii administračnej.

Z diagramu vyplýva, že študent/zamestnanec vykonáva iba podmnožinu úkonov dostupných pre rovrhárov akéhokoľvek typu, viď obrázok 1.3. Je dôležité poznamenať, že študent/zamestnanec môže svoje rezervácie upravovať len do chvíle, kedy nie je rezervácia schválená príslušným rozvrhárom.

Posledný diagram 1.4 zobrazuje funkcionalitu špecifickú len pre rozvrhára s najvyšším postavením v hierarchii rolí a to fakultného, čo môžete vidieť aj na obrázku 1.4. Tento diagram vychádza z prvého 1.1, kde už je znázornený vzťah dedičnosti medzi katederným a fakultným rozvrhárom, čo znamená, že



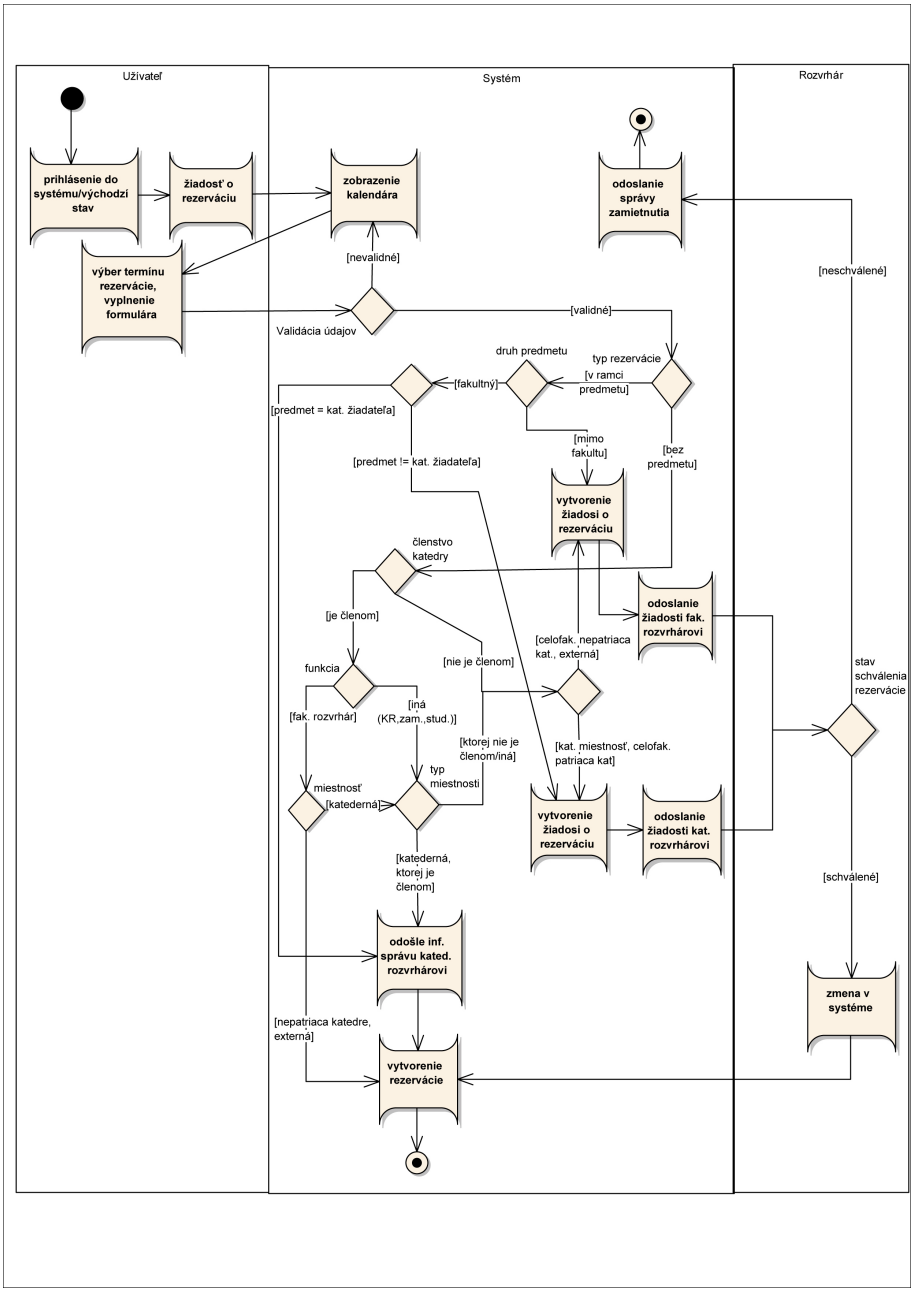
Obr. 1.3: Diagram vzťahu funkcionalít študent/zamestnanec – rozvrhár



Obr. 1.4: Use case diagram funkcionalít špecifických pre rolu fakultného rozvrhára

katederný rozvrhár má dostupné práva len na využívanie vlastností z predchádzajúcich dvoch diagramov.

1.3 Proces rezervácie



Obr. 1.5: Diagram aktivít procesu rezervácie

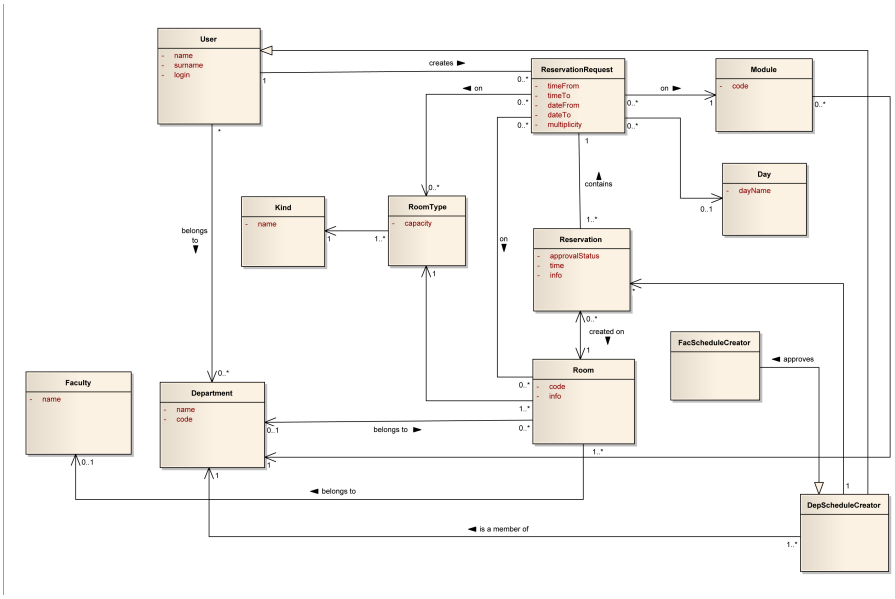
V našom prípade rezervácia prechádza zložitým procesom vytvárania. Tento proces pomocou diagramu aktivít. Mnou vytvorený diagram zobrazuje proces

rezervácie od vytvorenia až po uzavretie, čiže schválenie/neschválenie rozvrhárom.

Pozostáva z troch swimlines, pričom sa prostredná – Systém, vykonáva automaticky a teda zachycuje časť back-endu. Zložitosť diagramu je opodstatnená z dôvodu, že v sebe zahŕňa viacero druhov rezervácií. Je možné vytvoriť rezerváciu viazanú na predmet či miestnosť. V diagrame rozlišujem kto rezerváciu vytvoril a v akom je vzťahu k predmetu či miestnosti. Ako príklad nám poslúži situácia, kedy si študent vytvorí rezerváciu na miestnosť A:1024, ktorá spadá pod katedru SI a teda požiadavka o schválenie sa odošle príslušnému katedernému rozvrhárovi. Pri vytvorení rovnakej rezervácie ale katedernému rozvhárovi z katedry SI sa už žiadosť zbytočne nevytvára, ale je vytvorená rovno samotná rezervácia. Zvyšné dve swimlines (užívateľ a rozvrhár) v sebe nesú proces, ktorý je už realizovaný manuálne užívateľmi príslušných rolí.

1.4 Návrh kľúčových entít

Zachytiť kľúčové pojmy doménovej oblasti je najzásadnejšou časťou tohto modelu. Doménový model identifikuje vzťahy medzi jednotlivými objektmi problémovej domény a niekedy tiež ich atribúty. [2] V prípade, že je doménový model navrhnutý zle, odzrkadlí sa to v celej logike aplikácie, preto je dôležité zvažovať každý vzťah, atribut či potrebu vytvorenia danej entity. [3]



Obr. 1.6: Doménový model zobrazujúci entity použité v aplikácii

V doménovom modeli zobrazujem entity využívané v našom systéme. Celý

model je potrebné vnímať objektovo, pretože každá entita má samostatné zastúpenie v našej aplikácii.

Prvotný návrh pozostával iba zo siedmych entít no časom sa ukázalo, že bude výhodné model rozšíriť, z dôvodu zabezpečenia nezávislosti v prípade zmien.

Jednou z diskutabilných častí bola entita `Type`, ktorú sme napokon rozdelili a pridružili jej entitu `Kind` a tá v sebe nesie názov typu miestnosti (počítačová, prednášková, atď.). Zmenou sme dosiahli, že je možné kedykoľvek pridať nový typ miestnosti a logika aplikácie sa nezmení. Pôvodná úvaha bola entitu `Kind` uviesť ako datový typ `enum`, no z dôvodu možného vytvorenia či kombinovania rôznych druhov učební by to nebolo optimalizované.

Ostatné entity ako napríklad `Day`, už v sebe nesú atributy typu `enum`, čo je v tomto prípade `DayName`. Keďže tento typ v sebe nesie všetky možné stavy (dni), vytvorenie novej entity by tomto prípade bolo zbytočné. Vzťah dedičnosti `User`, `DepScheduleCreator` a `FacScheduleCreator` je znázornený na základe vyššie vytvorených use case diagramov, kde `DepScheduleCreator` podedí časť vlastností `FacScheduleCreator` a následne `User` ešte ďalšiu podmnožinu vlastností entity `DepScheduleCreator`.

1.5 Sekvenčné diagramy

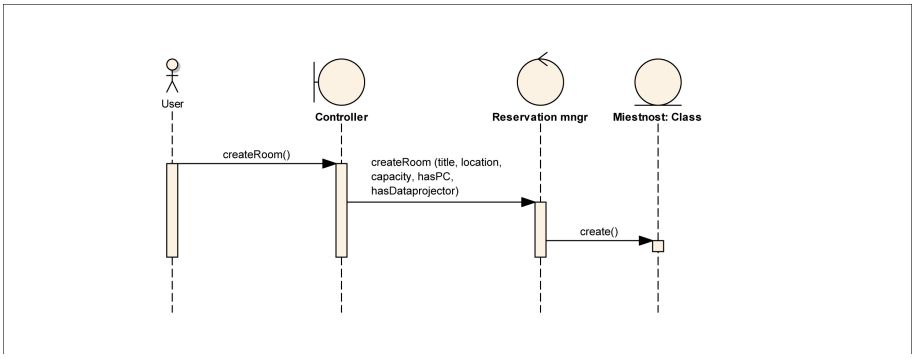
V sekvenčných diagramoch je znázornená sekvencia procesu požiadavky používateľa systému, cez jeho spracovanie až po odpoveď systému. Pre nás tieto diagramy znamenali prehľad akcií, ktoré spadajú pod front-endovú a back-endovú časť. Ich dôležitosti znamenala striktné vymedzenie požiadaviek, ktoré ako front-end budem potrebovať od back-endu. Na základe týchto diagramov sa začalo vytvárať rozhranie *REST API*, ktoré slúži pre komunikáciu medzi naším front-endom a back-endom, kde front-endu poskytuje sadu metód umožňujúcih bezproblémové zobrazovanie potrebných dát.

V prvom sekvenčnom diagrame ilustrujem proces vytvorenia miestnosti, ktorý pozostáva z požiadavku o vytvorenie, v ktorom parametricky zadávam údaje v tvare objektu Room, viď obrázok 1.7.

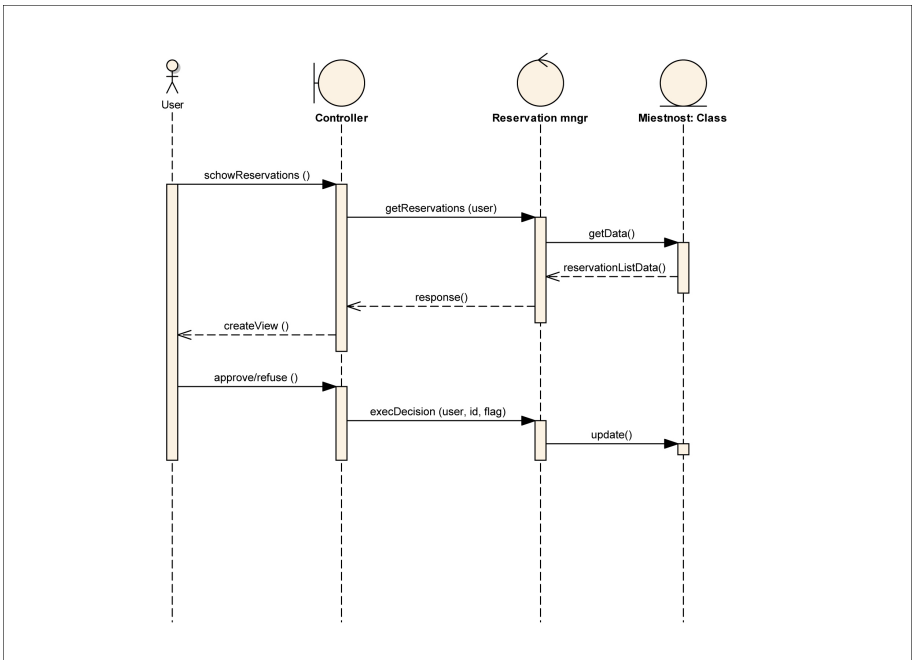
Ako prvá sa vykonáva sekvencia zobrazenia zoznamu rezervácií, ktorá je pre nasledujúce tri diagramy identická a až nad získaným súborom objektov sa vykonáva požiadavka, ktorá zoznam upravuje v každom diagrame odlišne.

Tento diagram (1.8) znázorňuje funkcionalitu katederného či Fakultného rozvrhára a to schvaľovanie rezervácie, kde sa nad získaným zoznamom vykoná príkaz zmeny stavu rezervácie.

```
pending -> approved
pending -> refused
```

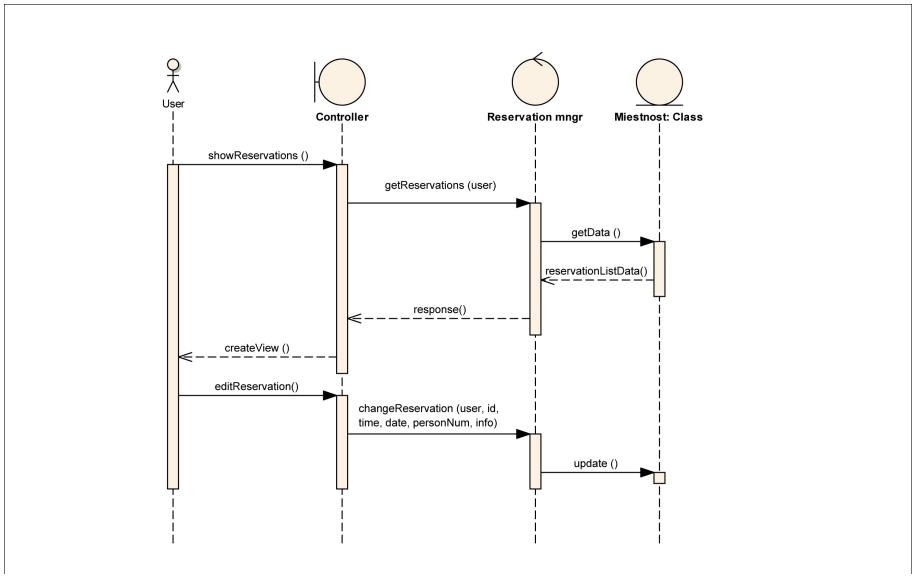


Obr. 1.7: Sekvenčný diagram vytvorenia miestností



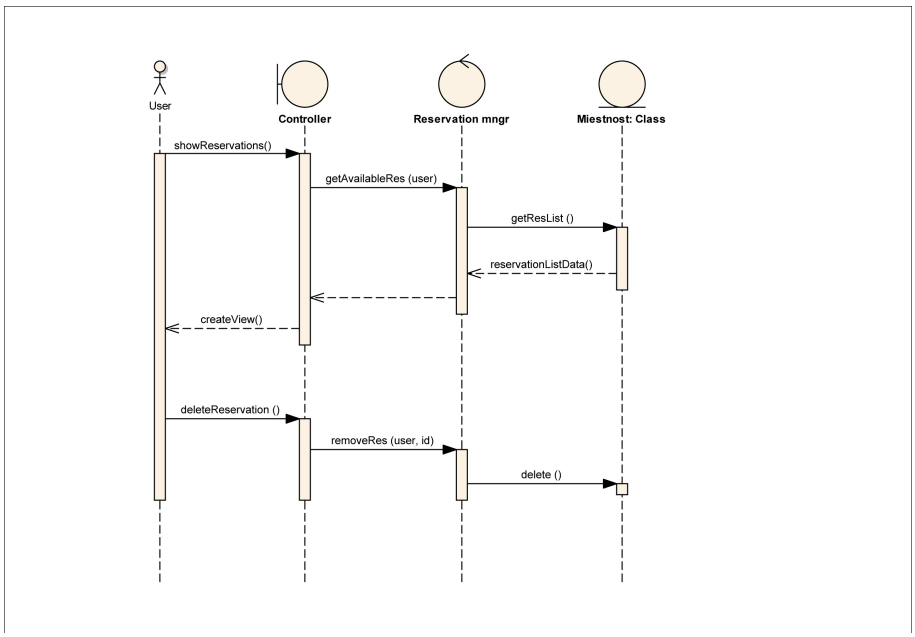
Obr. 1.8: Sekvenčný diagram schválenia/neschválenia rezervácií

V ďalšom diagrame, zobrazujem úpravu rezervácie a odosielanie požiadavky na systém v podobe parametrického volania metódy umožňujúcej úpravu a následné uloženie zmien ako môžete vidieť na obrázku 1.9.



Obr. 1.9: Sekvenčný diagram úpravy rezervácie

Posledný diagram ilustruje zrušenie rezervácie ako študentom/zamestnancom, za predpokladu stavu „čakania na schválenie“, tak rozvrhárom za akýchkoľvek okolností.



Obr. 1.10: Sekvenčný diagram zrušenia rezervácie

1.6 Nástroje a technológie použité na realizáciu

Rozhodoval som sa medzi frameworkami *Spring MVC* a *JSF*, kde som bral do úvahy aspekty ako sú jednoduchosť použitia či dokumentácia a komunita. Keďže moja práca pozostáva z návrhu front-endovej časti aplikácie nemohol som vynechať ani stránku grafickej úrovne takže UX, designu a celkového feelingu.

V prvom rade som bral do úvahy jednoduchosť a rýchlosť osvojenia si základov frameworku na úrovni použiteľnej pre moju bakalársku prácu. Z nájdených zdrojov som vytiahol dôležité informácie zostavil krátke porovnanie spomínaných frameworkov.

1.6.1 Jednoduchosť použitia

Spring je veľmi rozsiahly a v dôsledku toho nie je možné jeho využívanie typom „jeden deň vyskúšam a druhý už nasadzujem“. Aby bolo možné využiť celkový potenciál frameworku, je potrebné vedieť prácu *Springu* ako celku, čo je jeho najväčšia nevýhoda. Existuje veľa onlineových učebných programov a dokumentácií, ako aj ukážkové aplikácie, ktoré sa dajú využiť ako zdroj, no *Spring* je vhodný pre budovanie serióznych aplikácií s pevnými základmi. Poskytuje bohaté užívateľské rozhranie a pohodové API, ktoré by bola veľká škoda zjednodušovať.

JSF sa snaží poskytnúť ľahko použiteľný framework pre vytváranie opakovane použiteľných webových komponentov do väčších Java aplikácii bez potreby zdĺhavého učenia. Je veľmi jednoduché sa s *JSF* dostať na úroveň fungujúceho programu a často nevyžaduje špeciálnu konfiguráciu keďže kód je dodávaný v každom Java EE kompatibilnom aplikačnom serveri.

JSF sa môže zdať trochu mätúce, kde sa občas aj ohromujúce a neuveriteľne užitočné funkcie môžu zdať popletené kvôli slabej dokumentácii. Dokumentácia je často špecifická pre určité prispôsobené prostredie a zdĺhavé hľadanie vecí, ktoré potrebujem, často vedie k frustráci. [4]

1.6.2 Dokumentácia a komunita

Aby sa dobrý framework mohol stále zlepšovať, najmä pokiaľ nie je finančne podporovaný, je veľmi dôležitá vynikajúca dokumentácia. Užitočná najmä preto, aby noví užívatelia rýchlo získali požadované informácie a dostali sa na úroveň použiteľnosti za čo najkratšiu dobu.

Spring ako najpoužívanejší framework Java vývojárov má rozhodne svoje výhody. Samotná stránka frameworku sama o sebe poskytuje celú radu výukových programov a to buď formou videa alebo písaných tutoriálov. Nájdu sa tam ako začiatočníci tak aj pokročilí vývojári. *Spring* má taktiež okolo seba veľmi silnú komunitu, ktorá neustále zlepšuje funkcionalitu či pokrytie frameworku. Opravujú bugy, pridávajú rôznorodé featury či upravujú už vstavanú

funkcionalitu. Ak užívateľ potrebuje vyhľadať odpoveď na otázku či problém, má k dispozícii fórum, o ktoré sa znova stará široká komunita.

Na druhú stranu *JSF* nie až tak rozšírený no je plne podporovaný s referenčnou implementáciou od spoločnosti *Oracle*. Narozdiel od *Springu* tu majú dokumentáciu na starosti platení zamestnanci Oraclu, ktorí sa o ňu starajú a obohacujú ju o množstvo vzorov či príkladov. Bohužiaľ, väčšina Oraclovskej dokumentácie je postavená na referenčnej implementácií Oracle Java EE nástrojov. Čo znamená, že ak ste používateľom NetBeansu a *GlassFish*, čo sa týka aj mojej situácie, ste nútení hľadať zdroje dokumentácie tretích strán. Čo ale nevidím ako prekážku, pretože nie som jediný, kto si vybral túto cestu a keďže problémy, ktoré riešim sú bežné implementačné, nie je problém nájsť ich riešenie jednoducho prostredníctvom vyhľadávača alebo formou video tutoriálov na *YouTube*, ktoré sú mi najbližšie, pretože človek rovno vidí implementáciu a riešenie problémov v reálnom čase s komentovaním príslušného lektora. [4]

1.6.3 UX, vzhľad a feel

Posledným a pre front-endovú časť najdôležitejším ukazateľom je práve to, ako bude výsledná aplikácia vyzerať, ako ju bude výsledný užívateľ vnímať. V tejto sekcii preto porovnávam náročnosť tvorby UI týchto dvoch frameworkov. *Spring* má veľmi bohatú sadu funkcií, nachádzajúcu sa na strane servera, ktoré sú udržiavané a ich kód sa neustále rozvíja. No aj napriek tomu jeho rozšírenia neponúkajú žiadny bohatý framework pre budovanie zaujímavých rozhraní. Dáva ale stále možnosť písať solídny back-end a používať iný framework pre budovanie UI, čo by bola ale škoda, ak je možné mať všetko v jednom.

Tu sa dostávame práve k *JSF* ktorý je skvelý pre vývojárske tými či jednotlivcov, ktorí chcú vytvoriť plnohodnotné užívateľské rozhranie bez toho, aby sa museli stať majstrami JavaScriptu. Existuje veľa skvelých prvkov už v samotnom katalógu, kde sa dajú nájsť tiež fantastické doplnky k *JSF*. Pre príklad slúžia prídavky ako napríklad *IceFaces*, *PrimeFaces* či *RichFaces*. S použitím týchto pokročilých komponentov sa webové aplikácie vytvorené pomocou *JSF* stanú plnohodnotnými náhradníkmi za svojich stolových náprotivkov.

Po týchto úvahách som sa rozhodol pre použitie frameworku *JSF*, kde mala veľkú váhu jednoduchosť osvojenia si princípov a funkcií, na úroveň požadovanú pre vytvorenie aplikácie rozsahom mojej bakalárskej práce. Pri dokumentácii som bol spokojný pri oboch frameworkoch skoro rovnako, kde pri *JSF*, braného ako pomerne rozšírený framework, som očakával väčšiu oficiálnu podporu aj pre iné IDE či aplikačné serveri. Najviac u mňa ale rozhodla tá časť, ktorou sa budem prezentovať, čiže celkový design, UX a vlastne koncový výsledok, aký bude možné s frameworkom dosiahnuť v dostupnom čase a s rozumným množstvom úsilia. Tu som sa jednoznačne utvrdil v rozhodnutiu,

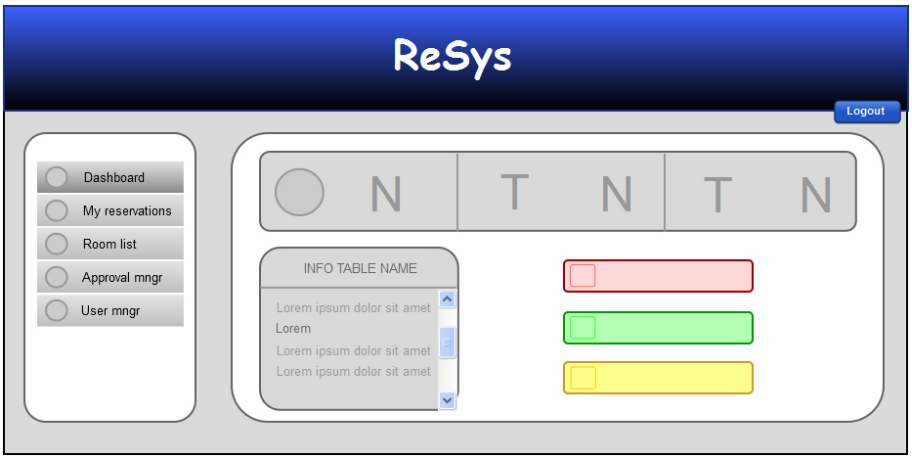
že použitie frameworku *JSF*, pre túto bakalársku prácu bude správnou cestou. [4]

1.7 Návrh GUI

Sekcia návrhu pozostáva z prevažne tvorby wireframov, čo sú pracovné skice, náhľady výslednej aplikácie. V prípade wireframov nejde o finálnu podobu designu, ale len o kostru (wireframe – drôtený model) grafického rozhrania. Tieto skice prevažne pozostávajú zo šablonovitého rozmiestnenia prvkov, ktoré sa vo výslednej implementácii môžu objaviť. Prvky sú reprezentované jednoduchými geometrickými útvarmi a kulisa predstavivosti je dotváraná textovou formou, zriedkavo i farebne. [5]

Je množstvo nástrojov či metód slúžiacich ako prototypovacie nástroje. V mojom prípade som sa rozhodol využiť nástroj na tento účel stvorený. Ide o jednoduchý prototypovací nástroj Pencil, ktorý v sebe síce nezahŕňa širokú škálu šablón či objektov, no pre môj návrh postačujúcu.

Prvý wireframe nesie predbežný návrh dashboardu, ktorý pozostáva z troch častí, Za predpokladu, že neberiem do úvahy časť headeru (nadpis, odhlasovanie) a postrannú navigáciu (menu).



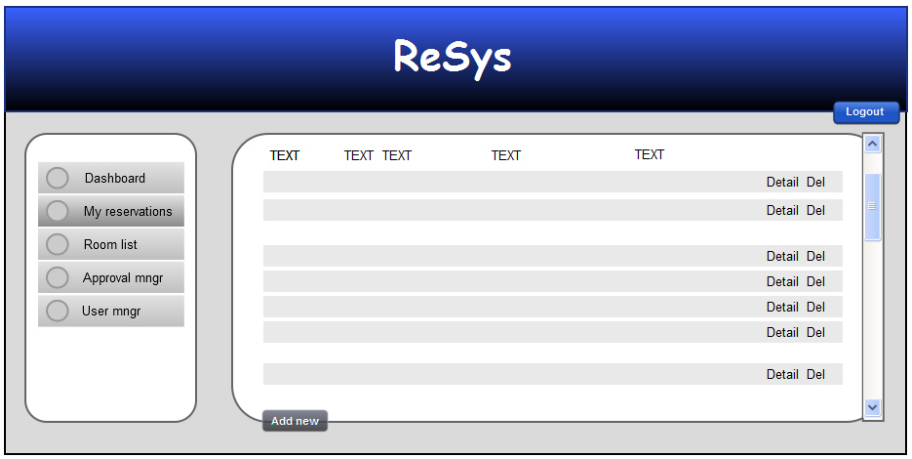
Obr. 1.11: Návrh dashboardu aplikácie

Prvá časť bude zobrazovať základné údaje o počte schválených/neschválených či všetkých uskutočnených rezerváciách. Ľavý spodný panel obsahuje log informácií a v pravej časti budú obsiahnuté odkazy na novo schválené, neschválené alebo čakajúce rezervácie.

Dashboard má slúžiť k rýchlej informovanosti užívateľa a mal by obsahovať všetky podstatné údaje, aby bol užívateľ odbremenený od zdĺhavého vyhľadávania požadovaných informácií. Z toho dôvodu tam nie sú obsiahnuté žiadne

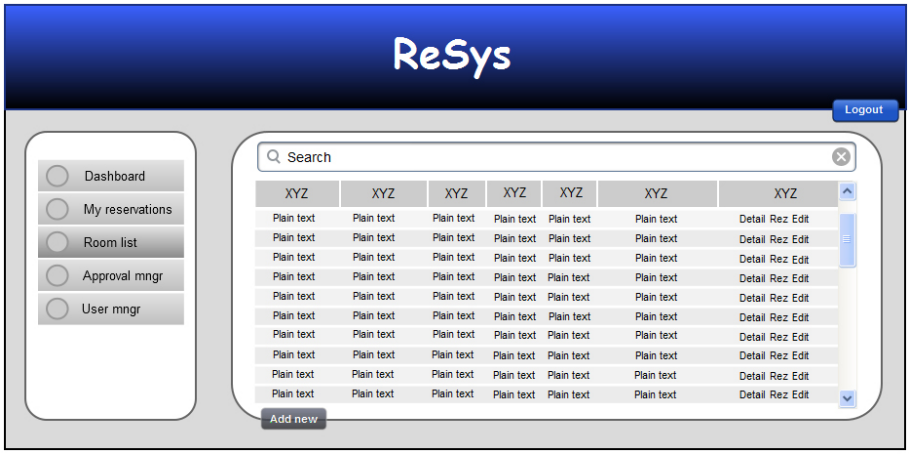
odkazy súvisiace s administratívovou či už na užívateľskej alebo rozvrhárskej úrovni. Tieto časti sú obsiahnuté v ostatných častiach návrhu.

Nasledujúci wireframe v sebe nesie zoznam rezervácií jedinečný pre každého prihláseného užívateľa zvlášť



Obr. 1.12: Zoznam mojich rezervácií

Wireframe s týmto zoznamom obsahuje prehľad o všetkých miestnostiach zaznamenaných v systéme. Ako je na wireframe načrtnuté, tlačidla detail, edit, rez priamo odkazujú na wireframy, ktoré boli pre danú funkcionalitu implementované. Tlačidlo edit je dostupné iba užívateľovi s administrátorskymi právami.

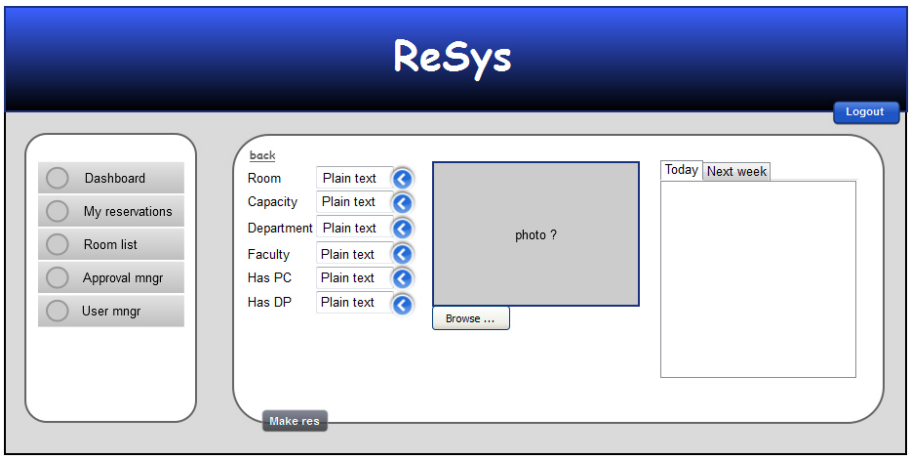


Obr. 1.13: Zoznam miestností v systéme

Nasledujúci wireframe ilustruje pohľad pri zobrazení detailu miestnosti.

Prvotný návrh bol myslený so zobrazením základných údajov s fotografiou lepšej predstavy interiéru miestnosti. Po dohode sme ale dospeli k záveru, že je táto informácia zbytočná. Z toho dôvodu sa vo výslednej implementácii nenachádza spolu s ďalšími prebytočnými údajmi ako sú:

- Has DP (či miestnosť obsahuje dataprojektor
- Has PC (tento údaj bol zmenený na typ miestnosti počítačová, prednášková, atď.
- Faculty (príslušná fakulta miestnosti

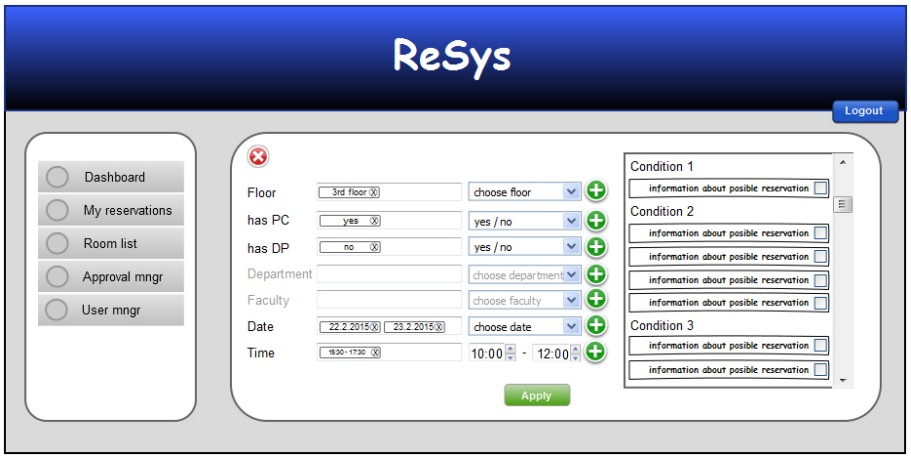


Obr. 1.14: Detail miestnosti

Tento wireframe už priamo súvisí s vytváraním rezervácie a to prostredníctvom ponúkaného rozhrania s kalendárom. Väčšina už existujúcich aplikácií obsahuje práve túto formu vytvárania udalostí, takže väčšina užívateľov už pozná princípy a nedochádza tak zdĺhavému skúmaniu funkcionality jednotlivých prvkov.

Obr. 1.15: Návrh vytvárania rezervácia prostredníctvom kalendára

V prípade vytvárania zložitejších rezervácii, ktoré v sebe zahŕňajú rôzne opakovania prípadne filtre, je z pohľadu kalendára možné presmerovanie na rozšírený formulár prostredníctvom tlačidla „Ext res“. Toto zobrazenie pozostáva z dvoch častí a to formulára a zoznamu vyhľadaných možností, z pomedzi ktorých si užívateľ vyberá najvhodnejšie z nich. Proces prebieha tak, že užívateľ po vyplnení formulára selekciu aplikuje, systém vyhľadá všetky možné a vyhovujúce kombinácie ktoré sa následne zobrazia v pravej časti, odkiaľ si užívateľ ešte volí, ktorá z nich mu vyhovuje najviac. Je dobré poznamenať, že pravá časť už zobrazuje možnosti presne zodpovedajúce kritériám vyhľadávania, takže ktorúkoľvek z možností si užívateľ vyberie už je ňou pokrytá jeho požiadavka.



Obr. 1.16: Návrh rozšírenej ponuky pri vytváraní komplexnejších rezervácií

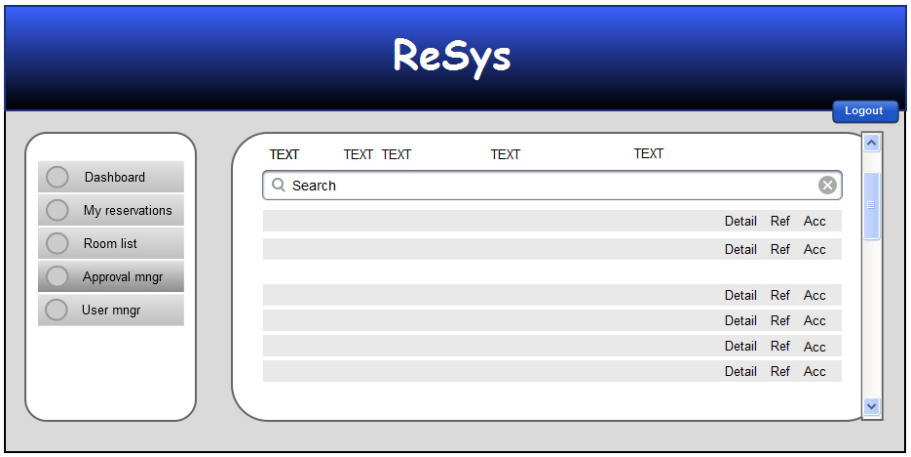
Detail rezervácie si môže zobraziť ktorýkoľvek prihlásený užívateľ. Líšia sa opäť len v tom, o akého užívateľa s akými právami ide. V prípade, že ide o rozvrhára, detail je obohatený o tlačidlá Approve (schváliť), Refuse (neschváliť). Tlačítko edit je zobrazené užívateľov role študent/zamestnanec iba v prípade, že ide doposiaľ o neuzavretý stav a to *pending* (čakajúci na schválenie).

Obr. 1.17: Detail rezervácie

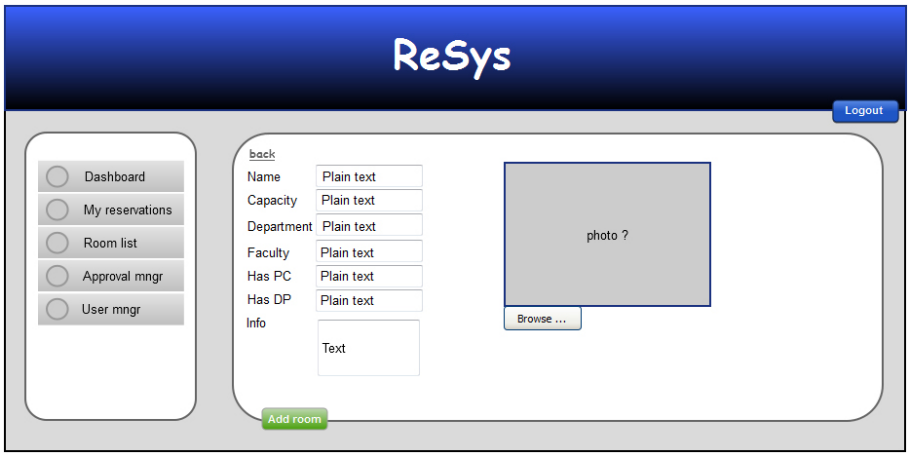
Keďže tento wireframe znázorňuje zoznam všetkých rezervácií vykonaných ktorýmkoľvek užívateľom je pochopiteľné, že ide o funkcionalitu určenú na zobrazenie iba užívateľom s rolou rozvhrár. V orípade katederného je vyobrazený iba zoznam rezervácií miestností spadajúcich pod jeho katedru, no fakultný rozbvrhár má dostupný úplný zoznam rezervácií, keďže v systéme neberieme do úvahy použiteľnosť rozvrhármi iných fakúlt a teda nutnosť implementácie ďalšieho filtra. (Obrázok 1.18).

Tlačidlá nachádzajúce sa v pravej časti slúžia k rýchlejšiemu spravovaniu vytvorených rezervácií. Uľahčuje sa tým práca rozvrhára a „šetria kliky“, kde rozvrhár nie je nútený zobrazovať každý detail zvlášť a následne až v tomto pohľade rozhodovať o tom, či rezerváciu schváli alebo nie.

Wireframe s pohľadom a funkcionalitou určenou iba pre fakultného rozbrhára. Ide o pridávanie miestností do systému jednoduchým vyplnením formulára. Po menšej diskusii sa ale z výslednej implementácie vypustil úplne z dôvodu využitia datového zdroja KOSapi, ktorý nám poskytol zoznam miestností s tým, že je pevne daný. V prípade pridávania miestnosti alebo rozširovania priestorov fakulty sa priestor zaeviduje v KOSe takže sa nám týmto rovno rozšíri aktuálny zoznam. (Obrázok 1.19)



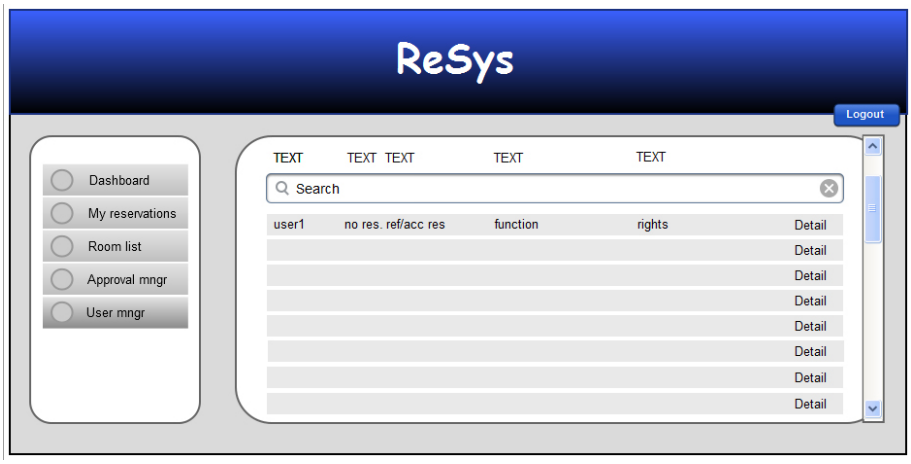
Obr. 1.18: Schvaľovací manažér



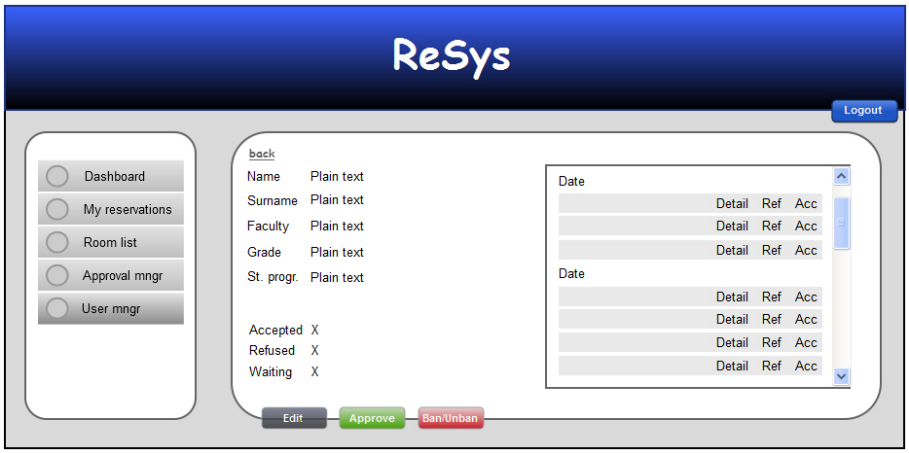
Obr. 1.19: Pridávanie miestnosti

Zoznam užívateľov, ktorý môžete vidieť navrhnutý na obrázku 1.20 je získaný z datového zdroja KOSapi. Funkcionalita určená iba fakultnému rozvrhárovi.

Wireframe naväzuje na ten prechádzajúci, kde sú bližšie špecifikované informácia o ktoromkoľvek užívateľovi systému. Tlačidlá spodnej lišty, viď obrázok 1.21, slúžia k schváleniu vybranej rezervácie pravého panelu či zamedzeniu akejkoľvek činnosti užívateľa v systéme – BAN. Obrázok 1.18



Obr. 1.20: Zoznam užívateľov aplikácie



Obr. 1.21: Detail užívateľa

Kapitola 2

Realizácia

2.1 Návrh datového formátu pre front-end a back-end

Návrh bude realizovaný prostredníctvom *JAXB* schémy, ktorá využíva jazyk *XML* na mapovanie, resp. prepis tried. Následný kód vygeneruje celú štruktúru objektov ktorá tvorí programový základ našej aplikácie.

Z dôvodu, že je vytvorený *XSD* dokument rozsiahly, rozhodol som sa použiť iba vybrané časti kódu. Prepis doménového návrhu do *XML* kódu nie je jednoduchý a to práve preto, že niektoré časti entit, vzťahov medzi nimi či ich atributov, musia byť vnímané z iného hľadiska, a páve týmito časťami sa zaoberám v nasledujúcej sekcii.

Kód v tejto časti nám popisuje vzťah užívateľa, katederného a fakultného rozvrhára. V doménovom modeli je tento vzťah znázornený ako dedičnosť, čo z rolí logicky vyplýva, no v *XML* je riešením referencovať od hierarchicky najnižšie položenej entity `User` až po tú najvyššiu `DepScheduleCreator` a `FacScheduleCreator`. Teda v našom prípade entity typu katederný či fakultný rozvrhár sú zároveň vždy entita typu `User`, no toto tvrdenie nie je platné obojstranne.

Listing 2.1: Vzťah dedičnosti role Rozvrhár a Používateľ

```
<xsd:element name="user" type="user"/>
    <xsd:complexType name="user">
        <xsd:sequence>
            <xsd:element ref="department" minOccurs="0" maxOccurs="
                unbounded"/>
            <xsd:element ref="reservationRequest" minOccurs="0"
                maxOccurs="unbounded"/>
            <xsd:element name="name" type="xsd:string" />
            <xsd:element name="surname" type="xsd:string" />
            <xsd:element name="login" type="xsd:string" />
        </xsd:sequence>
    </xsd:complexType>
```

```
<xsd:element name="facScheduleCreator" type="facScheduleCreator"/>
    <xsd:complexType name="facScheduleCreator">
        <xsd:sequence>
            <xsd:element ref="depScheduleCreator" />
            <xsd:element ref="user" />
        </xsd:sequence>
    </xsd:complexType>

    <xsd:element name="depScheduleCreator" type="depScheduleCreator"/>
    <xsd:complexType name="depScheduleCreator">
        <xsd:sequence>
            <xsd:element ref="department" />
            <xsd:element ref="reservation" minOccurs="0" maxOccurs="
                unbounded"/>
            <xsd:element ref="user" />
        </xsd:sequence>
    </xsd:complexType>
```

Táto ukážka dáva do pozornosti entitu `Day`, ktorá obsahuje atribut `dayName`. Na tomto príklade je ale zaujímavé to, že nejde o bežný typ, ale konkrétne je atribut `dayName` typu `enum`. Znamená to, že obsahuje vopred nadefinované možnosti (dni v týždni), resp. stavy ktoré môže nadobudnúť a každý iný stav, nepatriaci do tejto množiny hlási chybu.

Listing 2.2: Ukážka typu enum atributu entity Day

```
<xsd:element name="day" type="day"/>
    <xsd:complexType name="day">
        <xsd:sequence>
            <xsd:element name="dayName" type="daysInWeek" />
        </xsd:sequence>
    </xsd:complexType>

    <xsd:simpleType name="daysInWeek">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="monday"/>
            <xsd:enumeration value="tuesday"/>
            <xsd:enumeration value="wednesday"/>
            <xsd:enumeration value="thursday"/>
            <xsd:enumeration value="friday"/>
            <xsd:enumeration value="saturday"/>
            <xsd:enumeration value="sunday"/>
        </xsd:restriction>
    </xsd:simpleType>
```

V poslednej časti zverejneného *XML* kódu poukazujem ta entitu `Reservation`, ktorá obsahuje tak, ako predchádzajúci príklad atribut typu `enum`. V tomto prípade ide ale o druhy stavov, v ktorých sa rezervácia môže nachádzať.

Listing 2.3: Typy stavov ktoré môže rezervácia nadobudnúť

```
<xsd:element name="reservation" type="reservation"/>
    <xsd:complexType name="reservation">
        <xsd:sequence>
            <xsd:element name="approvalStatus" type="status" />
            <xsd:element name="time" type="xsd:date" />
```

```
            <xsd:element name="info" type="xsd:string" />
            <xsd:element ref="reservationRequest" minOccurs="1"
                maxOccurs="1"/>
            <xsd:element ref="room" />
        </xsd:sequence>
    </xsd:complexType>

    <xsd:simpleType name="status">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="approved"/>
            <xsd:enumeration value="refused"/>
            <xsd:enumeration value="pending"/>
        </xsd:restriction>
    </xsd:simpleType>
```

2.2 Implmentácia GUI

Po vygenerovaní kostry tried a teda postavení základného stavebného kameňa pre back-endovú časť, zostavení wireframov a rozhodnutí o použitej technológií nastáva vhodná chvíľa k implementácii front-endovej časti.

2.2.1 Bootstrap framework

Pri tvorbe projektov, ktorých výstupy sa zobrazujú vo webovom prehliadači počítača alebo na mobilných zariadeniach často riešime problém adaptívneho či responzívneho designu. Často sa stretávame aj s problémom rýchleho návrhu alebo jednoduchého rozmiestnenia komponentov ako sú navigácia, galéria, rôzne zoznamy či tabuľky. Mnoho developerov trávilo hodiny pri dobrom návrhu a následnom kódení šablón, aby dosiahli aspoň čiastočne uspokojivého výsledku zobrazenia dát. Postupom času ale vznikol nástroj, ktorý tieto možnosti ponúka. Pre lepšiu predstavivosť ide o prostriedky ako napríklad návrh jednoduchého layoutu, navigácií, tabov až po predefinované štýly tlačidiel na úrovni jednoduchého systematického priradzovania css tried elementom. Z tohoto hľadiska dáva kodérom možnosť sústrediť sa veci dôležitejšie a teda nestrácať zbytočne čas s implementáciou prvkov, ktoré už dávno existujú a sú pomerne frekventovane používané. Reč je o css frameworku *Bootstrap*.

V mojom prípade som *Bootstrap* využíval k rozvrhnutiu jednotlivých prvkov do layoutu, ktorý pozostáva z headeru, navbaru a contentu ako môžete vidieť na priložených screenshotoch výslednej aplikácie. *Bootstrap* mi pomohol aj s rýchlym návrhom tabuliek, ktoré v sebe niesli zoznamy rezervácií, miestností či užívateľov. Návrh akčných tlačidiel bol taktiež ponechaný defaultnému boostrapovému štýlovaniu s menšou úpravou odsadení a veľkostí.

Tento framework ponúka skutočne obrovské možnosti využitia, no pre tento projekt som využil iba malú podmnožinu z nich.

2.2.2 Java PrimeFaces

Tento názov nesie komponenta vyvinutá pre platformu Java, konkrétne framework *Java PrimeFaces*, teda na ktorom je postavená celá naša aplikácia.

Podobne ako boostrap poskytuje množstvo už predefinovaných prvkov, ktoré stačí jednoducho použít, upraviť kód a naplniť datami z back-endu. Je dostupný formou open source knižnice, takže developer má kompletný prístup k zdrojovému kódu používaných prvkov. Najväčším benefitom a pre mňa aj hlavným dôvodom použitia tejto knižnica bola už implementovaná funkcia Schedule teda kalendár, s funkcionalitou postačujúcou pre základné využívanie a testovanie. Ďalším prvkom, ktorý mi zjednodušil a urýchlil vývoj bol zoznam s integrovanou funkcionalitou stránkovania dát. Postačovalo odstrániť nepotrebné časti kódu, vytvoriť si jednoduchú servisnú triedu na obsluhu dát a o všetko ostatné sa o nás postarala už existujúca implementácia prvku. Veľmi nápomocné boli aj formulárové prvky, ktoré pracovali na báze javascriptu, kde sa kliknutím na požadovaný element zobrazilo okno umožňujúce úpravu príslušných údajov a ich následné uloženie.

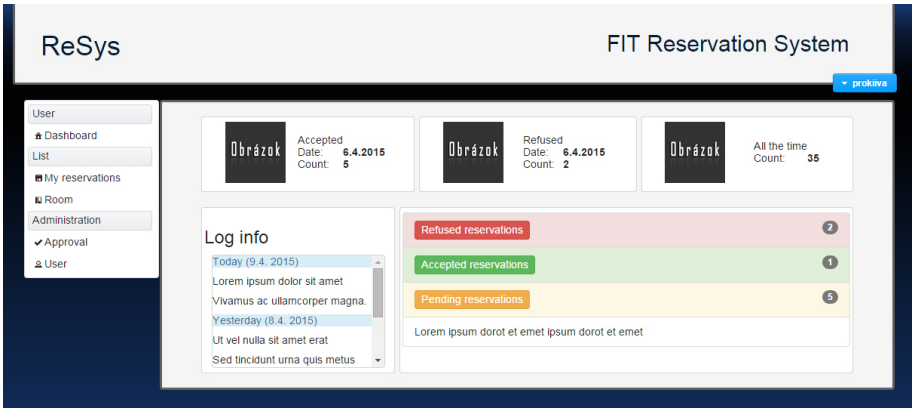
2.2.3 Náhľady výsledkov implementácie

S využitím nástrojov opísaných v predchádzajúcich sekciách som vytvoril funkčný prototyp fron-endovej časti aplikácie. Na následných obrázkoch prezentujem výsledok, ktorý bol dosiahnutý za pomoci frameworku *JSF* využívajúci komponentu *PrimeFaces* a css frameworku *Bootstrap*.

Nasledujúci dashboard sa zobrazí ihneď po úspešnom prihlásení užívateľa či uz v role študent/zamestnanec alebo rozvrhár. Je dôležité si uvedomiť že rozvrhár je tiež užívateľ a je preto vhodné, aby bol informovaný o všetkých svojich rezerváciach ako ktorýkoľvek iný užívateľ.

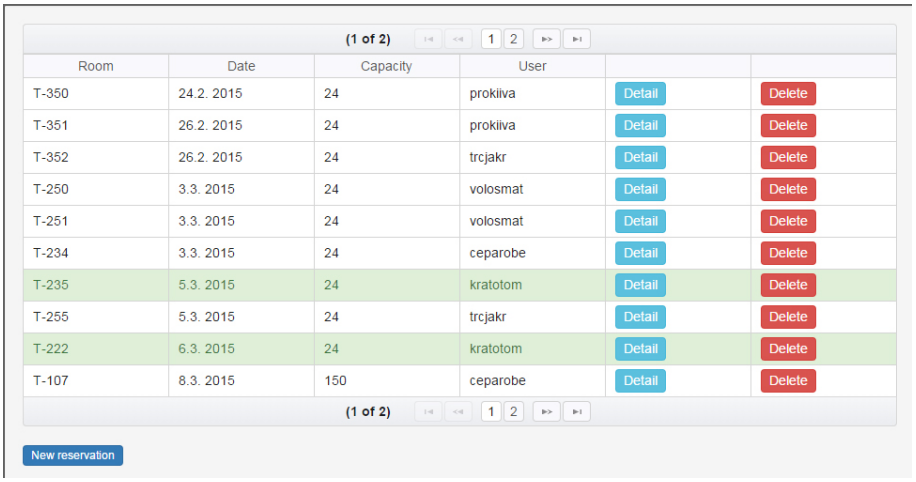
Dashboard panel obsahuje všetky dôležité údaje/data, ktoré užívateľ potrebuje k rýchlemu informovaniu sa o stave rezervácií. Panel pozostáva z troch častí. Horný riadok obsahuje údaje o schválených a neschválených rezerváciách za poslednú dobu či informáciu o celkovom počte všetkých rezervácií vytvorených týmto užívateľským účtom.

Spodná ľavá časť je log, v ktorom sú textovo zobrazené informácie či odkazy od rovrhárov. Účelom spodnej pravej časti je informovať o novo schválených/neschválených (v prípade existencie je button väčší a farebný kontrast poľa vyšší) či čakajúcich rezerváciách.



Obr. 2.1: Finálna podoba dashboardu užívateľa

Príslušný wireframe zobrazuje zoznam rezervácií z pohľadu rozvrhára, kde sú jednotlivé rezervácie farebne odlišované. Zelenou farbou sú vyznačené rezervácie schválené, červenou neschválené a žltá farba vyznačuje rezervácie, o ktorých schválení doposiaľ nebolo rozhodnuté. Sú v stave *pending*.

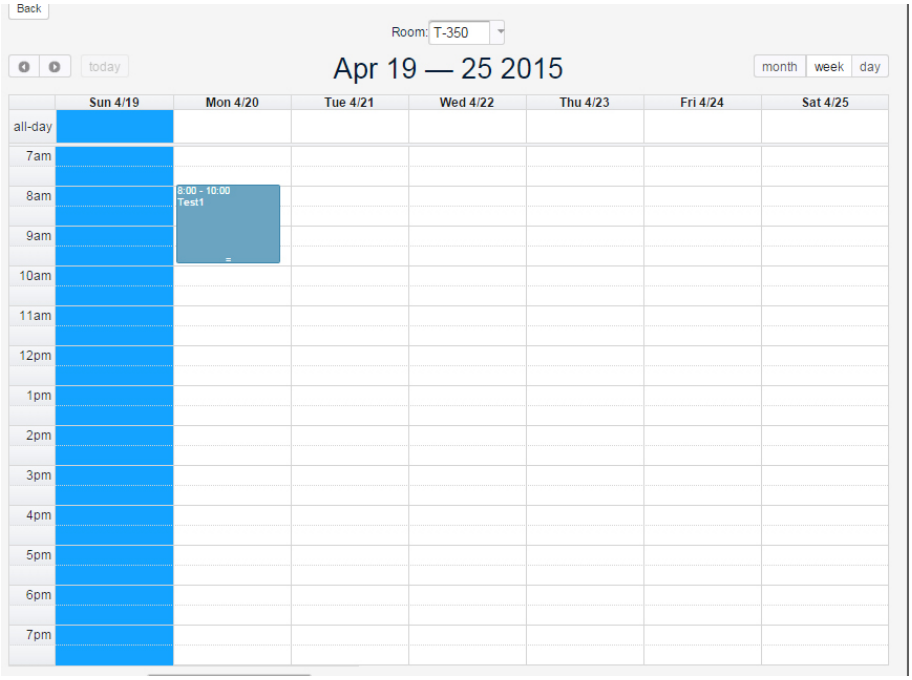


Obr. 2.2: Finálna podoba zoznamu rezervácií rozvrhára

Kalendár sa zobrazí pri vytvorení novej rezervácie a slúži k zlepšeniu predstavivosti a časovom umiestnení rezervácie či už ide o zobrazenie denné, týždenné alebo mesačné. Kliknutím do kalendára a korektnom vyplnení pop-up okna sa vytvorí event, ktorý je možné odoslať a vytvoriť tým požiadavku na rezerváciu. Je možné eventov vytvoriť viac a teda vytvoriť komplexnú požiadavku na rezerváciu. Zvažoval som použitie komponenty *schedule (kalendár)*, no napokon som zistil, že plne vyhovuje mojim požiadavkam. Bolo by teda

zbytočné znova implementovať funkcionalitu, ktorá je rovno pripravená na použitie. Keďže používam knižnicu PrimeFaces aj so všetkými príslušnými zdrojovými kódmi, mám „voľnú ruku“ pri úprave či doplňovaní funkcionality tejto komponenty.

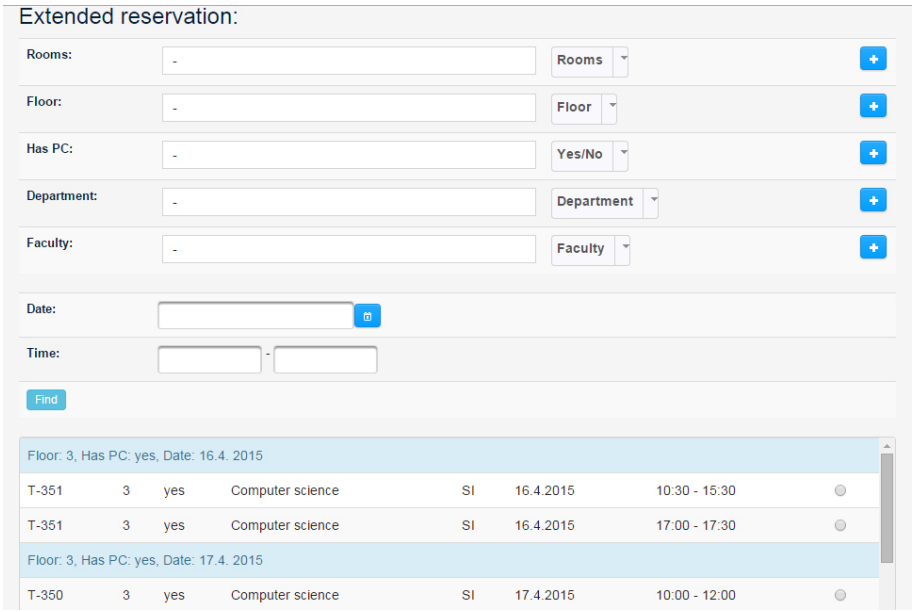
V spodnej časti máme zobrazený formulár, ktorého vyplnením sa dá rezervácia bližšie špecifikovať. Aplikácia umožňuje aj detailnejšie vytvorenie rezervácie, prípadne opakovaného zarezervovania miestností na určité obdobie. Do tejto časti sa užívateľ dostáva prostredníctvom tlačidla „Ext filter“, kde sú zobrazené ďalšie panely.



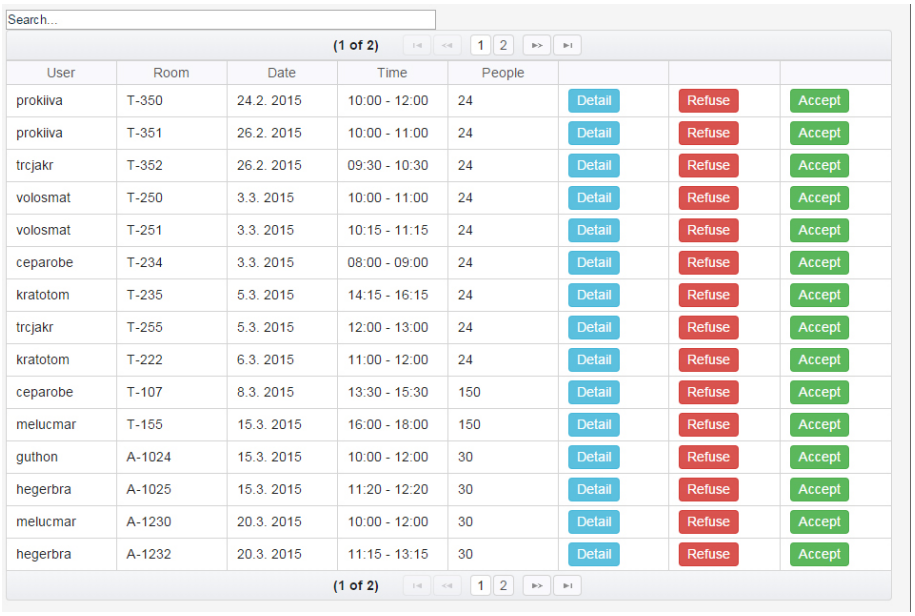
Obr. 2.3: Finálna podoba kalendára, umožňujúceho vytváranie rezervácií

Rozšírená forma rezervačného formulára, ktorá umožňuje prispôsobovať požiadavky na mieru a to spôsobom „naklikania si“ parametrov, na základe ktorých systém vyhľadáva dostupné rezervácie. Je možné, že systém užívateľovi predloží viacero možností. V tomto prípade sa v spodnom panely, ako je demonštrované na obrázku 2.4, zobrazujú všetky z nich a užívateľ má možnosť vybrať si tú, ktorá mu najviac vyhovuje.

Nasledujúci obrázok 2.5 nám znázorňuje pohľad rozvrhára na zoznam všetkých vytvorených rezervácií. V príslušnej časti aplikácie rozvrhár rozhoduje o schválení či neschválení rezervácie a v prípade potreby je možná jej úprava.



Obr. 2.4: Finálna podoba rozšírenej funkcionality pri vytváraní komplexnejších rezervácií



Obr. 2.5: Finálna podoba zoznamu rezervácií v stave čakajúcom na schválenie/neschválenie

Kapitola 3

Testovanie

3.1 Testovanie kompatibility s prehliadačmi

Ku každému procesu vytvárania aplikácie neodmysliteľne patrí aj obdobie testovania vytvoreného produktu. Keďže nebolo presne stanovené pre aké zariadenia sa systém bude používať, nebolo nutné vykonávať zložité testovania. V mojom prípade úplne postačoval test kompatibility, čiže test použiteľnosti v najbežnejších prehliadačoch. Na následujúcej tabuľke zobrazujem testovanú funkcionalitu a jej dosiahnutú úspešnosť.

				
Verzia systému	43.0	37.0.2	12.16	11
Rýchlosť	|	|	|	-
Zobrazenie	|	|	|	|
Podpora komponentov	|	|	|	|

Tabuľka 3.1: Výsledok testov funkčnosti na jednotlivých prehliadačoch

Systém bol testovaný na operačných systémoch Windows 7 64 bit a Linux Ubuntu 12.10, pričom na Linuxe boli vykonané testy iba na Chrome a Firefoxe. Aplikácia obstála vo všetkých testoch veľmi dobre, no z dôvodu veľkého množstva JavaScriptu obsiahnutého v kóde *Java PrimeFaces* komponentov na prehliadači Internet Explorer mala problémy pri splnení rýchlostného testu. Pre názornú ukážku som postavil do pomeru najrýchlejší (Google Chrome) a najpomalší (Internet Explorer) prehliadač. Hodnoty zaznamenané v tabuľke

Komponenta	Súbor	Rýchlosť načítania		Veľkosť súboru	
		Pred	Po	Pred	Po
Schedule	schedule.js	513 ms	297 ms	131.7KB	49.97KB
Scrollpanel	scrollpanel.js	321 ms	154 ms	43.72 KB	14.12 KB
Paginator	paginator.js	91 ms	65 ms	10.09KB	5.91KB

Tabuľka 3.3: Porovnanie časovej zložitosti komponent pred a po minifikácii JavaScriptu

sú stále priemerom z troch nameraných, aby sme získali čo najpresnejší výsledok.

Operácia	Súbor	Chrome	IE
Načítanie základných frameworkov	jquery.js	273 ms	163 ms
	bootstrap.css	171 ms	166 ms
Vykresľovanie komponenty schedule	schedule.js	513 ms	276 ms
	schedule.css	241 ms	227 ms
Vykresľovanie komponenty scrollpanel	scrollpanel.js	321 ms	215 ms
	scrollpanel.css	184 ms	178 ms
Vykresľovanie komponenty paginator	paginator.js	91 ms	86 ms
	paginator.css	68 ms	68 ms

Tabuľka 3.2: Porovnanie časovej zložitosti komponent medzi prehliadačmi

Ako možno z tabuľky vyčítať, rýchlostné rozdiely zobrazovania jednotlivých *PrimeFaces* komponentov sú značné. Túto situáciu som sa pokúsil vyriešiť minifikáciou JavaScriptového kódu. Je to proces odstraňovania prebytočných znakov typu *whitespace*, tab a ďalšie, ktoré síce dopomáhajú k lepšiemu

formátovaniu kódu pre programátora, no počítaču to prácu neuľahčuje. Práve naopak a preto čím kratší kód, tým väčšia bude rýchlosť načítania stránky.

Pokus sa napokon vyplatil a namerané časy som opäť zaznamenal pre porovnanie do tabuľky:

Touto úpravou bola aplikácia rýchlostne optimalizovaná aj pre prehliadač Internet Explorer, čo znamená, že testovanie je úspešne ukončené.

Výsledná aplikácia doposiaľ nebola prepojená s back-endovou časťou, ale po dohode s vedúcim práce je tento stav implementácie postačujúci. Napriek tomu bola ale veľká časť môjho úsilia sústredená na analýzu či návrh užívateľského rozhrania.

Záver

Cieľom mojej bakalárskej práce bolo navrhnúť a implementovať užívateľské rozhranie webovej aplikácie umožňujúcej rezerváciu či správu učební. Cieľ sa mi podarilo splniť a to najmä vďaka dobre vypracovanej analýze a návrhom, ktorými sa zaoberá podstatná časť mojej práce.

Osobný prínos

Účasť na projekte takýchto rozmerov bola pre mňa prínosom najmä v tom, že som mal možnosť stáť pri zrode systému a jeho tvorby od samého základu až po testovanie funkčného prototypu. Ako pri návrhu tak i pri implementácii som využíl množstvo nástrojov a metód, ktoré znova o niečo rozšírili moje obzory a dali mi nenahraditeľné skúsenosti. Po čase som si znova pripomenul prácu s prototypovacím nástrojom Pencil pri návrhu wireframov, či s nástrojmi Bootstrap a PrimeFaces pri ich realizácií. Za prínos považujem aj to, že som sa naučil pracovať s Java EE frameworkom JSF, na ktorom je celá aplikácia postavená. Pri implementačnej časti som si vyskúšal aj generovanie kostry tried využitím JAXB schémy písanej v jazyku XML, kde som si overil správnosť návrhu konceptu aplikácie.

Pohľad ďalej

Keďže riadky kódu nemožno ohraničiť, tak ani výsledok našej práce nepovažujem za ukončený. Je tu stále veľa nevyriešených problémov, či nezodpovedaných otázok. Takže nech už bude naša práca stavebným kameňom iných projektov alebo len zapadne medzi ostatné, nadobudnuté skúsenosti nám už navždy ostanú.

Literatúra

[1] Čápka, D.: UML - Use Case Diagram. [online], 2011, [cit. 23. 4. 2015]. Dostupné z: http://www.itnetwork.cz/uml-use-case-diagram

[2] Löbb, P.: Aplikace vzorů v životním cyklu softwaru. [online], masarykova univerzita, Fakulta informatiky, [cit. 23. 4. 2015]. Dostupné z: https://is.muni.cz/th/365359/fi_m/Thesis_Lobb.txt

[3] Mlejnek, I. J.: Analýza problémové domény. [online], 2015, [cit. 25. 3. 2015]. Dostupné z: https://edux.fit.cvut.cz/courses/BI-SI1/_media/lectures/04/04.prednaska.pdf

[4] Maple, S.: The Curious Coder's Java Web Frameworks Comparison: Spring MVC, Grails, Vaadin, GWT, Wicket, Play, Struts and JSF. [online], 2013, [cit. 17. 3. 2015]. Dostupné z: http://zeroturnaround.com/rebellabs/the-curious-coders-java-web-frameworks-comparison-spring-mvc-grails-vaadin-gwt-wicket-play-struts-and-jsf/4/

[5] Schmidt, J.: Návrh a prototypování prvků uživatelského rozhraní. [online], 2014, [cit. 26. 3. 2015]. Dostupné z: https://edux.fit.cvut.cz/courses/BI-TUR/_media/lectures/06/tur6design.pdf

Dodatok **A**

Inštalačná príručka

Reszervačný systém je realizovaný formou webovej aplikácie, implementovaný v jazyku Java EE a testovaný na webovom serveri GlassFish. Táto príručka v sebe zahŕňa postup slúžiaci práve k nasadeniu aplikácie na už spomínanom serveri.

A.1 Požiadavky

Aplikácia bola primárne vyvíjaná a testovaná na operačnom systéme Windows 7 64 bit a Linux Ubuntu 12.10 s verziou Javy 1.7.0, podporou webového servera GlassFish 3.1.2 na webových prehliadačoch:

- Google Chrome 43.0.2357.52
- Mozilla Firefox 37.0.2
- Opera 12.16
- Internet Explorer 11

A.2 Inštalácia a nasadenie

Pre spustenie aplikácie je potrebná Java 1.7.0+, webový server GlassFish 3.1.2+ a webový prehliadač (odporúčaný je jeden z vyššie uvedených).

Systém som testoval na serveri poskytnutým vedúcim práce. Z dôvodu nemožnosti zaručenia funkčnosti servera v dobe testovania inými užívateľmi prikladám postup vytvorenia lokálneho webového servera a následný *deploying* aplikácie.

Postup:

1. stiahnutie GlassFish webového servera verzie min 3.1.2

2. inštalácia servera, spočíva v rozbalení stiahnutých súborov do akejkoľvek zložky
3. spustenie servera pomocou príkazu
   - na OS Unix: `glassfish4/glassfish/bin asadmin start-domain`
   - na OS Windows: `glassfish4\glassfish\bin asadmin start-domain`
4. upload .war súboru priloženého v zložke /app prostredníctvom webového rozhrania servera dostupnom v mojom prípade na: `http://resysapp.com`
5. spustenie aplikácie prostredníctvom odkazu vo webovom rozhraní
6. prihlásenie do systému použitím prihlasovacích údajov:
   - Username: admin
   - Password: admin
7. zastavenie servera pomocou príkazu
   - na OS Unix: `glassfish4/glassfish/bin asadmin stop-domain`
   - na OS Windows: `glassfish4\glassfish\bin asadmin stop-domain`

Dodatok **B**

Use case a slovník

B.1 Slovník a pojmy

Celá zverejnená príloha vznikla ako výsledok našej spoločnej práce, na ktorej sa podieľali všetci členovia týmu.

Zoznam účastníkov:

- U1: Fakultný rozvrhár
- U2: Katederný rozvrhár
- U3: Študent/Zamestnanec

Rozdelenie účastníkov podľa hierarchie:

Fakultný rozvrhár
: užívateľ s administratívnou funkciou nadradený funkcii katederný rozvrhár. FIT ČVUT má iba jedného fakultného rozvrhára.

Katederný rozvrhár
: užívateľ s administratívnou funkciou podradený funkcii fakultný rozvrhár. Každá katedra má jedného katederného rozvrhára.

Užívateľ patriaci pod katedru
: ide o užívateľa, ktorý patrí pod akú koľvek katedru. Rezerváciu miestnosti patriacej rovnakej katedre potvrdí systém automaticky, bez kontaktovania rozvrhárov. Do tejto kategórie patria učitelia a študenti.

Užívateľ nepatriaci pod katedru
: užívateľ nepatrí pod žiadnu katedru, každú rezerváciu mu v závislosti na type miestnosti potvrdzuje katederný alebo fakultný rozvrhár.

Rozdelenie miestností:

Katederná miestnosť
Miestnosť ktorú spravuje určitá katedra.

Celofakultná miestnosť
Miestnosť nepatriaca žiadnej katedre, spravuje ju fakulta.

Mimofakultná miestnosť
Miestnosť spravovaná inou fakultou.

B.2 Usecases

1. UC1: Zobraziť miestnosti
   1.1. Zadá názov miestnosti
   1.2. Vyberie si typ miestnosti
2. UC2: Rezervovať miestnosť
   2.1. Miestnosť je katederná
      i. Užívateľ patrí rovnakej katedre ako miestnosť
      ii. Užívateľ nepatrí rovnakej katedre ako miestnosť
   2.2. Miestnoť nie je katederná
      i. Miestnosť je celofakultná
         A. Užívateľ vytvára rezerváciu pre predmet
         B. Užívateľ vytvára rezerváciu pre seba
         C. Užívateľ patrí katedre
         D. Užívateľ nepatrí katedre
      ii. Miestnoť je mimofakultná
3. UC3: Pridanie miestnosti
4. UC4: Zablokovanie miestnosti
5. UC5: Úprava miestnosti
6. UC6: Zrušenie rezervácie
   6.1. Zobraziť všetkých používateľov a rušiť rezervácie užívateľa (použije UC 10)
   6.2. Vyhľadať miestnosť (použije UC 1)
7. UC7: Úprava rezervácie
8. UC8: Schválenie/Neschválenie rezervácie

8.1. Miestnosť je katederná
   i. Užívateľ patrí rovnakej katedre ako miestnosť 10)
   ii. Užívateľ nepatrí rovnakej katedre ako miestnosť
8.2. Miestnoť nie je katederná
   i. Miestnoť je celofakultná
      A. Užívaťel vytvára rezerváciu pre predmet
      B. Užívaťel vytvára rezerváciu pre seba
      C. Užívaťel patrí katedre
      D. Užívaťel nepatrí katedre
   ii. Miestnoť je mimofakultná

9. UC9: Zobrazenie svojich rezervácií
10. UC10: Zobrazenie používateľov
11. UC11: Blokovanie používateľov
12. UC12: Úprava práv používateľov
13. UC13: Vyhľadať dostupné miestnosti
   13.1. Zadá dátum a názov miestnosti
   13.2. Zadá dátum a vyberie si typ miestnosti

Dodatok C

Zoznam použitých skratiek

GUI Graphical user interface

IDE Integrated development environment

JSF Java Server Faces

UX User experience

XML Extensible markup language

Dodatok D

Obsah priloženého CD

- `readme.txt` ..... stručný popis obsahu CD s inštalačnou príručkou
- `exe` ..... adresár so spustiteľnou formou implementácie
  - `resysApp.war` ... súbor obsahujúci aplikáciu, určený pre nasadenie na serveri
- `src`
  - `impl` ..... zdrojové kódy implementácie
  - `schema` ..... zdrojové kódy schémy datového formátu
  - `img` ..... použité obrazové materiály
  - `thesis` ..... zdrojová forma práce vo formáte LaTeX
- `text` ..... text práce
  - `prokiiva_BP_2015.pdf` ..... text práce vo formáte PDF