Sem vložte zadanie Vašej práce.

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Bakalárska práca

Administrátorská část rezervačního systému učeben FIT

Patrik Turcaj

Vedúci práce: Ing. Ondřej Guth, Ph.D.

12. mája 2015

Poďakovanie

Chcel by som poďakovať vedúcemu práce ktorým je Ing. Ondřej Guth, Ph.D. za ochotu a čas ktorý mi venoval pri uvedení do problematiky a pri jej následnom riešení.

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 12. mája 2015 .....

Odkaz na túto prácu

Turcaj, Patrik. *Administrátorská část rezervačního systému učeben FIT*. Bakalárska práca. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2015.

Abstrakt

Cieľom tejto bakalárskej práce je analyzovať, navrhnúť a implementovať Administrátorskú časť rezervačného systému pre FIT. Aplikácia získava dáta zo školských systémov a poskytuje ich formou webovej služby. Implementácia je založená na platforme Java EE.

Klíčová slova Webová služba, Java EE, Rezervácia miestností, Sirius, KOSapi

Abstract

The goal of this bachelor thesis is to analyze, design and implement Administration part of reservation system for FIT. Application collects data from other school systems and then provides this data as a web service. Implementation is based on the Java EE platform.

Keywords Web service, Java EE, Room reservation, Sirius, KOSapi

Obsah

Zoznam obrázkov

Úvod

Súčasný systém rezervácie miestností na FIT ČVUT so sebou prináša viaceré problémy, vďaka ktorým sa stáva rezervovanie miestností na fakulte neefektívne. Bežne dostupné rezervačné systémy však nie sú pre fakultu dosť variabilné, pretože fakulta používa rozdelenie miestností v závislosti na katedrách a role používateľov s rôznymi právami na rezerváciu týchto miestností. Tieto skutočnosti sú dôvodom na vytvorenie rezervačného systému miestností na FIT ČVUT, ktorý by zabezpečoval celý proces rezervácií miestností na FIT a uľahčil by tak rezerváciu žiadateľom aj rozvrhárom.

Najväčšou motiváciou zapojiť sa do projektu je pre mňa skutočnoť, že sa jedná o prácu návrhovo-implementačnú, ktorá musí byť založená na platforme `Java EE` a s ktorou som pred touto prácou nemal žiadne skúsenosti. Aplikáciu bude spoločne implementovať viac študentov. Keďže práca bola zadaná pre viac ľudí, ktorí majú rovnaké zadanie práce, pracovali sme na analýze spoločne. Preto sa v mojej práci nevyskytnú všetky časti analýzy, ale len tie, na ktorých som pracoval sám. Spoločné časti práce, ktoré sú potrebné pre správne pochopenie celej práce budú uvedené v prílohách.

Cieľ práce

Pretože na aplikácií bude pracovať viac ľudí, bolo potrebné projekt rozdeliť. Aplikácia sa rozdelila na webovú časť a administrátorskú časť, ktoré spolu budú komunikovať ako webová služba. Úlohou webovej časti je implementovať webové rozhranie, ktoré bude slúžiť na prepojenie používateľov a administrátorskej vrstvy.

Cieľom mojej práce je analyzovať, navrhnúť a implementovať administrátorskú časť webovej aplikácie, ktorá bude podporovať celý proces rezervácie miestností na FIT.

Administrátorská časť musí úzko spolupracovať so školskými externými systémami, aby bola schopná automaticky reagovať na zmeny v rozvrhu. Aplikácia musí nadviazať na súčasný spôsob rezervácie miestností, zachovať používanú hierarchiu používateľov a s tým spojené zabezpečenie aplikácie a brať ohľad na rozdelenie miestností na fakulte.

Aplikácia bude otestovaná pomocou k tomu vhodných metód a ďalej dostatočne zdokumentovaná tak, aby bolo jednoduché nasadiť ju v rámci infraštruktúry FIT. Práca ďalej zhodnotí ekonomicko-manažérske prínosy aplikácie ako pre používateľov, tak aj pre fakultu.

Výsledkom celého projektu by mala byť aplikácia ktorá výrazne zjednoduší prácu fakultnému rozvrhárovi, poskytne používateľom informácie o obsadenosti miestností na fakulte a umožní im rezervovať si miestnosť na požadovaný čas. Tým pádom bude umožnené rezervovať si miestnosť nie len pre vyučujúcich, ale aj pre študentov fakulty. Výsledná aplikácia by mala byť schopná plne nahradiť a zefektívniť doteraz používaný spôsob rezervácií miestností.

Štruktúra práce

Kapitola č. 1 je venovaná zoznámeniu sa s problémom, vysvetleniu pojmov potrebných k pochopeniu ďalších častí práce, analýze funkčných a nefunkčných požiadaviek, znázorneniu a vysvetleniu hierarchie používateľov, opísaniu doteraz používaného systému rezervácie miestností a vybraným prípadom použitia aplikácie.

V kapitole č. 2 bude nasledovať návrh riešenia aplikačnej logiky, integrácia a poskytovanie dát a architektúra aplikácie. Ďalej tu budú spomenuté hlavné objekty aplikácie, ich popis a spôsob ukladania a získavania. Na konci kapitoly budú uvedené zvolené technológie.

Ďalšia kapitola č. 3 sa venuje implementácií aplikácie. Tu budú priblížené hlavné implementované objekty aplikácie, logika práce s nimi a rozdelenie aplikácie na vrstvy. Na záver implementácie budú pridané zaujímavé ukážky implementácie.

Nasleduje kapitola č. 4 kde bude aplikácia testovaná podľa prípadov použitia a testovacích scenárov.

Ďalšou v poradí je kapitola č. 5 o nasadení aplikácie , v ktorej je zdokumentovaná konfigurácia prostredia na správne fungovanie aplikácie a návod, ako aplikáciu správne nainštalovať a spustiť. Predposlednou kapitolou č. 6 je ekonomicko-manažérske zhrnutie prínosov práce pre FIT ČVUT.

Záver sa bude venovať zhodnoteniu práce, prínosom práce pre autora aj pre FIT a výhľadom do budúcnosti.

Kapitola 1

Analýza

V tejto kapitole je popísaná funkcionalita implementovanej aplikácie, jej porovnanie s doteraz používaným spôsobom rezervácie miestností a predovšetkým rozpis požiadaviek na systém a ich detailné rozobratie na problémy, aby bolo jasné aká funkcionalita sa od výslednej aplikácie očakáva. Ďalej sú uvedené prípady použitia pre vybrané funkčné požiadavky a definuje sa, ktoré dáta budú prijímané z jednotlivých externých zdrojov.

1.1 Terminológia

Pred uvedením samotnej analýzy požiadaviek sú v tejto kapitole definované pojmy potrebné k pochopeniu ďalších častí bakalárskej práce.

Rozdelenie účastníkov podľa hierarchie

Fakultný rozvrhár
: Používateľ s funkciou administrátora, nadradený funkcii katedrový rozvrhár. FIT ČVUT má iba jedného fakultného rozvrhára.

Katedrový rozvrhár
: Používateľ s funkciou administrátora, podradený funkcii fakultný rozvrhár. Každá katedra má jedného katedrového rozvrhára.

Používateľ patriaci pod katedru
: Používateľ, patriaci pod akúkoľvek katedru FIT.

Používateľ nepatriaci pod katedru
: Používateľ, nepatriaci pod žiadnu katedru FIT.

Rozdelenie miestností

Katedrová miestnosť
: Miestnosť ktorú spravuje určitá katedra.

Celofakultná miestnosť
Miestnosť nepatriaca žiadnej katedre, spravuje ju fakulta.

Mimofakultná miestnosť
Miestnosť spravovaná inou fakultou.

1.2 Požiadavky

V tejto kapitole sú popísané jednotlivé funkčné a nefunkčné požiadavky, ktoré boli schválené na základe stretnutí s vedúcim a zároveň zadávateľom práce.

1.2.1 Funkčné požiadavky

Zobraziť miestnosti

Systém umožní zobraziť používateľovi zoznam všetkých miestností, zoznam miestností podľa typu, alebo konkrétnu miestnosť podľa názvu. Pretože vychádzame so súčasného spôsobu rezervácie miestností, budú sa musieť u miestnosti evidovať atribúty ako názov, príslušnosť miestnosti ku katedre, príslušnosť miestnosti k fakulte, kapacitu a typ miestnosti.

Rezervovať miestnosť

Používateľ bude môcť v systéme zadať požiadavku na rezerváciu miestnosti. Požiadavka na rezerváciu obsahuje začiatok a koniec rezervácie, žiadanú miestnosť, dni v ktoré sa má rezervácia konať a dátum alebo počet týždňov, po ktorých má byť opakovanie rezervácie ukončené. Miestnosť je možné žiadať pre seba ako používateľa, alebo pre predmet vyučovaný na fakulte.

Pridanie miestnosti

V prípade, že FIT ČVUT získa nové vyučovanie priestory, napríklad od inej fakulty, umožní systém pridať administrátorovi do systému novú miestnosť. Po pridaní miestnosti do systému bude možné prevádzať s miestnosťou všetky operácie, ktoré systém pre miestnosti povoľuje.

Zablokovanie miestnosti

V prípade že miestnosť používaná FIT ČVUT bude dočasne mimo prevádzky, bude administrátorovi umožnené danú miestnosť v systéme zablokovať, aby ju nebolo možné rezervovať. K tejto požiadavke sa samozrejme viaže aj opätovné odblokovanie miestnosti.

Vyhľadať dostupné miestnosti

Umožňuje používateľovi graficky zobraziť dostupnosť jednotlivej miestnosti v danom období. Miestnosti vyhľadáva v období, ktoré používateľ vymedzí dátumom, teda zadáva dátum začiatku a konca obdobia. Vyhľadať dostupné miestnosti je možné podľa mena konkrétnych miestností alebo podľa typu miestností.

Úprava miestnosti

V prípade zmeny vlastností miestnosti bude v systéme administrátorovi umožnené zmeniť jednotlivé atribúty konkrétnej miestnosti.

Zrušenie rezervácie

Rezervačný systém je vytváraný na to, aby zefektívnil spôsob rezervácie miestností. Ak sa používateľovi naskytnú okolnosti, za ktorých nebude môcť v ním rezervovanom čase využiť miestnosť, bude mu umožnené svoju rezerváciu zrušiť, aby sa nestalo že by rezervovaná miestnosť ostala neobsadená. Fakultný rozvrhár bude môcť kedykoľvek zrušiť akúkoľvek rezerváciu.

Schválenie/Neschválenie rezervácie

Funkcia prijatia alebo odmietnutia rezervácie pre fakultného aj katederného rozvrhára. Katedrový rozvrhár schvaľuje rezervácie, ktoré sa týkajú jeho katedry. Fakultný rozvrhár schvaľuje rezervácie, ktoré sa netýkajú žiadnej katedry.

Zobrazenie svojich rezervácií

Systém umožní používateľovi po prihlásení zobraziť všetky svoje rezervácie a požiadavky na rezervácie.

Zobrazenie používateľov

Fakultný rozvrhár má možnosť zobraziť všetkých používateľov systému. Pri zobrazení používateľov budú rozvrhárovi zobrazené rezervácie používateľa alebo informácie o používateľovi.

Blokovanie používateľov

Funkcia blokovanie používateľa umožní fakultnému rozvrhárovi zabrániť danému používateľovi vytvárať požiadavky na rezervácie.

Úprava práv používateľov

Fakultný rozvrhár využije zobrazenie používateľov popísané vyššie. Následne mu bude umožnené používateľovi vytvoriť alebo zrušiť funkciu katedrový rozvrhár.

Odoslanie e-mailu

Systém bude schopný automaticky odoslať e-mail rozvrhárovi v prípade, že používateľ podá žiadosť o rezerváciu miestnosti ktorá patrí pod katedru rozvrhára, prípadne vytvorí rezerváciu na predmet, ktorý patrí pod katedru rozvrhára. Systém odošle žiadateľovi e-mail pri každom vyhodnotení požiadavky na rezerváciu. Teda ak rozvrhár požiadavku schváli alebo neschváli, príde používateľovi e-mail o schválení alebo neschválení rezervácie.

1.2.2 Nefunkčné požiadavky

- Systém bude sprístupnený ako webová služba
- Platforma `Java EE 7`
- Kompatibilita s existujúcimi zdrojmi
- Systém rozšíriteľný pre ďalšie vstupné a výstupné zdroje

1.3 Rozbor problematiky

1.3.1 Správa miestností

Z požiadaviek na systém v sekcií 1.2 vyplýva, že nie je vhodné získavať miestnosti z inej aplikácie, tak aby boli miestnosti pevne definované, pretože tým stráca systém svoju variabilitu. Ak bude správa miestností súčasťou aplikácie, bude mať administrátor systému nad miestnosťami plnú kontrolu, ktorá mu umožní rozhodnúť, ktoré miestnosti v prípade potreby sprístupniť alebo zablokovať, prípadne môže meniť detaily jednotlivých miestností.

K dosiahnutiu takejto variability je potrebné navrhnúť systém, ktorý vhodne vyrieši ukladanie spomenutých miestností a bude spĺňať požadované operácie nad miestnosťami v závislosti na roli používateľa.

1.3.2 Správa rezervácií

Hlavnou požiadavkou na systém je samozrejme správa rezervácií. Je pochopiteľné, že od systému je vyžadovaná kompletná správa rezervácií, teda pridávanie, úprava a rušenie záznamov, ktorý používateľovi umožní požiadavku nie len vložiť, ale v prípade potreby aj upraviť alebo úplne zrušiť. Na to, aby

používateľ efektívne vybral miestnosť a čas pre svoju udalosť, bude potrebné používateľovi zobraziť už existujúce udalosti v danej miestnosti. Administrátorská časť aplikácie teda bude musieť poskytnúť webovej časti vstupné dáta tak, aby z nich bolo jednoducho zistiteľné, v akom čase je miestnosť dostupná.

Pre potreby rozvrhárov musí mať systém funkciu na schválenie alebo neschválenie rezervácie, ktorým bude rozvrhár môcť požiadavku používateľa buď prijať alebo zamietnuť.

Je teda potrebné navrhnúť systém tak aby konzistentne ukladal dáta o rezerváciách, vedel správne priradiť rolu používateľovi a podľa toho mu sprístupniť operácie určené pre jeho rolu.

1.3.3 Používateľské role

Súčasný spôsob rezervácie miestností má v sebe zahrnutú hierarchiu používateľov ktorú je potrebné dodržať aj implementovanom systéme. Používatelia budú rozdeľovaní podľa nasledujúcej hierarchie:

1. Fakultný rozvrhár
2. Katedrový rozvrhár
3. Bežný používateľ

Bežný používateľ

Najviac používateľov pripadá na rolu bežný používateľ. Roli bežný používateľ sú v systéme sprístupnené základné funkcie systému, ktoré slúžia na bežnú správu rezervácií. Rolu bežný používateľ v systéme ešte rozdeľujeme na používateľa patriaceho pod katedru (akademický pracovník) a používateľa nepatriaceho pod žiadnu katedru. Bežnému používateľovi budú umožnené funkcie:

- Zobraziť dostupné miestnosti
- Vytvoriť rezerváciu
- Upraviť svoju rezerváciu
- Zobraziť svoje rezervácie
- Zrušiť svoju rezerváciu

Katedrový rozvrhár

Každá katedra na FIT ČVUT má v systéme jedného rozvrhára. Táto rola je v hierarchií implementovaného systému nadradená roli bežný používateľ. Katedrový rozvrhár má sprístupnené všetky funkcie bežného používateľa, naviac však potvrdzuje žiadosti o rezerváciu ktoré patria pod jeho katedru. Roli katedrový rozvrhár sú sprístupnené naviac tieto funkcie:

- Schválenie/Neschválenie rezervácie
- Zobraziť rezervácie podľa používateľa
- Zobraziť rezervácie pre miestnosť
- Zobraziť rezervácie na potvrdenie
- Zobraziť miestnosti
- Zablokovanie miestnosti

Fakultný rozvhrár

Fukncia fakultný rozvrhár je najvyššie v hierarchií systému. Fakultný rozvrhár je zároveň administrátorom systému a sú mu sprístupnené všetky funkcie systému. Fakulta má iba jedného fakultného rozvrhára. Fakultný rozvrhár má právo vytvoriť alebo odobrať rolu katedrový rozvrhár akademickému pracovníkovi fakulty. Fakultný rozvrhár môže potvrdiť akúkoľvek rezerváciu. Fakultný rozvrhár má sprístupnené všetky funkcie ktoré má sprístupnené aj katedrový rozvrhár, naviac má však sprístupnené aj tieto funkcie:

- Pridanie miestnosti
- Úprava miestnosti
- Zobrazenie používateľa
- Blokovanie používateľa

Preto je potrebné implementovať v systéme autentifikáciu používateľov, navrhnúť spôsob akým bude systém jednotlivé role používateľom priraďovať a zabezpečiť ho tak, aby v prípade, že webová časť povolí používateľskej roli metódu nadradenej role, administrátorská časť znemožnila použitie danej metódy.

1.3.4 Integrácia dát z Google Calendar

Pre efektívnejšiu rezerváciu miestností musí systém vedieť zobraziť používateľovi obsadenosť miestnosti, o ktorú používateľ žiada. To zahŕňa nie len rezervácie vytvorené naším systémom, ale aj pravidelnú výuku.

Keďže FIT ČVUT je pomerne mladá fakulta a nemá dostatok vlastných priestorov, zdieľa vybrané miestnosti s Fakultou Architektúry. K jednotlivým udalostiam výuky FA ČVUT máme prístup pomocou zdieľaného `Google Calendar` pre jednotlivé miestnosti.

1.3.5 Integrácia dát z informačného systému KOS

Ďalšou požiadavkou na systém je, aby vedel spolupracovať so školským informačným systémom `KOS`[1]. Je to z dôvodu, že systém musí pri rezervácií miestnosti zohľadniť aj pravidelnú výuku v miestnosti, používať iba miestnosti evidované fakultou, povoliť rezervácie iba osobám patriacim na FIT ČVUT, využívať katedry FIT ČVUT a podobne. `KOS` je školský študijný informačný systém, ktorý mimo iné zhromažďuje uvedené dáta o ČVUT.[1] Aby bol systém aktuálny čo sa týka spomenutých dát ohľadne fakulty, bude väčšinu z nich získavať práve z `KOS`.

1.3.6 Poskytovanie dát

Administrátorská časť aplikácie má byť implementovaná ako webová služba, je teda potrebné získať požadované dáta z externých zdrojov, spracovať ich do potrebného formátu a navrhnúť systém tak, aby vedel odpovedať na každú funkčnú požiadavku. Ďalej je potrebné vybrať typ webovej služby, protokol a formát poskytovaných dát.

1.4 Prípady použitia

Ako bolo v úvode spomenuté, na analýze spolupracovalo viac študentov. Preto budú v tejto kapitole rozpísané iba prípady použitia pre tie funkčné požiadavky, ktoré autor vypracoval sám. Ostatné prípady použitia budú uvedené v prílohe B.

PP č.2: Rezervovať miestnosť

Používateľ sa autentifikuje na webe, vyhľadá si miestnosť podľa prípadu použitia č.13 v prílohe B a v prípade záujmu o rezerváciu zadá požiadavku na

[1] `https://kos.is.cvut.cz/kos/logout.do`

systém. Systém následne skontroluje či je táto požiadavka splniteľná. To znamená, že musí skontrolovať či je miestnosť v danom čase voľná alebo či používateľ patrí pod FIT. Miestnosti sa rezervujú buď jednorázovo, na dátumom určené obdobie alebo na jeho periódu (napr. každý pondelok). Za predpokladu, že si miestnosť chce rezervovať, môžu nastať nasledujúce prípady:

- **Miestnosť je katedrová**
  - **Používateľ patrí rovnakej katedre ako miestnosť**
    Systém automaticky rezervuje miestnosť bez ďalšieho súhlasu rozvrhára a informuje používateľa.
  - **Používateľ nepatrí rovnakej katedre ako miestnosť**
    Odoslanie žiadosti o potvrdenie rezervácie katedrovému rozvrhárovi katedry ktorej patrí daná miestnosť. Kým žiadosť nie je potvrdená, používateľ ju môže zrušiť. Ak ju rozvrhár schváli, zapíše ju do systému `KOS`. Systém toto rozhodnutie oznámi používateľovi.
- **Miestnosť nie je katedrová**
  - **Miestnosť je celofakultná**
    - **Používateľ vytvára rezerváciu pre predmet**
      Odoslanie žiadosti o potvrdenie rezervácie katedrovému rozvrhárovi katedry o ktorej predmet sa jedná. Kým žiadosť nie je potvrdená, používateľ ju môže zrušiť. Ak ju rozvrhár schváli, zapíše ju do systému `KOS`. Systém toto rozhodnutie oznámi používateľovi.
    - **Používateľ vytvára rezerváciu pre seba**
      - **Používateľ patrí katedre**
        Odoslanie žiadosti o potvrdenie rezervácie katedrovému rozvrhárovi, ktorej je používateľ členom. Kým žiadosť nie je potvrdená, používateľ ju môže zrušiť. Ak ju rozvrhár schváli, zapíše ju do systému `KOS`. Systém toto rozhodnutie oznámi používateľovi.
      - **Používateľ nepatrí katedre**
        Odoslanie žiadosti o potvrdenie rezervácie fakultnému rozvrhárovi. Kým žiadosť nie je potvrdená, používateľ ju môže zrušiť. Ak ju rozvrhár schváli, zapíše ju do systému `KOS`. Systém toto rozhodnutie oznámi používateľovi.
  - **Miestnosť je mimofakultná**
    Odoslanie žiadosti o potvrdenie rezervácie fakultnému rozvrhárovi. Kým žiadosť nie je potvrdená, používateľ ju môže zrušiť. Ak ju rozvrhár schváli, zapíše ju do systému `KOS`. Systém toto rozhodnutie oznámi používateľovi.

Proces rezervovania miestností uzatvára prípad použitia č. 8 uvedený v prílohe B, kde sú rozpísané prípady, kedy treba rezervovanie miestnosti zapísať aj do iného systému. Tento proces je pre lepšie pochopenie graficky znázornený diagramom aktivít C.

PP č.4: Zablokovanie miestnosti

Rozvrhár po autentifikácií na webe využije prípad použitia č. 1 uvedený v prílohe B, pomocou ktorého si zobrazí miestnosť. Následne zvolí možnosť `Upraviť` alebo `Blokovať`. Fakultný rozvrhár môže blokovať akúkoľvek miestnosť, katedrový rozvrhár len miestnosti svojej katedry. Pri vyhľadaní miestnosti bude pri miestnosti zobrazená informácia o zablokovaní miestnosti.

PP č.5: Úprava miestnosti

Fakultný rozvrhár si po autentifikáci na webe zobrazí miestnosť pomocou prípadu použitia č. 1 uvedený v prílohe B. V príslušnej miestnosti zvolí možnosť upraviť miestnosť. Atribúty evidované u miestnosti sú: názov miestnosti, kapacita, príslušnosť ku katedre, príslušnosť k fakulte a typ miestnosti. Pri type miestnosti bude upravujúcemu ponúknuté na výber z evidovaných typov miestností. V prípade, že požadujúci typ nie je evidovaný, bude fakultnému rozvrhárovi povolené vytvoriť nový typ miestností. Každý typ miestností má pre seba špecifikované kapacity, ktoré u daného typu existujú. Výber katedry a fakulty bude rozvrhárovi ponúknutý z evidovaných fakúlt a katedier.

PP č.12: Úprava práv používateľov

Fakultný rozvrhár sa pri príchode na web autentifikuje. Zobrazí si používateľa podľa identifikátoru. Následne bude fakultnému rozvrhárovi ponúknutá možnosť pridania, alebo odobratia funkcie fakultný rozvrhár pre danú osobu.

1.5 Súčasný spôsob rezervovania miestností

Neefektívnosť súčasného spôsobu rezervácií sa stala hlavným dôvodom na vytvorenie nového rezervačného systému, ktorý by uľahčil prácu rozvrhárom aj rezervujúcim. Súčasne používaný spôsob rezervacií je založený predovšetkým na e-mailovej komunikácií medzi žiadateľom o rezerváciu a niektorým z rozvrhárov. V časti č. 1.4 - prípady použitia pre rezerváciu miestnosti sme si objasnili, aké prípady môžu nastať a aké zložité je rozhodnúť, ktorému rozvrhárovi je potrebné žiadosť poslať. Keďže žiadateľom o rezerváciu nemusia byť tieto prípady známe, často sa stáva, že žiadateľ o rezerváciu požiada e-mailom o miestnosť u úplne iného rozvrhára. Ten musí e-mail od žiadateľa preposlať správnemu rozvrhárovi, ktorý musí overiť, či žiadaná miestnosť nie je v danom

čase rezervovaná. V prípade že by bola obsadená, musí rozvrhár oznámiť túto skutočnosť žiadajúcemu a celý proces sa opakuje.

Kapitola 2

Návrh

V kapitole návrh bude podrobne popísaný návrh administrátorskej časti webovej aplikácie. Rovnako budú bližšie objasnené zvolené technológie a dôvody, prečo boli zvolené, detailný návrh objektov, ich popis, spôsob ukladania a dôvod, prečo bol daný spôsob ukladania zvolený.

2.1 Návrh riešenia problematiky

V tejto kapitole práce je rozpísaný návrh problematiky uvedený v časti č. 1.3.

2.1.1 Používateľské role

Pretože sú evidované rôzne role používateľov, je potrebné používateľov rozpoznávať. Aby sme sa vyhli registrácií, bude na autentifikáciu použitý systém `Shibboleth`, ktorý FIT ČVUT používa aj pre ostatné školské aplikácie. Autorom implementovaná časť získa od webovej časti identifikátor používateľa, v tomto prípade username, ktorý spĺňa požiadavku jednoznačnosti. Webová časť posiela username administrátorskej časti s každým dotazom na webovú službu. Systém pred každým dotazom overí v systéme `KOS`, či používateľ patrí na FIT ČVUT, či patrí pod katedru alebo nie a či nezastáva funkciu rozvrhára. Podľa toho mu bude priradená rola a budú mu sprístupnené jednotlivé funkcie systému.

2.1.2 Integrácia dát z informačného systému KOS

Po upozornení, že dáta z `KOS` nie sú úplne konzistentné a práca s nimi v spojení s touto prácou by prekročila dostupný čas, vyhradený na implementáciu aplikácie, bude systém získavať potrebné dáta zo systému `KOSapi`[2].

[2] `https://kosapi.fit.cvut.cz/projects/kosapi/wiki`

KOSapi je aplikačné rozhranie ktoré sprostredkuje prístup k vybraným častiam databáze KOS pomocou RESTful webovej služby.[2] Tým pádom výrazne uľahčuje vytváranie školských aplikácií, ktoré potrebujú so systémom KOS pracovať.[3] KOSapi poskytuje dáta vo formáte XML[3] a teda nebude zložité z týchto dát získať potrebné informácie.[4]

Pre získanie jednotlivých kalendárnych udalostí fakulty (výuka) bude použitý systém vytvorený absolventom nazývaný Sirius.[5]

2.1.3 Integrácia dát z Google Calendar

Jednotlivá výuka v učebniach bude získavaná z KOS pomocou systému Sirius a pomocou zdieľaného Google Calendar s FA ČVUT podľa toho, pod ktorú z fakúlt miestnosť patrí.[6] Na to, aby bolo možné získať dáta z kalendárov, poskytuje Google svoje vlastné aplikačné rozhranie. Na výber sú REST architektúra alebo Google APIs Client Library for Java.[7]

Spočiatku bolo v pláne využiť knižnice pre Java, avšak implementovaný systém potrebuje dáta z kalendárov iba čítať, teda bude jednoduchšie použiť architektúru REST tak, ako je použitá aj pri integrácií dát z KOS. Tá poskytuje dáta vo formáte JSON, teda bude jednoduché z nich získať nami požadované informácie.[8]

2.1.4 Poskytovanie dát webovej časti

Jednou z obecných požiadaviek na administrátorskú časť rezervačného systému je, aby bola ľahko rozšíriteľná pre vstupné a výstupné zdroje. Preto by nebolo vhodné spojiť webovú časť a administrátorskú časť do jedného celku, ale je potrebné implementovať prácu ako webovú službu ktorá poskytuje dáta.

Do úvahy prichádza spôsob REST[4] a protokol SOAP[5]. Dá sa povedať že oba štýly sú si do istej miery navzájom konkurenčné, oba zvládnu poskytovať dáta vo formáte XML a je len na autorovi zvážiť klady a zápory každého spôsobu a rozhodnúť, ktorý je vhodnejší pre jeho prácu.

REST je určený k uskutočňovaniu CRUD[6] operácií, na ktoré využíva HTTP[7] operácie, ako GET, POST, PUT a DELETE, ktoré majú pevne definovaný význam.[9] REST nie je určený k uskutočňovaniu ľubovoľných procedúr ako SOAP, je však otázkou, či sú pre administrátorskú časť iné operácie ako CRUD

[3] Extensible Markup Language
[4] Representational state transfer
[5] Simple Object Access Protocol
[6] Create, Read, Update, Delete
[7] Hypertext Transfer Protocol

potrebné. Dáta posielané cez REST nemajú nijak špecifikovanú formu výstupu, preto ich nie je možné čítať strojovo. V prípade použitia REST je teda potrebné zistiť formát dát z dokumentácie, alebo sa na výstup reálne pozrieť.[2]

Protokol SOAP nie je obmedzený iba na CRUD operácie. Je však možné jednoducho implementovať služby, ktoré by CRUD operáciám odpovedali. Služba SOAP umožňuje implementovať aj iné operácie ako CRUD, čo by mohlo zavážiť pri výbere použitej služby pri implementácií. Tieto operácie však nie sú štandardizované a je len na autorovi, ako si jednotlivé operácie pomenuje. Pre ukážku, ekvivalent HTTP služby POST by mohol vyzerať napríklad ako `insert(), save()` alebo `put()`. Tu by však programátor, ktorý by chcel využiť aplikáciu implementovanú v SOAP ako externý zdroj dát, musel použiť dokumentáciu pre zistenie jednotlivých názvov operácií.[10]

Najväčšou výhodou takejto webovej služby je, že nie je závislá na platforme ani programovacom jazyku. Požiadavka na jednoduchú rozšíriteľnosť je teda splnená.

Konečné rozhodnutie padlo na spôsob REST a to hlavne z dôvodu, že je používané aj na dostávanie dát z externých zdrojov. Rovnako však úlohu v rozhodovaní zohrala aj skutočnosť, že REST má pevne definované HTTP operácie. Fakt, že dáta posielané REST nemajú špecifickú formu je možné riešiť použitím technológie JAXB[8].

V technológií JAXB ide o to, že obe strany aplikácie si predom definujú schému XML výstupu pre každú triedu, ktorú si budú posielať. Tým sa docieli, že prenášané dáta medzi časťami nebude potrebné parsovať z XML ale budú automaticky prekonvertované na triedu definovanú v schéme. Vďaka tomu si časti posielajú priamo Java objekty, ale ostáva zachovaná koncepcia RESTful webovej služby.[11]

Skutočnosť, že webová časť má záujem o dáta z administrátorskej časti sa realizuje pomocou HTTP príkazov. Príkazy pre prípady použitia spomenuté v tejto bakalárskej práci budú uvedené nižšie, zoznam ostatných definovaných REST rozhraní, ktoré vypracovali kolegovia pracujúci na webovej časti budú uvedené v prílohe.

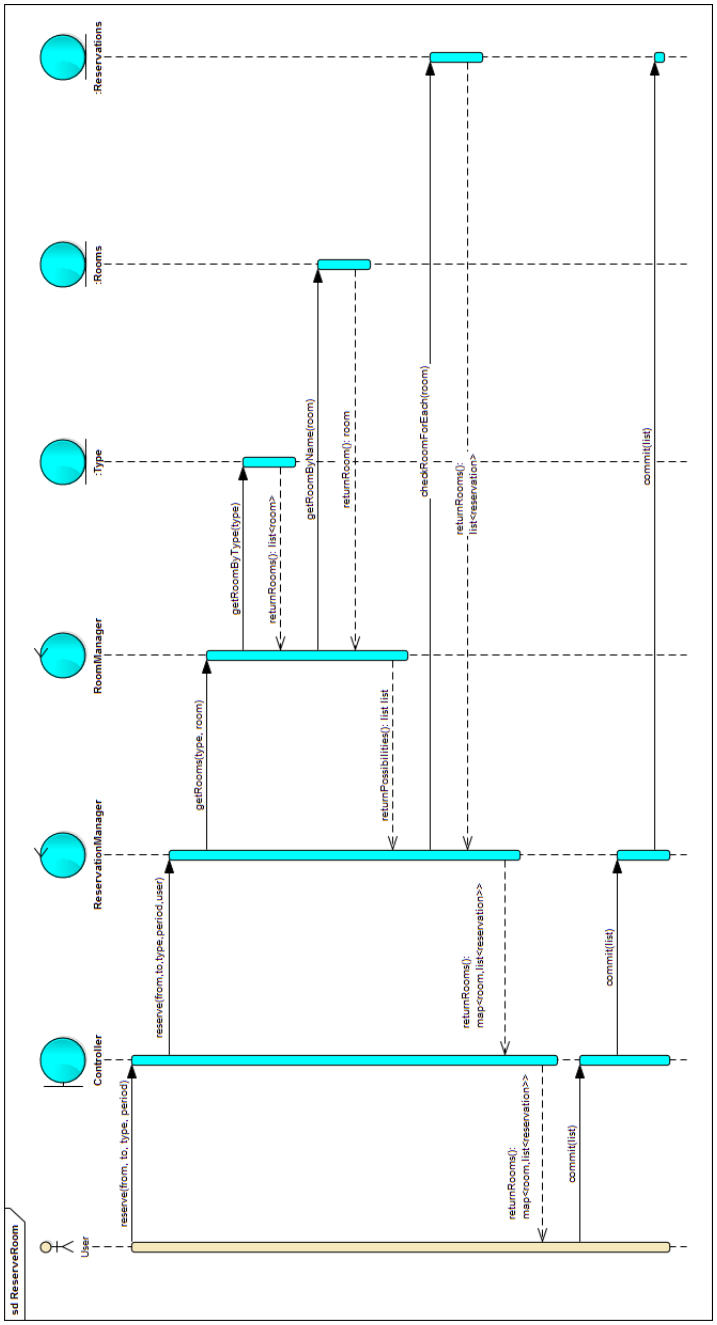
Pre lepšiu predstavu komunikácie medzi webovou a administrátorskou časťou sú nižšie priložené vybrané sekvenčné diagramy aplikácie.

[8] Java Architecture for XML Binding

Rezervovať miestnosť

`POST url_webovej_časti/api/events`

Posielané dáta budú vďaka architektúre `JAXB` objekt `rezervácia`.

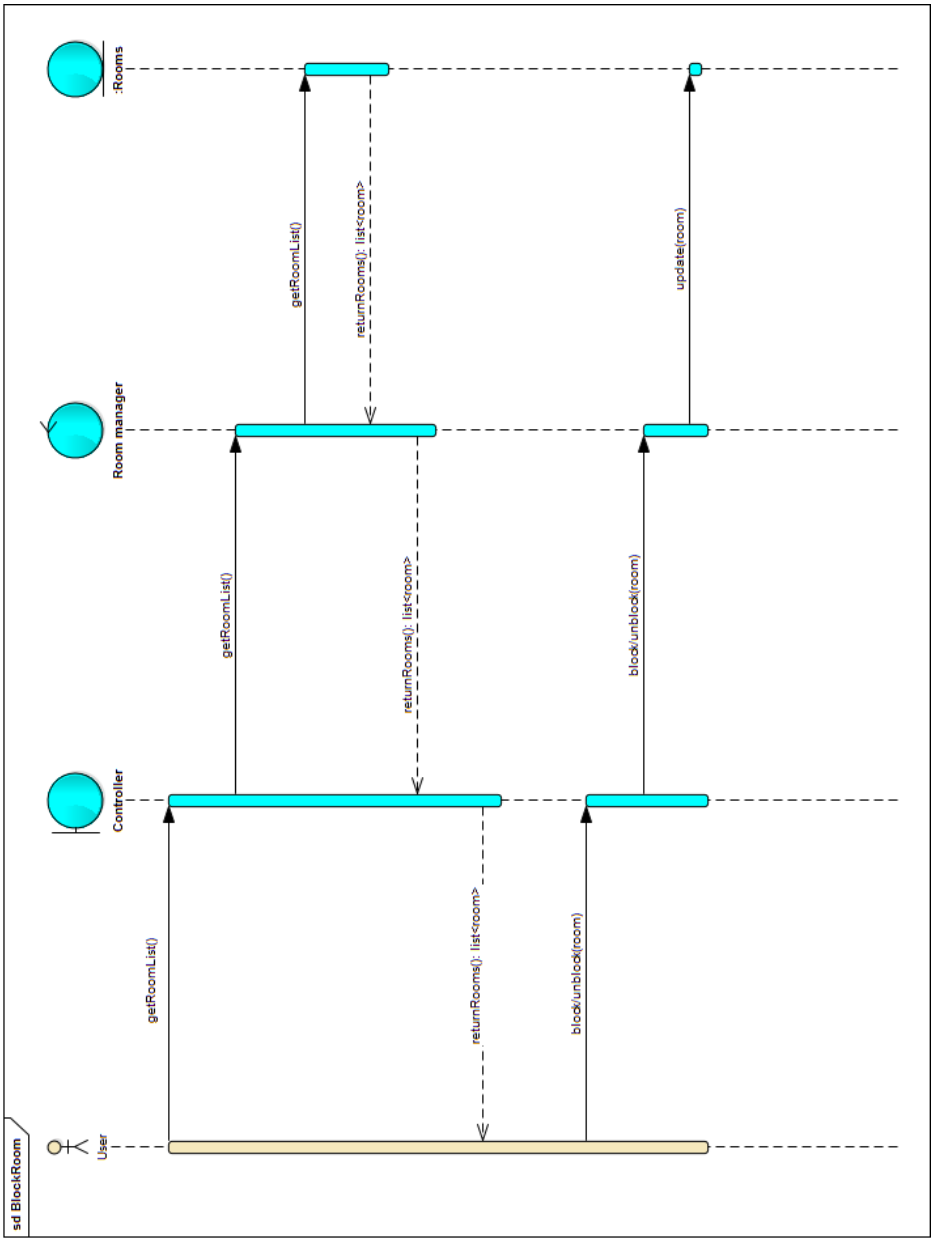


Obr. 2.1: Sekvenčný diagram pre rezerváciu miestnosti.

Zablokovanie miestnosti

`PUT url_webovej_časti/api/rooms`

Posielané dáta sú objekt `miestnosť`.

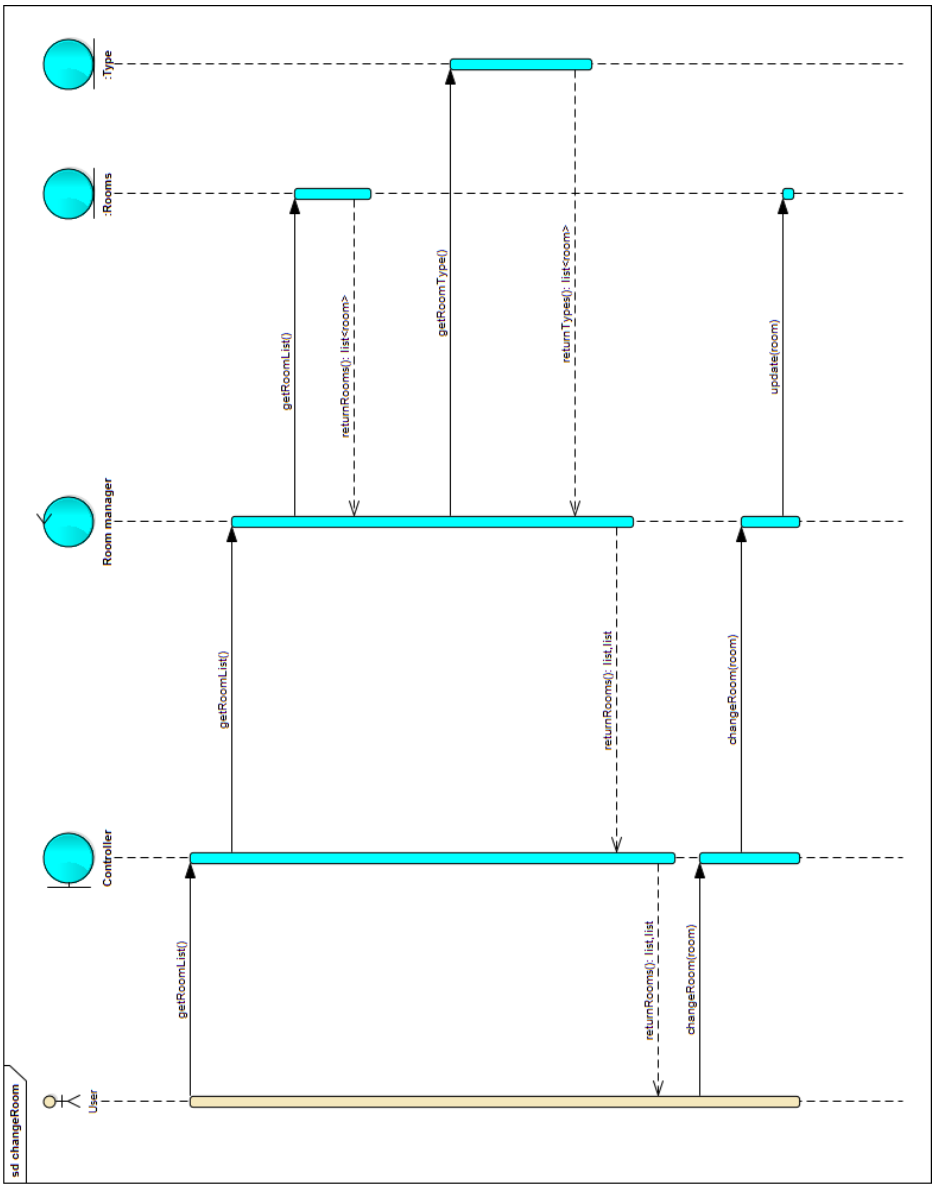


Obr. 2.2: Sekvenčný diagram pre zablokovanie miestnosti.

Úprava miestnosti

`PUT url_webovej_časti/api/rooms`

Posielané dáta sú objekt `miestnosť`.

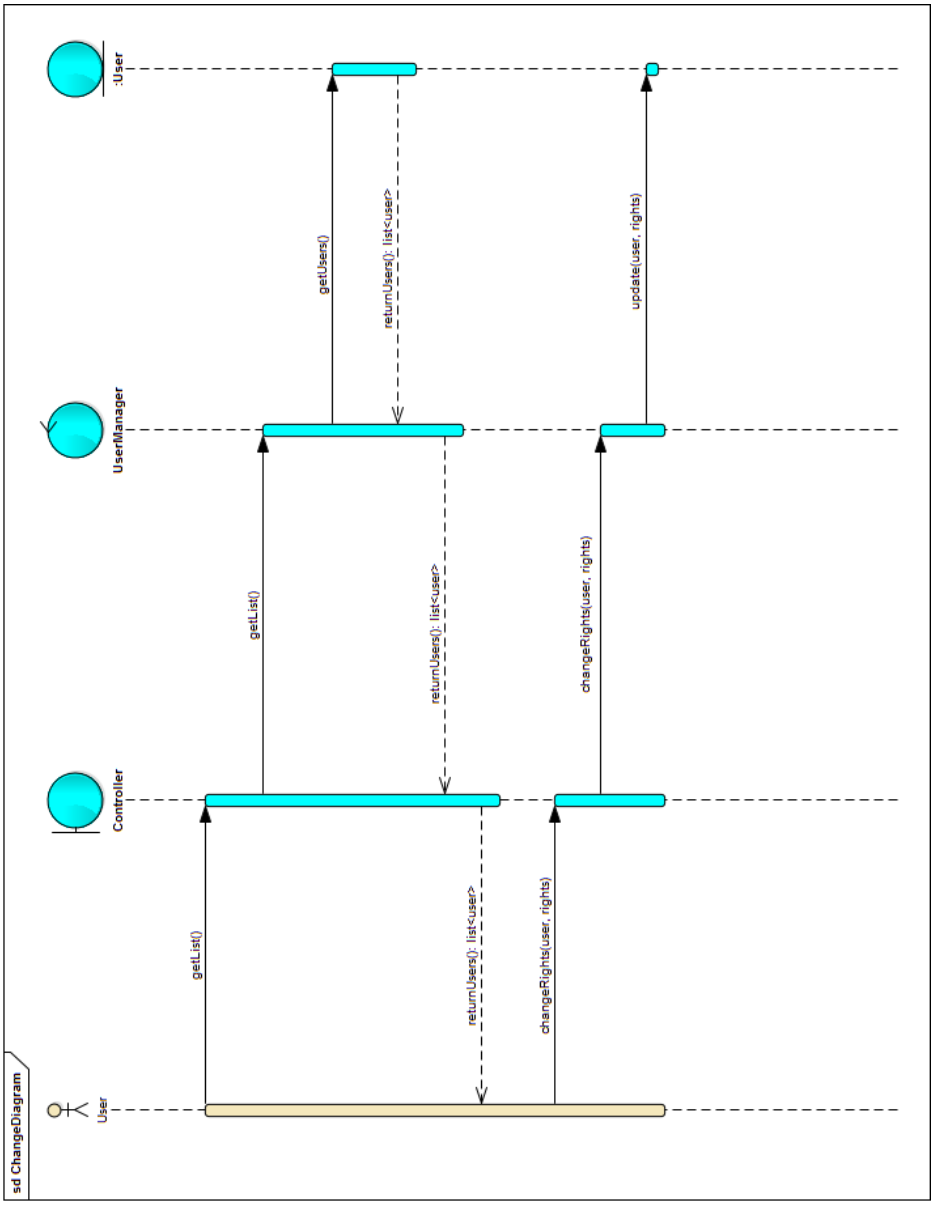


Obr. 2.3: Sekvenčný diagram pre zmenu vlastností miestnosti.

Úprava práv používateľov

`PUT url_webovej_časti/api/users`

Posielané dáta sú objekt `používateľ`.

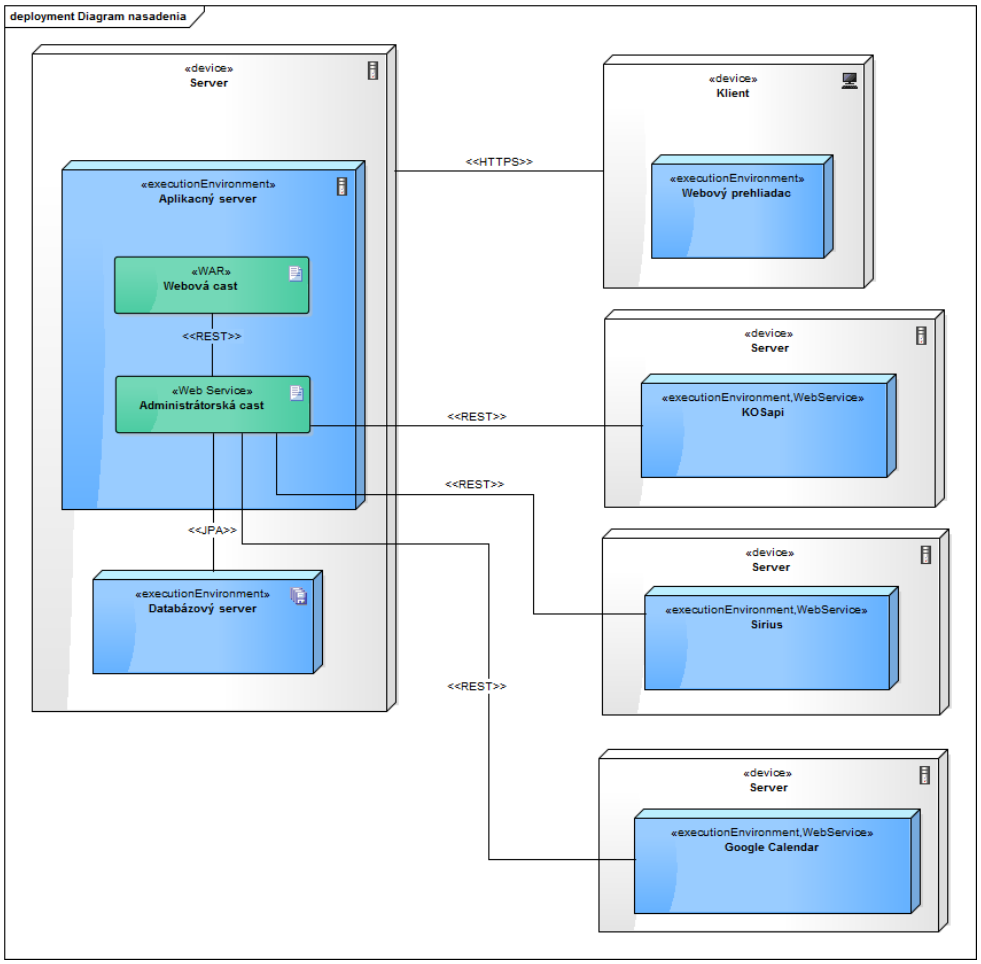


Obr. 2.4: Sekvenčný diagram pre zmenu práv používateľa.

2.2 Architektúra aplikácie

V predchádzajúcej kapitole bol uvedený návrh implementácie dát z externých systémov, návrh poskytovania dát webovej časti a je známe aj rozdelenie ap-

likácie na webovú a administrátorskú časť, je teda vhodné uviesť diagram nasadenia aplikácie.



Obr. 2.5: Diagram nasadenia aplikácie.

Diagram zobrazuje návrh kompletnej architektúry aplikácie, jednotlivé externé zdroje s ktorými administrátorská časť komunikuje, pričom asociácie medzi komponentami sú pomenované podľa typu komunikácie jednotlivých komponentov.

2.3 Objekty a ich popis

Teraz, keď už je hotový návrh získavania dát z externých systémov, budú popísané najhlavnejšie objekty implementácie. Predovšetkým je potrebné poukázať na dôvody, prečo boli dané objekty navrhnuté tak, ako boli navrhnuté. Jedná

sa o väčšinu objektov aplikácie. Pre lepšiu predstavu budú objekty doplnené návrhovým modelom.

Miestnosť

Implementovaná práca musí byť schopná poskytnúť zoznam všetkých miestností webovej časti aplikácie. Pre správne fungovanie aplikácie je potrebné u entity `Miestnosť` evidovať atribúty ako kapacita, názov, príslušnosť ku katedre, príslušnosť k fakulte a typ miestnosti. Pôvodným plánom bolo dostať požadované dáta zo systému `KOS`. FIT ČVUT používa mimo iné aj miestnosti FA ČVUT, ktoré v systéme `KOS` patria pod FA ČVUT. U týchto miestností nie je možné v systéme `KOS` rozpoznať, ktorú z nich používa aj FIT ČVUT, čo je hlavným dôvodom prečo nie je možné získavať miestnosti z `KOS`.

Kvôli spomenutým problémom bolo po dohode s vedúcim práce rozhodnuté že najrozumnejším riešením bude ukladať všetky miestnosti systému do systémovej databázy. Systém teda bude obsahovať funkcie ako vytvoriť, upraviť a zmazať miestnosť.

Typ miestnosti

Na FIT ČVUT sú rozdelené miestnosti na viac typov. Používateľ si bude môcť zobraziť výpis miestností podľa typu, teda rozumné by bolo mať typy miestností uložené v databáze a vytvoriť vzťah medzi entitami `Miestnosť` a `Typ miestnosti`. Typy miestností do databázy nie je potrebné zadávať manuálne, `KOSapi` nám poskytuje typy miestností pre FIT ČVUT. Teda stačí, aby mal administrátor systému v prípade potreby možnosť obnoviť databázu uložených typov miestností. Administrátor však bude môcť pridať do dazabázy aj typ miestnosti, ktorý v KOS nebude evidovaný.

Kapacita miestnosti

Na FIT ČVUT sú aj také miestnosti, ktoré sú rovnakého typu ale majú rozdielnu kapacitu. Ako príklad môže byť uvedený typ `Prednášková miestnosť`, ktorá môže mať kapacitu napríklad 150 ale aj 300 ľudí. Aby sa v implementácií predišlo duplikovaniu typov miestností ktoré majú rozdielnu kapacitu, budú tieto dva parametre rozdelené na osobitné objekty. Kapacita miestnosti bude v databáze ukladaná ako osobitná entita, ktorá bude mať reláciu s `Typ miestnosti` a to tak, že každý `Typ miestnosti` bude pre objekt `Kapacita` cudzím kľúčom. Vďaka tomu typy miestností nebudú duplicitné, ale pre každý typ bude ďalej možné vybrať jeho kapacitu.

Katedra

Ako bolo uvedené, zoznam katedier je možné získať z informačného systému `KOS`. Avšak pre pohodlnejšiu prácu s nimi, bude v databáze vytvorený objekt `Katedra`, ktorý bude cache pamäťou katedier, aby sa predišlo častému hľadaniu katedier v `KOS`. Aby však cache pamäť vytvorená systémom bola aktuálna, administrátor systému bude mať možnosť jednoducho, jedným kliknutím aktualizovať uložený zoznam katedier.

Fakulta

Keďže je rozdelenie miestností podľa fakulty evidované na fakultnú a mimofakultnú miestnosť, je potrebné mať fakulty uložené. Zatiaľ fakulta používa len miestnosti Fakulty Informačných Technológií a Fakulty Architektúry, nie je však vylúčené, že by FIT ČVUT mohla v budúcnosti využívať aj miestnosti iných fakúlt. `Fakulta` bude uložená ako osobitná tabuľka v databáze.

Vďaka tomuto návrhu rozdelenia atribútov miestnosti na jednotlivé objekty v databáze je možné efektívne pokryť všetky potreby vyhľadávania miestností, ktoré systém požaduje. Tým sa predišlo zbytočným duplicitám a redundancií dát v perzistencií, avšak vhodným zvolením `SQL` operácie `JOIN` sa dá dosiahnuť požadovaný výstup.

Rezervácia

Veľmi dôležitá entita aplikácie je samotná rezervácia. `Rezervácia` je objekt, ktorý obsahuje atribúty stav rezervácie, dátum začiatku, dátum konca, miestnosť, používateľ. Ako bolo spomenuté, objekt `Rezervácia` bude prijímaný z externého dátového zdroja, ktorým je v tomto prípade `Sirius`. Pre efektívnu prácu s objektami rezervácie je dôležité, aby rezervácie zo systému `Sirius` mali rovnakú formu, ako rezervácie vytvárané v implementovanom systéme.

Každú rezerváciu, ktorú rozvrhár potvrdí, musí zapísať do informačného systému `KOS`. Zatiaľ teda nie je potrebná vlastná tabuľka v databáze pre objekt `Rezervácia`, rezervácie je možné dostávať z `KOS`. Systém však umožňuje rezervujúcemu rezervovať si miestnosť katedry, pod ktorú sám patrí. Vtedy systém automaticky rezervuje miestnosť, bez ďalšieho súhlasu rozvrhára. Teda nie je do procesu zahrnutý nikto kompetentný, kto by danú udalosť do `KOS` zapísal. Pre tento prípad teda bude potrebné vytvoriť v databáze tabuľku pre rezervácie.

Ako bude neskôr popísané u entity `Fakultný rozvrhár`, ďalším dôvodom na to aby sme ukladali `Rezerváciu` do tabuľky v databáze je, že obsahuje atribút

stav. Kým rezervácia je v stave `čaká na schválenie`, je treba ju evidovať v databáze, až kým ju rozvrhár neschváli alebo nezamietne.

Používateľ

Entita `Používateľ` obsahuje atribúty meno, priezvisko, e-mail, `username` a príslušnosť ku katedre. Keďže študenti podľa systému `KOS` nepatria pod žiadnu katedru, bude u študentov spomenutý atribút prázdny. V prípade akademického pracovníka bude atribút štandardne vyplnený fakultou, pod ktorú patrí. Hodnoty atribútov pre entitu sú získavané z informačného systému `KOS` pomocou systému `KOSapi`. Keďže fakulta eviduje v systéme viac ako 6500 ľudí, neprichádza do úvahy aby boli všetci ľudia ukladaní do systémovej databázy.

Výhodou je, že tak ako `KOSapi` umožňuje vrátiť výpis všetkých ľudí fakulty, vie ich poskytnúť aj jednotlivo, na základe username. Keďže administrátorská časť pri každom prijatí príkazu od webovej časti dostane aj `username` používateľa, bude pri každom takomto príkaze najprv kontrolované, či `username` v `KOS` existuje a či sa jedná o používateľa patriaceho pod katedru alebo nie.

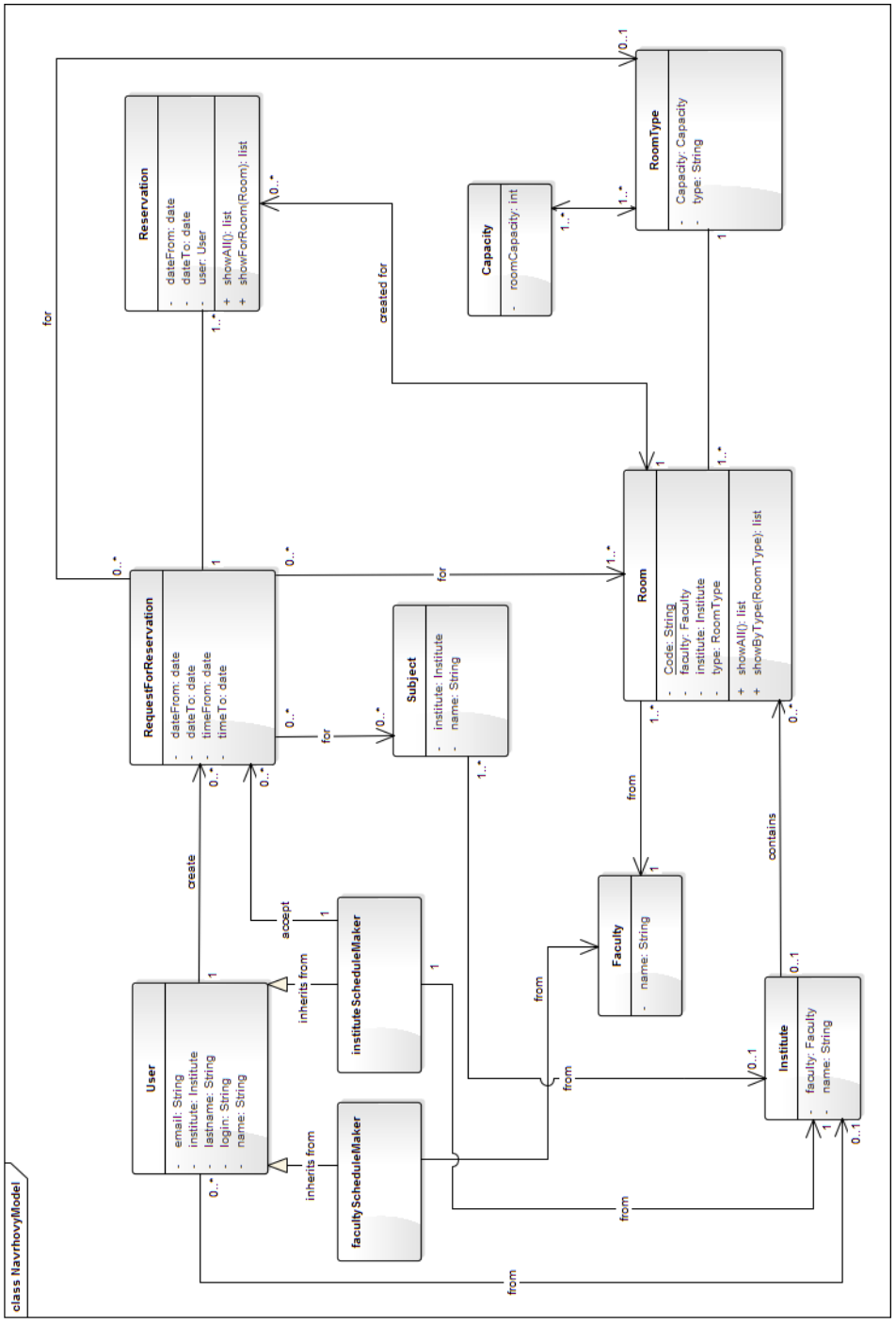
Katedrový rozvrhár

Objekt `Katedrový rozvrhár` vychádza z entity `Používateľ` a má aj totožné atribúty. Objekt `Katedrový rozvrhár` naviac obsahuje kolekciu `Rezervácií`, ktoré mu systém priradí na potvrdenie. Keďže `Katedrový rozvrhár` je vytváraný iba v našej aplikácií, bude implementačne najpraktickejšie mať vytvorenú tabuľku `katedrových rozvrhárov` v databáze a pri každom prijatom príkaze od webovej časti bude systém overovať, či používateľ nezastáva funkciu katedrového rozvrhára.

Fakultný rozvrhár

Fakultný rozvrhár rovnako vychádza z entity `Používateľ` a obsahuje rovnaké atribúty ako katedrový rozvrhár. Fakultný rozvrhár je v systéme iba jeden a zastáva funkciu administrátora systému. Fakultný rozvrhár bude rovnako ako katedrový rozvrhár uložený vo vytvorenej databáze.

V tejto kapitole bol popísaný návrh jednotlivých dôležitých častí systému, na ktorých bude aplikácia stáť. Samozrejme, nie sú to všetky objekty, ktoré sa v systéme vyskytujú. V systéme existujú aj objekty, ktoré sú akýmisi pomocnými objektami potrebnými pre správne fungovanie logiky aplikácie. V nasledujúcej kapitole bude popísané, ako sú jednotlivé objekty spracovávané systémom.



Obr. 2.6: Návrhový model tried.

2.4 Zvolené technológie

2.4.1 Glassfish

Po dohode s vedúcim a zadávateľom práce bolo rozhodnuté že výsledná aplikácia pobeží na aplikačnom serveri Glassfish. Glassfish je akýmsi predstaviteľom Java EE, ktorý dopĺňa technológie štandardnej edície Java SE. Súčasťou serveru Glassfish sú technológie ako Java Persitence API, Enterprise Java Beans a teda sa perfektne hodí pre tento typ aplikácie, keďže v nej sú dané technológie použité.[12]

2.4.2 PostgreSQL

Pre ukladanie spracovaných dát bol použitý databázový systém PostgreSQL. Zvolený bol najmä preto, lebo je voľne dostupný a obsahuje všetky dátové typy potrebné pre implementovanú aplikáciu. Pre lepšiu prácu s databázou bolo použité poskytované používateľské rozhranie pgAdminIII. Keďže finálna aplikácia bude spustená na aplikačnom serveri Glassfish, ktorý nevie pracovať s databázou ako PostgreSQL, bolo potrebné nainštalovať na server JDBC ovládač. Ďalším krokom bolo vytvorenie databázy, ktorá bola vytvorená pomocou používateľského rozhrania pgAdminIII. Následne sa musí cez administrátorskú konzolu serveru Glassfish vytvoriť Connection Pool – ten je používaný na prepojenie implementovanej aplikácie a samotnej databázy.[13] Tabuľky v databáze však neboli vytvárané pomocou SQL príkazov, použitá bola technológia Java Persitence API (JPA).

2.4.3 JPA

Java Persistence API je špecifikácia na prístup, ukladanie a správu dát medzi objektami/triedami v Java a relačnou databázou. JPA je iba samotná špecifikácia, nie je to produkt ktorý by zabezpečoval ukladanie dát sám o sebe. Na fungovanie vyžaduje databázu do ktorej môže dáta ukladať. JPA nevyžaduje znalosť jazyka SQL, s databázou pracuje na základe anotácií implementovaných v Java triede. Hlavným dôvodom prečo bolo JPA zvolené je, že na rozdiel od JDBC umožňuje do databázy uložiť celý objekt.[14]

2.4.4 JAXB

Java Architecture for XML Binding je architektúra umožňujúca vývojárom mapovať Java objekty do XML reprezentácie. JAXB má dve hlavné funkcie, vyjadriť Java objekt v XML reprezentácií a inverznú funkciu, vytvoriť Java objekt z XML reprezentácie.[11]

2.4.5 REST

REST je dátovo orientovaný softvérový architektonický štýl, ktorý je použiteľný pre jednotný a jednoduchý prístup k zdrojom. Každý zdroj má definovaný vlastný URI a REST definuje štyri základné prístupy k nim pomocou HTTP metód GET, POST, PUT a DELETE.[2]

Kapitola 3

Implementácia

V implementačnej bolo za úlohu vytvoriť administrátorskú časť webovej aplikácie implementovanej na platforme `Java EE`. Pre vývojové a testovacie účely bola vytvorená vlastná webová časť aplikácie, pretože komunikácia medzi webovou a administrátorskou časťou bola v harmonograme práce medzi poslednými úlohami. Implementácia začínala získavaním dát z externých zdrojov. Implementácia aplikácie však nebola dokončená a to z dôvodu veľkého rozsahu projektu. Po dohode s vedúcim práce sa autor kvôli časovej náročnosti sústredil predovšetkým na analýzu, návrh a iba časť implementácie. Administrátorská časť však bola implementovaná do bodu, kedy odpovedá na nasledovné funkčné požiadavky:

- Rezervovať miestnosť
- Pridanie miestnosti
- Vyhľadať dostupné miestnosti
- Úprava miestnosti
- Zrušenie rezervácie
- Schválenie / Neschválenie rezervácie
- Zobrazenie svojich rezervácií
- Zobrazenie používateľov
- Úprava práv používateľov
- Zablokovanie miestnosti

Chýbajúcimi požiadavkami sú:

- Odoslanie e-mailu

- Blokovanie používateľov

V tejto časti budú popísané najzaujímavejšie časti implementácie a body, ktoré sa pre časovú náročnosť projektu nestihli dokončiť.

3.1 Integrácia dát

V tejto časti je popísaný podrobný spôsob získavania dát z externých systémov.

3.1.1 Integrácia dát z informačného systému KOSapi

Predtým ako je možné získavať dáta od externých školských zdrojov, je potrebné svoju aplikáciu registrovať v službe `APPS MANAGER`[9]. Tento systém zabezpečuje prístup do aplikačných rozhraní všetkých potrebných školských systémov a je založený na štandarde `OAuth`.

1. Pri registrácií sa zvolí, o ktoré konkrétne `API` má autor záujem. V prípade našej aplikácie sú to `KOSapi` a `Sirius`. Systém aplikácií priradí `Client ID` a `Client Secret`. Ďalej je potrebné zvoliť názov aplikácie a doménu, na ktorú bude odoslaný `access_token`.

2. Po tom čo boli aplikácií priradené spomenuté údaje, je potrebné odoslať `GET` požiadavku na `URL` `https://auth.fit.cvut.cz/oauth/oauth/authorize` ku ktorému sa pridajú parametre, v ktorých je špecifikovaný požadovaný výstup. V našom prípade to je `code`, ďalej uvedieme priradené `Client ID` a doménu na ktorú bude kód odoslaný.

3. Po odoslaní požiadavku systém vyzve, aby sa autor autorizoval pomocou školského `username` a hesla. Po autorizácií autor dostane kód, ktorým zažiada o `access_token`.

4. O `access_token` sa žiada príkazom `POST` na `URL` `https://auth.fit.cvut.cz/oauth/oauth/token` kde sa v parametroch uvedie ako požadovaný výstup `authorization_token`. Ďalej sa uvedie `Cliend ID` a `Client Secret` ktoré boli aplikácií pridelené v kroku 1. Nakoniec je uvedená doména, na ktorú bude `access_token` odoslaný.[15]

   Teraz môže aplikácia kedykoľvek pristupovať do `KOSapi` bez toho aby si `KOSapi` od používateľa žiadalo prihlasovacie údaje. Stačí, že sa do `REST` príkazu pridá ako parameter získaný `access_token`.

[9] `https://auth.fit.cvut.cz/manager/index.jsf`

Ako bolo spomenuté, KOSapi poskytuje dáta ako RESTful webová služba. Implementovaná aplikácia posiela GET požiadavky na URL https://kosapi.fit.cvut.cz/api/3/ kde do URL pridá parametre ako požadovaný zdroj (napr. Teachers) a spomínaný access_token. Výstupom tohto požiadavku sú dáta vo formáte XML. Dáta z formátu XML sú dostávané pomocou jazyka XPath, ďalej spracované a uložené do požadovaného formátu.

5. Systém vráti objekt vo formáte XML, ktorý je potrebné celý načítať do dočasného dokumentu. Ako ukážka je priložená ukážka súboru zastupujúca instanciu učiteľa:

```
<atom:content atom:type="xml" xsi:type="teacher">
    <firstName>Ladislav</firstName>
    <lastName>Vagner</lastName>
    <personalNumber>304060</personalNumber>
    <titlesPost>Ph.D.</titlesPost>
    <titlesPre>Ing.</titlesPre>
    <username>xvagner</username>
    <division xlink:href="divisions/18101/">
    katedra teoretické informatiky
    </division>
    <email>ladislav.vagner@fit.cvut.cz</email>
    <extern>false</extern>
    <supervisionPhDStudents>ALLOWED</supervisionPhDStudents>
</atom:content>
```

6. Pomocou jazyka XPath sa zvolí časť XML, ktorá je pre autora zaujímavá. V prípade našej aplikácie /feed/entry/content.

7. Pre každú instanciu používateľa sa vytvorí prázdny objekt Používateľ v našom systéme a priradia sa mu atribúty získané z XML.

3.1.2 Integrácia dát zo systému Sirius

Systém Sirius slúži na získavanie jednotlivých kalendárnych udalostí z informačného systému KOS. Pôvodným plánom bolo využiť access_token získaný zo systému https://auth.fit.cvut.cz/manager/index.jsf. Pretože systém Sirius v čase implementácie nebol pripojený na OAuth server, bolo potrebné získať access_token inou cestou. Po dohode so správcom systému Sirius nám bol vygenerovaný univerzálny access_token ktorý nám povolí prístup do všetkých zdrojov systému Sirius.

Systém Sirius je rovnako sprístupnený ako RESTful webová služba. Systém poskytuje vlastnú RAML konzolu ktorá obsahuje všetky potrebné informácie k API, dostupné zdroje, zoznam podporovaných HTTP metód, query a url parametre, vstupné formáty a ukážky výstupov. S týmito informáciami je veľmi ľahké získať požadované dáta. Stačí, aby sa použila GET požiadavka na URL https://sirius.fit.cvut.cz/api/v1/ a doplnila cesta podľa toho, ktorý zdroj chce autor získať. Výstupom zo systému Sirius sú dáta vo formáte JSON.[5] Ako ukážka je priložená ukážka súboru zastupujúca instanciu jednej udalosti:

```
{"meta":{"
count":1,
"offset":0,
"limit":20},
"events":
[{"id":35990,
"name":null,
"sequence_number":12,
"starts_at":"2015-03-26T09:15:00.000+01:00",
"ends_at":"2015-03-26T10:45:00.000+01:00",
"deleted":false,
"capacity":320,
"event_type":"lecture",
"parallel":"2",
"links":{"course":"BI-LIN",
"room":"T9:105",
"teachers":["stampfra"],
"students": ["turcapat"...]
```

3.1.3 Integrácia dát z Google Calendar

Google ponúka pre svoje API autorizáciu prostredníctvom protokolu OAuth, ktorý bol využitý aj v minulých prípadoch integrácie. Na to, aby bol umožnený prístup bez používateľského súhlasu pri každom požiadavku na rozhranie, je potrebné získať API_key. Postup na jeho získanie je nasledovný:

1. Prvým krokom je vytvorenie developerského účtu Google.

2. Po vytvorení účtu je potrebné vytvoriť projekt v Google Developers Console[10].

3. Ďalším krokom je vygenerovanie API_key v záložke Credentials.

[10] https://console.developers.google.com/project

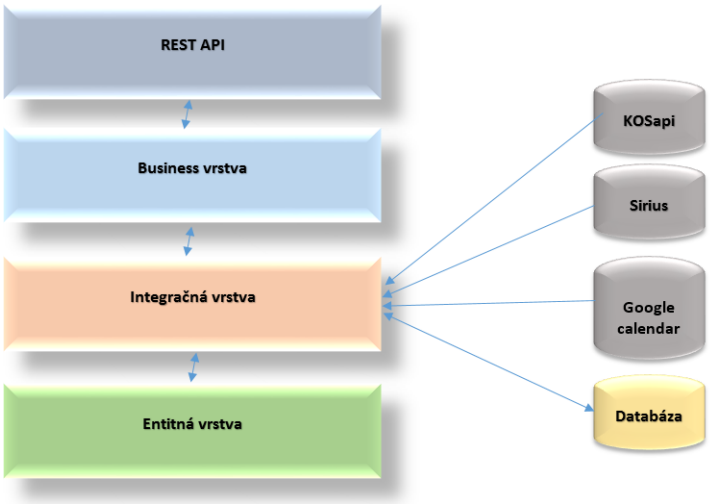
Po získaní kľúča sa ním pracuje rovnako ako s `access_token` pri integrácií dát z `KOS`. Pre získanie dát bude posielaná `GET` požiadavka na `URL` `https://www.googleapis.com/calendar/v3/calendars/` s tým, že ako `URL` cesta bude použitý `Calendar ID` požadovaného kalendára a reťazec `events` ktorý označuje, že má autor záujem o udalosti v kalendári. `URL` parametrami sú minimálny čas a dátum kedy musí udalosť začínať, maximálny dátum a čas kedy musí udalosť končiť, časovú zónu kvôli časovému pásmu a samozrejme vygenerovaný `API_key`.

Výstupom sú dáta vo formáte `JSON` ktorý už bol v aplikácií spomenutý a teda je jednoduché dáta spracovať a rozdeliť do požadovaného formátu.[16] Ako ukážka je priložená ukážka súboru zastupujúca instanciu jednej udalosti:

```
{
 "kind": "calendar#events",
 "etag": "\"1430909401956000\"",
 "summary": "_Posluch. č.105 300",
 "description": "",
 "updated": "2015-05-06T10:50:01.956Z",
 "timeZone": "UTC",
 "accessRole": "freeBusyReader",
 "defaultReminders": [],
 "nextSyncToken": "CKD97_T4rMUCEAAYCg==",
 "items": [
  {
   "kind": "calendar#event",
   "etag": "\"2829496114792000\"",
   "id": "87la9ln4sr23p106dqot3l0eus",
   "status": "confirmed",
   "updated": "2014-10-31T09:34:17.396Z",
   "start": {
    "dateTime": "2015-09-10T06:00:00Z"
   },
   "end": {
    "dateTime": "2015-09-10T16:00:00Z"
   },
   "iCalUID": "87la9ln4sr23p106dqot3l0eus@google.com"
  },
```

3.2 Rozdelenie na vrstvy

Po dôkladnej analýze problému a navrhnutí hlavných objektov aplikácie bolo rozhodnuté implementovať prácu v štyroch vrstvách.



Obr. 3.1: Rozdelenie aplikácie na vrstvy.

Prvá, najspodnejšia vrstva je vrstva entitných tried. Sú v nej definované triedy vytvorené podľa rozdelenia uvedeného v návrhu. Pretože sa autor rozhodol použiť technológiu `JPA`, každá trieda musí spĺňovať základné pravidlá entitnej triedy. V prvom rade sú to `JPA` anotácie `javax.persistence.Entity` ktoré špecifikujú, že daná trieda je entita. Trieda musí obsahovať `public` alebo `protected` konštruktor bez parametrov a perzistentné premenné nesmú byť `public` a klient k nim nesmie pristupovať priamo. Poslednou podmienkou je, aby trieda obsahovala atribút s anotáciou `javax.persistence.Id`. Tá určuje atribút ktorý sa stane `primary_key` v databázovej tabuľke. Voliteľnými atribútmi, ktoré sú v práci využívané, sú napríklad `javax.persistence.Column` ktorý slúži na pomenovanie atribútu v tabuľke.[14] Ako príklad je uvedená časť entitnej triedy `Typ miestnosti`:

```
@NamedQuery(name = "findAll",
query = "SELECT roomType FROM RoomType roomType")
@Entity
@XmlRootElement(name = "room_type")
public class RoomType implements Serializable {

    @Id
    private String name;

    @OneToMany(mappedBy = "group")
    private Collection<Room> rooms;
```

```
    @OneToMany(mappedBy = "roomType")
    private Collection<RoomCapacity> capacity;

    public RoomType() {
    }
    ...
```

Druhá, integračná vrstva slúži na integráciu potrebných dát. Táto vrstva obsahuje triedy ktoré sa starajú o získavanie dát z `KOS`, či už prostredníctvom `KOSapi` alebo systému `Sirius`. Patrí sem aj trieda v ktorej sa získavajú dáta z `Google Calendar`. Ďalej sa vo vrstve nachádzajú triedy starajúce sa o prácu so systémovou databázou. Každá takáto trieda obsahuje instanciu `Java` triedy `Entity Manager`, ktorá zaisťuje interakciu s perzistentným kontextom. Táto trieda obsahuje všetky nami požadované operácie nad databázou, ako `find()`, `remove()`, `persist()` a `merge()`.[17]

Tretia vrstva je business vrstva, v ktorej prebieha aplikačná logika systému. V tejto vrstve sa napríklad overuje dostupnosť požadovanej miestnosti. To znamená, že dáta získané z druhej vrstvy sa vyberú a skontroluje sa, či miestnosť v požadovaný čas už nie je obsadená. V tejto vrstve rovnako zaisťujeme bezpečnosť aplikácie. Na to sú používané anotácie `Java Security`. Z používaných je možné spomenúť hlavne `@DeclareRole()` ktorá slúži na deklaráciu rolí, ktoré môžu metódy v triede volať a `@RolesAllowed()` ktorá pre každú metódu v triede určuje, ktorá z deklarovaných rolí ju smie používať. [18] Ako príklad je uvedená časť business triedy `PersonFacade`:

```
@DeclareRoles({"facultySch", "departmSch", "normalUser"})
@Stateless
public class PersonFacade {
    @EJB
    PersonDAO personDAO;

    @RolesAllowed({"facultySch"})
    public Person getTeacherByUsername(String name) {
        return personDAO.getTeacherByUsername(name);
    }
    @PermitAll
    public Boolean isPersonKnown(String name) {
        Boolean known = personDAO.isPersonKnown(name);
        return known;
        ...
    }
```

Poslendnou vrstvou je `REST API` vrstva, ktorá spracované dáta poskytuje webovej časti. V tejto vrstve sú definované jednotlivé `REST` rozhrania, pomocou ktorých webová časť pristupuje k dátam.

3.3 Implementácia problematiky

3.3.1 Autentifikácia používateľov

Pretože jednotlivé časti aplikácie neboli spojené, bolo v testovacej webovej časti naimplementované zadávanie `username` pri rezervácií, aby bolo možné otestovať správne fungovanie kontroly používateľa. Toto zadávanie `username` má za úlohu simulovať získaný `username` od webovej časti. Zadávanie `username` je implementované len pri rezervácií, pôvodne však malo byť pri každom požiadavku webovej časti na administrátorskú časť. Keďže webová časť nebola autorovým hlavným predmetom záujmu, bolo implementované overovanie `username` len pri jednej operácií, ktorou je rezervácia miestnosti.

Skôr ako sa začne vyhodnocovať zavolanie metódy, overí sa, či je `username` evidované v `KOSapi`, pomocou príkazu `GET` na `URL KOSapi`[11], kde sa za toto `URL` pripojí reťazec `/api/3/people/username?acces_token` ktorý vráti návratový kód `200` pre úspech. V prípade že je návratový kód `200`, systém pokračuje v overovaní, či používateľ patrí pod nejakú katedru. Daná skutočnosť je overovaná požiadavkou `GET`, kde sa za `URL` pripojí `/api/3/teachers/username?acces_token`. Tu je rovnako podľa návratového kódu zistené, či daný používateľ pod nejakú katedru patrí. Týmto `GET` príkazom sa však prehľadáva vo všetkých zamestnancoch ČVUT, nie len FIT ČVUT. Môže sa teda stať, že používateľ bude v `KOS` evidovaný, ale nepatrí pod FIT ČVUT, ale pod katedru inej fakulty. To však nevadí, pretože aj keby používateľ bol v `KOS` ale nepatril pod FIT ČVUT ale pod inú fakultu, systém by jeho katedru nepoznal a priradil by mu rolu `Bežný používateľ` a jeho katedra by sa nezhodovala so žiadnou katedrou evidovanou pri `Miestnosti`.

Ak by webová časť z nejakého dôvodu povolila prístup k metódam ktoré podľa hierarchie nemajú byť používateľovi prístupné, anotácie `JavaSecurity` sa postarajú o znemožnenie zavolania metódy.

Je teda možné prehlásiť, že overovanie používateľa funguje správne, a v prípade že by bola implementácia aplikácie stihnutá, nebol by problém zavolať metódu pri každom požiadavku od webovej časti. Priraďovanie rolí používateľom nebolo implementované.

[11] `https://kosapi.fit.cvut.cz/`

3.3.2 Bezpečnosť

Ako bolo spomenuté, o bezpečnosť aplikácie sa starala technológia `Java Security`. Keďže webová a administrátorská časť neboli kvôli časovej náročnosti prepojené, autor implementoval vo svojej testovacej webovej časti `http basic authentication`.[19] Na serveri v `admin console` bola vytvorená rola bežný používateľ a rola admin, avšak s databázou rozvrhárov sa ich vzhľadom na časovú náročnosť nepodarilo prepojiť. Overovaný bol teda iba prípad, keď už by priraďovanie fungovalo, ako by sa systém pre jednotlivé role správal. Následne bolo otestované z oboch rolí použiť metódu prístupnú len pre rolu admin. Pomocou tejto technológie sa overilo, že bezpečnosť administrátorskej časti je implementovaná správne a používateľ nižšej role nemá prístup k metódam nadradenej role. V prípade, že by jednotlivé časti aplikácie boli prepojené, bezpečnosť by za splnenia ostatných podmienok, ako napríklad definovanie rolí na serveri a priradenie rolí používateľom, ktoré neboli implementované, fungovala správne.

3.3.3 Správa rezervácií

Aby boli simulované jednotlivé rezervácie, boli v testovacej webovej časti vytvorené vstupy, ktoré majú podobu jednotlivej `Rezervácie`. Teda rovnaký prípad, ako keby administrátorská časť prijala požiadavku od webovej časti od kolegov a rozdelila ju na jednotlivé `rezervácie`.

Overovanie obsadenosti miestnosti je rozdeľované na dva prípady. V prvom prípade sa jedná o miestnosť patriacu pod FIT ČVUT. Vtedy je potrebné získať dáta zo systému `Sirius`. Výstupom zo `Sirius` sú všetky výuky zapísané v `KOS` pre danú miestnosť v čase vymedzenom dátumom. Dáta sú následne spracované a sú z nich vytvorené rezervácie v stave `potvrdená`. K týmto rezerváciám je potrebné pridať rezervácie evidované iba v našej systémovej databáze, ktoré sú v stave `potvrdená`. Takto sa vytvorí kolekcia všetkých potvrdených rezervácií pre danú miestnosť v zadanom období. Následne sú porovnávané `Rezervácie` ktoré administrátorská časť prijala z požiadavku s každou `Rezerváciou` v kolekcií potvrdených rezervácií. V prípade, že by sa čas `Rezervácie` z požiadavky prekrýval s `Rezerváciou` v kolekcií, je požiadavka vyhodnotená ako `neschválená`.

Druhým prípadom je, keď požadovaná miestnosť v požiadavku na rezerváciu patrí pod FA ČVUT. Na to, aby bola získaná plánovaná výuka FA ČVUT už neslúži `KOS`, pretože aplikácií nie je umožnený prístup k dátam FA ČVUT. FA ČVUT poskytuje svoje dáta pre miestnosti využívané FIT ČVUT v službe `Google Calendar`. Spracované dáta z `Google Calendar` sú uložené do kolekcie rezervácií v stave `potvrdená`. K týmto `Rezerváciám` sú pridané, rovnako ako v predchádzajúcom prípade, `Rezervácie`, ktoré eviduje náš systém pre

danú miestnosť v zadanom čase v stave `potvrdená`. `Rezervácia` z požiadavku je porovnaná s každou `Rezerváciou` v kolekcii a v prípade, že sa rezervácie prekrývajú, je `Rezervácia` z požiadavku vyhodnotená ako `neschválená`.

Tu sa delenie na dva prípady končí a nastáva časť, od ktorej je overovanie požiadavky pre oba prípady spoločné. Ak `Rezervácia` nebola vyhodnotená ako `neschválená`, systém overí príslušnosť miestnosti ku katedre, príslušnosť rezervujúceho ku katedre a zobrazí súhrn, či `Rezervácia` prešla všetkými testami overovania.

3.3.4 Poskytovanie dát

Pre správne poskytnutie dát webovej časti bolo potrebné spolu s kolegami navrhnúť doménový model a podľa neho nadefinovať `XML` štruktúru tried, na ktorú by sa mapovali nami implementované objekty. Bol teda navrhnutý doménový model a vytvorila sa podľa neho `XML` štruktúra, avšak doménový model bol navrhnutý s menšími chybami, ako napríklad nesprávne definované dátové typy a nesprávna početnosť pri asociáciách. Keďže implementácia komunikácie medzi jednotlivými časťami aplikácie bola v harmonograme medzi poslednými úlohami, chyby v návrhu neboli odhalené včas a po dohode s vedúcim práce sme sa dohodli, že vzhľadom k časovej náročnosti sa budeme ďalej sústrediť na analýzu, návrh a iba časť implementácie.

Administrátorskú časť však autor stihol naimplementovať do bodu, kedy sa správa ako webová služba. Je teda možné zadať `GET` požiadavku na administrátorskú časť. Tá však z dôvodu, že definovaná `XML` štruktúra jednotlivých tried obsahovala chyby a z tohto dôvodu nebola použitá, funguje iba pre dve požiadavky, a to pre zobrazenie dostupnosti miestnosti a zobrazenie všetkých miestností. Pre tento prípad bola vytvorená vlastná `XML` štruktúra tried `Rezervácia` a `Miestnosť`. Konkrétna funkcionalita webovej služby je demonštrovaná na požiadavke `GET server_url/RSM_BP-war/api/events/{roomName}` ktorá nám zobrazí všetky rezervácie pre zadanú miestnosť vo formáte `XML`. Webová časť teda môže jednoducho získať potrebné dáta na zobrazenie používateľovi. Ako ukážka je priložená ukážka súboru zastupujúca instanciu rezervácie:

```
<rezervacia>
    <approved>true</approved>
    <course>MI-POA</course>
    <eventType>tutorial</eventType>
    <id>T9:3452015-03-26T11:00:00</id>
    <room>
        <blocked>false</blocked>
        <capacityT>24</capacityT>
        <division>katedra aplikované matematiky</division>
```

```
        <faculty>FIT</faculty>
        <group>
            <name>Pocitacova</name>
        </group>
        <name>T9:345</name>
    </room>
    <user>kaspaji3</user>
    <dFrom>2015-03-26T11:00:00+01:00</dFrom>
    <dTo>2015-03-26T12:30:00+01:00</dTo>
</rezervacia>
```

Týmto však chcel autor ukázať, že aplikáciu implementoval ako webovú službu a že stačí iba opraviť navrhnutú `XML` štruktúru, zabudovať ju do implementácie a poskytovanie dát bude plne funkčné.

Pre zhrnutie, implementácia administrátorskej časti je v bode, kedy získava všetky potrebné dáta zo všetkých požadovaných vstupných zdrojov, má vytvorenú vlastnú databázu pomocou technológie `JPA`, odpovedá na takmer všetky funkčné požiadavky uvedené v kapitole č. 1.2, zvláda overovane používateľov a kontrolu kolízií rezervácií a je implementovaná ako `REST` webová služba, ktorá zatiaľ z vyššie spomenutých dôvodov poskytuje `XML` výstup iba pre dva funkčné požiadavky.

Kapitola 4

Testovanie

V tejto kapitole sa autor venuje testovaniu administrátorskej časti aplikácie. Keďže implementácia aplikácie nebola dokončená a po dohode s vedúcim práce sa autor vzhľadom k náročnosti práce sústredil na analýzu, návrh a iba časť implementácie, bude v tejto kapitole popísané testovanie implementovanej časti podľa prípadov použitia uvedených v prílohe B. Okrem nižšie uvedených testov bola aplikácia priebežne testovaná počas implementácie a nedostatky boli odstránené priamo pri implementácií.

4.1 Testy podľa prípadov použitia

Pretože implementácia aplikácie nebola dokončená a jednotlivé časti aplikácie neboli spojené, autor vytvoril na vývojové a testovacie účely vlastné webové rozhranie aplikácie, ktorá simuluje požiadavky reálnej webovej časti. Výstupy sú posielané takisto na spomenuté webové rozhranie.

PP č.1: Zobraziť miestnosti

Administrátorská časť umožňuje zobraziť evidované miestnosti podľa typu, všetky miestnosti alebo konkrétnu miestnosť podľa názvu. Výstupom je objekt, poprípade `list` objektov `Miestnosť`.

PP č.2: Rezervovať miestnosť

Administrátorská časť umožňuje uložiť prijatý objekt `Rezervácia`, pričom je v systéme implementovaná metóda, ktorá kontroluje, či sa rezervácia neprekrýva s už existujúcou rezerváciou získanou z `KOS`, alebo rezerváciou uloženou v našej databáze. Metóda takisto vracia katedru miestnosti a katedru používateľa, avšak automatické schvaľovanie a posielanie rozvrhárom podľa katedier implementované nie je.

PP č.3: Pridanie miestnosti

Administrátorská časť na základe prijatých atribútov vytvorí objekt `Miestnosť` a uloží ho v systémovej databáze.

PP č.4: Zablokovanie miestnosti

Administrátor využije PP1 uvedený v prílohe B na zobrazenie miestnosti a bude mu umožnené zablokovať alebo odblokovať požadovanú miestnosť.

PP č.5: Úprava miestnosti

Pre splnenie tohto prípadu je potrebné najprv využiť PP1 uvedený v prílohe B. Následne administrátorská časť umožňuje zmeniť všetky evidované atribúty miestnosti pri objekte `Miestnosť`. Vstupom je objekt `Miestnosť`.

PP č.6: Zrušenie rezervácie

Administrátorská časť umožňuje zrušenie rezervácie. Na splnenie tohto prípadu môže používateľ využiť PP9 uvedený v prílohe B. Rozvrhár si môže zobraziť rezervácie tromi spôsobmi:

- Rezervácie na potvrdenie
- Rezervácie podľa miestnosti
- Rezervácie podľa používateľa

PP č.7: Úprava rezervácie

Úprava rezervácie nebola priamo implementovaná, avšak iba z dôvodu že webová a administrátorská časť aplikácie nie sú spojené. Pri prijatí požiadavku na zmenu rezervácie by administrátorská časť jednotlivú rezerváciu vymazala a znovu vytvorila s upravenými parametrami, teda použila by funkcie, ktoré systém už má implementované.

PP č.8: Schválenie/Neschválenie rezervácie

Administrátorská časť umožňuje schválenie a neschválenie jednotlivých rezervácií. Rozvrhár si môže zobraziť rezervácie tromi spôsobmi:

- Rezervácie na potvrdenie
- Rezervácie podľa miestnosti
- Rezervácie podľa používateľa

PP č.9: Zobrazenie svojich rezervácií

Administrátorská časť umožňuje zobraziť rezervácie podľa `username` používateľa, čo by bolo využité na splnenie tohto požiadavku. Výstupom je `list` objektov `Rezervácia`.

PP č.10: Zobrazenie používateľov

Administrátorská časť získava dáta o používateľoch zo systému `KOS` a dokáže zobraziť všetky pre aplikáciu zaujímavé dáta o používateľovi, ako je `username`, meno, priezvisko, e-mail a katedra. Výstupom je objekt `Používateľ`.

PP č.11: Blokovanie používateľov

Implementácia blokovania používateľov bola po dohode s vedúcim práce dočasne vyradená z funkčných požiadaviek na systém.

PP č.12: Úprava práv používateľov

Administrátorská časť aplikácie umožňuje pridať, prípadne odobrať právo katedrový rozvrhár pre používateľa. Vstupom je objekt `Používateľ`.

PP č.13: Zobraziť dostupnosť

Administrátorská časť umožňuje zobraziť jednotlivé výuky pre danú miestnosť vo vymedzenom čase. Výstupom je `list` objektov `Rezervácia` pochádzajúce zo systémovej databázy a informačného systému `KOS`.

4.2 Testy podľa scenárov

Pre testovacie účely boli vymyslené scenáre, pri ktorých hrozí, že by sa systém mohol správať neobvykle, mohol vyhodiť chybu, alebo spadnúť. Tieto scenáre nepokrývajú celú funkcionalitu implementovanej časti.

4.2.1 Scenáre testovania

Scenár č.1 – Zadanie nesprávneho názvu miestnosti

Tento scenár simuluje situáciu, kedy administrátorská časť prijme od webovej časti požiadavku na zobrazenie neexistujúcej miestnosti, prípadne prijme názov miestnosti zo syntaktickou chybou.

Scenár č.2 – Vloženie miestnosti s rovnakým názvom ale rôznymi parametrami

Tento scenár simuluje situáciu, kedy administrátorská časť prijme od webovej časti požiadavku na vloženie miestnosti, ktorá je už v systéme evidovaná, avšak tentokrát ju ukladáme s inými parametrami.

Scenár č.3 – Zadanie nesprávneho username

Tento scenár simuluje situácie, kedy administrátorská časť prijme od webovej časti požiadavku na vytvorenie katedrového rozvrhára s takým `username`, ktoré nie je v `KOS` evidované.

Scenár č.4 – Pridanie identickej rezervácie dvoma používateľmi

Tento scenár simuluje situácie, kedy administrátorská časť prijme od webovej časti požiadavku na vytvorenie rezervácie, avšak identická rezervácia sa už v systéme nachádza a bola prijatá od iného používateľa.

4.2.2 Výsledky scenárov

Výsledok scenáru č.1 – Zadanie nesprávneho názvu miestnosti

Miestnosti sú uložené v systémovej databáze, preto sa systém štandardne chová tak, že ak v databáze nenájde požadovanú hodnotu, vráti `null`.

Výsledok Systém sa chová správne.

Výsledok scenáru č.2 – Vloženie miestnosti s rovnakým názvom ale rôznymi parametrami

Pretože miestnosti sú ukladané v databáze systému a `Miestnosť` má ako identifikátor nastavený svoj názov ktorý je unikátny, systém pri vložení miestnosti ktorá už v systéme existuje vyhodí

```
javax.ejb.EJBEcteption: Transaction aborted
```

Výsledok Vďaka scenáru bola zistená závažná chyba ktorá mohla spôsobiť znefunkčnenie systému.

Odporučenie Overiť, či sa prvok v databáze už nenachádza a namiesto metódy `persist()` použiť metódu `merge()`.

Výsledok scenáru č.3 – Zadanie nesprávneho username

Používatelia nie sú uložený v žiadnej databáze, preto sú dostávaní zo systému `KOS`. Pri tom sa vytvára objekt `Používateľ`. Keďže však aplikácia na základe nesprávneho `username` nenašla žiadneho používateľa, systém sa ho pokúsil vytvoriť a uložiť do databázy rozvrhárov čo má za následok

```
javax.ejb.EJBTransactionRolledbackException
```

Výsledok Vďaka scenáru bola zistená závažná chyba, ktorá spôsobovala pri zle prijatom `username` neželané chovanie systému.

Odporučenie Overiť návratovú hodnotu `GET` príkazu do `KOS` a pokračovať v procese iba v prípade úspešného nájdenia používateľa v `KOS`.

Výsledok scenáru č.4 – Pridanie identickej rezervácie dvoma používateľmi

Kontrolovanie rezervácií bolo implementované tak, aby čas začiatku a čas konca novej rezervácie nemohol byť v už existujúcej rezervácií. Pri vývoji však nebol kontrolovaný prípad pre krajné hodnoty, ktorými sú práve začiatočný a koncový bod rezervácie.

Výsledok Pretože systém kontroloval iba vnútorný interval, ukladal preto do databázy identickú rezerváciu pre dvoch používateľov.

Odporučenie Pridať do kontroly intervalu medzné hodnoty časov rezervácie.

Ako bolo spomenuté, implementácia aplikácie nebola dokončená a teda bola testovaná len časť aplikácie. Aplikácia bola testovaná aj priebežne popri implementácií a nájdené chyby boli následne opravené. Pri testovaní implementovanej časti aplikácie bolo odhalených niekoľko chýb, ktoré boli z implementácie odstránené. Časť chýb, ktoré aplikácia momentálne má, sú z dôvodu že implementácia nebola stihnutá. Teda v prípade, že by aplikácia bola dokončená, by sa tam dané chyby nevyskytovali a nebolo by ich potrebné nijak riešiť.

Kapitola 5

Nasadenie aplikácie

Táto časť sa venuje nasadeniu aplikácie, teda spísaniu softwarových požiadaviek, konfigurácií a následnej inštalácií aplikácie. Nasadenie aplikácie graficky popisuje diagram nasadenia.

5.1 Softwarové požiadavky

- ľubovoľný operačný systém s podporou `Java`
- `Java EE 7`
- aplikačný server (doporučený`Glassfish 4.1`)
- relačný databázový stroj (doporučený `PostgreSQL`)

5.2 Konfigurácia

Pred tým, ako môže byť aplikácia nasadená na server, je potrebné na serveri vytvoriť databázu. Po vytvorení databázy je nutné nakonfigurovať `Connection pool`. Pri jeho vytváraní je potrebné

resource type nastaviť na `javax.sql.ConnectionPoolDataSource`

databaseName meno vytvorenej databázy

user používateľské meno pre prístup do databázy

password používateľské heslo pre prístup do databázy

serverName názov serveru

portNumber číslo portu

driverClass trieda ovládača k zvolenej databáze

Posledným krokom je vytvorenie `Resource`, v ktorom sa iba odkáže na vytvorený `Connection Pool`. V konfiguračnom súbore `persistence.xml` sa upraví parameter `jta-data-source` na meno vytvoreného `Resource`.

5.3 Inštalácia aplikácie

Na záver stačí nahrať skompilovaný `ear` súbor do servlet kontajneru aplikačného serveru.

Kapitola 6

Ekonomicko-manažérske prínosy aplikácie

Cieľom tejto kapitoly je posúdenie prínosov projektu Rezervačný systém miestností po ekonomickej a manažérskej stránke, ktorý je pripravovaný FIT ČVUT formou bakalárskej práce. V ekonomickej časti sa autor bude venovať predovšetkým pohľadu z finančnej stránky a porovnaniu s inými možnosťami rezervácie miestností. V manažérskej časti bude uvedené, ako implementovaná aplikácia ovplyvní jednotlivých používateľov z pohľadu úspory času.

Manažérska časť

Keďže súčasný stav rezervácií je pre všetkých účastníkov rezervácie systému obmedzujúci, predovšetkým čo sa týka času, fakulta sa rozhodla vytvoriť systém, ktorý by riešil hlavné problémy doterajšieho systému rezervovania. Najväčším problémom súčasného systému je komunikácia medzi žiadateľmi o rezerváciu a rozvrhármi, ktorá prebieha formou e-mailu. Výsledná aplikácia by ovplyvňovala obe strany používateľov nasledujúcim spôsobom

Žiadatelia o rezerváciu by sa už nestretávali s problémom, že musia kontaktovať rozvrhára na to, aby zistili kedy je miestnosť voľná alebo obsadená. V aplikácií by totiž videli všetky rezervácie ihneď, aj tie, ktoré nie sú zapísané v KOS. Tým sa výrazne urýchli proces výberu správneho dátumu a času miestnosti. Keďže aplikácia pracuje podľa určitej hierarchie používateľov, nestalo by sa, že si používateľ rezervuje miestnosť u iného rozvhára, čo má znovu za následok urýchlenie času rezervácie. V prípade, že by používateľ patril rovnakej katedre ako miestnosť, poprípade chcel rezervovať miestnosť pre predmet rovnakej katedry ako je katedra miestnosti, bola by mu rezervácia ihneď vyhodnotená bez ďalšieho súhlasu rozvrhára, čo v doterajšom spôsobe rezervácií možné nebolo. Rovnako by sa rezervácie rozšírili pre väčšiu skupinu ľudí, pretože veľa študentov o možnosti rezervovať si miestnosť ani netušili.

Pre používateľov bude ďalej evidovaný prehľad ich rezervácií, teda v prípade, že by používateľ zabudol kedy má rezerváciu vytvorenú, jednoducho si to skontroluje v systéme. Tieto všetky aspekty predstavujú obrovskú úsporu času pre žiadateľov o rezerváciu.

Rozvrhárom bude uľahčená práca predovšetkým preto, že im nebudú chodiť e-maily, ktoré sú určené pre iných rozvrhárov. Pre rozvrhárov bude takisto urobený prehľad žiadostí o rezerváciu, ktoré majú schváliť, preto sa nebude stávať, že sa rozvrhárom stratí žiadosť v e-mailovej schránke. Výraznou úsporou času bude aj fakt, že nebudú musieť žiadateľom oznamovať obsadenosť miestností. Komunikácia medzi žiadateľom a rozvrhárom sa teda obmedzí na maximálne možné minimum. Vďaka týmto skutočnostiam budú rozvrhári môcť využívať svoj čas oveľa efektívnejšie.

Ekonomická časť

Súčasný stav rezervácie miestností sa FIT ČVUT rozhodla riešiť použitím rezervačného systému miestností. Pre získanie takéhoto systému mala fakulta na výber z viacerých možností. Pre každú možnosť zhodnotíme jej výhody, zápory a riziká.

Prvou možnosťou je zakúpiť už existujúci rezervačný systém. Keďže je už známe, že fakulta potrebuje pre svoje potreby variabilný systém ktorý eviduje rôzne role používateľov, rôzne typy rezervovaných miestností a pre aktualitu musí získavať dáta zo školských systémov, je takmer isté, že by bolo potrebné zakúpený rezervačný systém dodatočne upraviť. Tento typ systémov je po väčšine platený paušálne firme ktorá ho prevádzkuje, k týmto nákladom by však FIT ČVUT naviac potrebovala systém upraviť podľa svojich požiadaviek, čím by sa nákup systému ešte viac predražil. Následne by bolo potrebné zaškolenie používateľov alebo aspoň správcu systému, pretože pôvodný návod aplikácie by bol neaktuálny z dôvodu zmien vytvorených na aplikácií na žiadosť FIT ČVUT, ktoré by rovnako stálo fakultu ďalšie financie.

Druhou možnosťou je použiť voľne dostupný rezervačný systém. Je vysoko pravdepodobné, že keby sa použije voľne dostupný systém, je ho rovnako potrebné upraviť podľa požiadaviek FIT ČVUT. Získanie systému by teda bolo bezplatné, avšak je otázne, koľko času a financií by stálo za prerobením takéhoto systému na fakultou požadovaný systém. Na to aby fakulta takýto systém prerobila, musela by na prestavbu systému niekoho poveriť, čo by fakultu stálo čas aj financie. Tento systém by bolo potrebné najprv analyzovať, zistiť aké technológie používa a nadviazať na prácu niekoho iného, ktorý základ systému vytvoril, čo býva vo veľa prípadoch veľmi namáhavé.

Treťou možnosťou je vytvoriť si vlastný systém. Táto možnosť sa javí ako najvýhodnejšia pre problém rezervácie miestností. Fakulta by tým ušetrila financie, ktoré by musela vynaložiť pri prvom návrhu riešenia. Čas na analýzu voľne dostupnej aplikácie, spomenutý v druhom návrhu riešenia, môže fakulta využiť na podrobnú analýzu a návrh svojho problému. Navyše by fakulta mohla použiť technológie podľa vlastnej voľby a nepotrebovala by nadväzovať na žiadnu prácu začatú niekým iným, čo by vo výsledku prinieslo menšie riziko že sa v systéme vyskytnú chyby.

Náklady na chod aplikácie

Aplikácia však prinesie zo sebou väčšie finančné náklady na chod systému ako doterajší spôsob rezervácií. Ten časovo vyťažoval predovšetkým účastníkov procesu rezervácie. Aplikácia by tento problém vyriešila, avšak za cenu zvýšenia finančných nákladov. Jednalo by sa hlavne o technické zázemie, teda server a jeho prevoz. Keďže sa však jedná o inštitúciu, ktorá sa zameriava na informačné technológie, predpokladá sa že zaistenie technického zázemia nebude žiadnym problémom.

Náklady na zaobstaranie

Náklady na zaobstaranie aplikácie predstavujú predovšetkým časovú náročnosť. Keďže je však aplikácia vyvíjaná formou bakalárskej práce, nepredstavuje vývoj aplikácie žiadnu prácu navyše pre fakultu.

V konečnom dôsledku sa teda vývoj aplikácie javí ako výrazná úspora času pre obe strany procesu rezervácií pričom pre fakultu predstavuje iba minimálne náklady na zaobstaranie a chod aplikácie. Pretože je aplikácia vyvíjaná formou bakalárskej práce, nepredstavuje pre fakultu nijakú prácu navyše.

Záver

Cieľom tejto práce bolo analyzovať, navrhnúť a implementovať administrátorskú časť rezervačného systému učební FIT ČVUT, ktorá mala byť použiteľná ako vstup pre súčasne zadanú bakalársku prácu, Webová časť rezervačného systému učební FIT ČVUT a zároveň pre bakalársku prácu Uživatelské rozhraní rezervačního systému učeben FIT.

V časti venovanej analýze boli predstavené požiadavky na aplikáciu vychádzajúce zo súčasného spôsobu rezervácie miestností. Požiadavky na aplikáciu boli stavané spôsobom, aby ostala zachovaná doteraz používaná hierarchia používateľov a bol braný ohľad na rôzne typy miestností, teda aby bolo aplikáciu jednoduché nasadiť v rámci infraštruktúry FIT ČVUT.

V časti venovanej návrhu aplikácie bola navrhnutá architektúra systému, uvedená aplikačná logika, voľba technológií a návrh komunikácie medzi časťami aplikácie.

Práca bola implementovaná ako webová služba na platforme Java EE. Bol kladený dôraz na získavanie dát z externých zdrojov a vytvorenie systémovej databáze, vďaka čomu aplikácia získala potrebné dátové zázemie. Dáta boli následne spracované a vývoj pokračoval implementovaním jednotlivých problémov vychádzajúcich z funkčných požiadaviek. Na základe rozhodnutia vedúceho práce bola implementácia aplikácie vzhľadom k rozsahu a časovej náročnosti pozastavená. Implementácia však bola stihnutá do bodu, kedy administrátorská časť spolupracuje so všetkými externými dátovými zdrojmi, obsahuje vlastnú databázu a odpovedá na väčšinu funkčných požiadaviek. Požiadavka na sprístupnenie administrátorskej časti ako webovej služby bola splnená, poskytovanie dát však bolo stihnuté len pre pár funkčných požiadaviek.

Implementovaná časť bola otestovaná podľa prípadov použitia a vytvorených testovacích scenárov. V časti venovanej inštalácií je dôkladne zdokumentovaný proces inštalácie tak, aby bolo možné aplikáciu jednoducho nasadiť v rámci infraštruktúry FIT ČVUT.

V práci sú uvedené ekonomicko-manažérske prínosy práce pre FIT ČVUT, v ktorých autor zhodnotil časové a finančné výhody a nevýhody plynúce z používania implementovaného systému.

Pre mňa ako autora práce sú najväčším prínosom určite získané znalosti technológií, s ktorými som sa predtým vôbec nestretol. Účasťou na projekte som rozvinul svoje znalosti v oblasti analýzy, návrhu a realizácie projektu daného rozsahu. Aj napriek tomu, že práca nebola implementovaná do finálneho štádia si myslím, že hotová časť aplikácie slúži ako veľmi dobrý základ na vybudovanie požadovanej aplikácie. Keďže ma práca na aplikácií zaujala, veľmi rád by som sa podieľal na jej dokončení a finálnom nasadení.

Literatúra

[1] FIT ČVUT: KOS [online]. [cit. 2015-05-06]. Dostupné z: http://intranet.cvut.cz/informace-pro-studenty/is/kos

[2] Rouse, M.: REST (representational state transfer) [online]. [cit. 2015-05-06]. Dostupné z: http://searchsoa.techtarget.com/definition/REST

[3] FIT ČVUT: KOSapi [online]. [cit. 2015-05-06]. Dostupné z: https://kosapi.fit.cvut.cz/projects/kosapi/wiki

[4] Refsnes Data: XML Tutorial [online]. [cit. 2015-05-06]. Dostupné z: http://www.w3schools.com/xml/

[5] Jakub, J.: KOS API jako webová služba [online]. [cit. 2015-05-06]. Dostupné z: https://dip.felk.cvut.cz/browse/pdfcache/jirutjak_2010bach.pdf

[6] Google Inc.: Google Calendar [online]. [cit. 2015-05-06]. Dostupné z: http://learn.googleapps.com/calendar

[7] Google Inc.: Google Calendar API [online]. [cit. 2015-05-06]. Dostupné z: https://developers.google.com/google-apps/calendar/

[8] Refsnes Data: JSON Tutorial [online]. [cit. 2015-05-06]. Dostupné z: http://www.w3schools.com/json/json_eval.asp

[9] Wikipedia: Hypertext Transfer Protocol [online]. [cit. 2015-05-06]. Dostupné z: http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocoldocs.oracle.com/javaee/6/tutorial/doc/bnbpz.html

[10] Rouse, M.: SOAP (Simple Object Access Protocol) Definition [online]. [cit. 2015-05-06]. Dostupné z: http://searchsoa.techtarget.com/definition/SOAP

[11] Ort, E.: Java Architecture for XML Binding (JAXB) [online]. [cit. 2015-05-06]. Dostupné z: http://www.oracle.com/technetwork/articles/javase/index-140168.html

[12] Oracle Inc.: GlassFish Server Open Source Edition Quick Start Guide [online]. [cit. 2015-05-06]. Dostupné z: https://glassfish.java.net/docs/4.0/quick-start-guide.pdf

[13] PostgreSQL: PostgreSQL 9.4.1 Documentation [online]. [cit. 2015-05-06]. Dostupné z: http://www.postgresql.org/docs/9.4/interactive/index.html

[14] Oracle Inc.: Introduction to the Java Persistence API [online]. [cit. 2015-05-06]. Dostupné z: http://docs.oracle.com/javaee/6/tutorial/doc/bnbpz.html

[15] FIT ČVUT: APPS MANAGER [online]. [cit. 2015-05-06]. Dostupné z: https://auth.fit.cvut.cz/manager/user/home.jsf

[16] Google Inc.: Google Identity Platform [online]. [cit. 2015-05-06]. Dostupné z: https://developers.google.com/identity/protocols/OAuth2WebServer

[17] Oracle Inc.: Interface EntityManager [online]. [cit. 2015-05-06]. Dostupné z: http://docs.oracle.com/javaee/6/api/javax/persistence/EntityManager.html

[18] Oracle Inc.: Introduction to Security in the Java EE Platform [online]. [cit. 2015-05-06]. Dostupné z: http://docs.oracle.com/javaee/6/tutorial/doc/bnbqa.html

[19] Simtec Limited: HTTP Authentication [online]. [cit. 2015-05-06]. Dostupné z: http://www.httpwatch.com/httpgallery/authentication/

Dodatok **A**

Zoznam použitých skratiek

CRUD operácie Create, Read, Update a Delete operácie

REST Representational State Transfer

RESTful Využívajúci systém REST

SOAP Simple Object Access Protocol

Cache pamäť Rýchla vyrovnávacia pamäť

HTTP Hypertext Transfer Protocol

XPartial Filter umožňujúci selektovať výstup GET príkazu

XPath Jazyk na selektovanie jednotlivých častí XML súboru

XML Extensible markup language

Dodatok B

Zoznam všetkých prípadov použitia

PP č.1 Zobraziť miestnosti

Systém umožňuje používateľovi zobraziť akúkoľvek miestnosť. Pre prehľadnosť si môže používateľ vybrať miestnosť podľa typu.

Zobrazenie podľa názvu. Používateľ vyberie zo zoznamu názov miestnosti. Systém vypíše základné informácie o vybranej miestnosti, ako napríklad: kapacita, typ a katedra.

Zobrazenie podľa typu. Používateľ si vyberie z ponúknutých typov miestnosti. Systém vypíše zoznam miestností vyhovujúcich vybranému typu. Po tom čo si používateľ vyberie miestnosť zo zoznamu. Systém o vybranej miestnosti vypíše základné informácie, ako napríklad kapacita, typ a katedra.

Ak si miestnosti zobrazuje fakultný rozvrhár, vidí nie len kapacitu, typ a katedru ale aj kedy je daná miestnosť rezervovaná.

PP č.2 Rezervovať miestnosť

Používateľ sa autentifikuje na webe, vyhľadá si miestnosť podľa prípadu použitia č.13v kapitole č. B a v prípade záujmu o rezerváciu zadá požiadavku na systém. Systém následne skontroluje či je táto požiadavka splniteľná. To znamená, že musí skontrolovať či je miestnosť v danom čase voľná alebo či používateľ patrí pod FIT. Miestnosti sa rezervujú buď jednorázovo, na dátumom určené obdobie alebo na jeho periódu (napr. každý pondelok). Za predpokladu, že si miestnosť chce rezervovať, môžu nastať nasledujúce prípady:

- **Miestnosť je katedrová**

- **Používateľ patrí rovnakej katedre ako miestnosť**
  Systém automaticky rezervuje miestnosť bez ďalšieho súhlasu rozvrhára a informuje používateľa.
- **Používateľ nepatrí rovnakej katedre ako miestnosť**
  Odoslanie žiadosti o potvrdenie rezervácie katedrovému rozvrhárovi katedry ktorej patrí daná miestnosť. Kým žiadosť nie je potvrdená používateľ ju môže zrušiť. Ak ju rozvrhár schváli, zapíše ju do systémuKOS. Systém toto rozhodnutie oznámi používateľovi.

- **Miestnosť nie je katedrová**
  - **Miestnosť je celofakultná**
    - **Používateľ vytvára rezerváciu pre predmet**
      Odoslanie žiadosti o potvrdenie rezervácie katedrovému rozvrhárovi katedry o ktorej predmet sa jedná. Kým žiadosť nie je potvrdená používateľ ju môže zrušiť. Ak ju rozvrhár schváli, zapíše ju do systému KOS. Systém toto rozhodnutie oznámi používateľovi.
    - **Používateľ vytvára rezerváciu pre seba**
      - **Používateľ patrí katedre**
        Odoslanie žiadosti o potvrdenie rezervácie katedrovému rozvrhárovi, ktorej je používateľ členom. Kým žiadosť nie je potvrdená používateľ ju môže zrušiť. Ak ju rozvrhár schváli, zapíše ju do systému KOS. Systém toto rozhodnutie oznámi používateľovi.
      - **Používateľ nepatrí katedre**
        Odoslanie žiadosti o potvrdenie rezervácie fakultnému rozvrhárovi. Kým žiadosť nie je potvrdená používateľ ju môže zrušiť. Ak ju rozvrhár schváli, zapíše ju do systému KOS. Systém toto rozhodnutie oznámi používateľovi.
  - **Miestnosť je mimofakultná**Odoslanie žiadosti o potvrdenie rezervácie fakultnému rozvrhárovi. Kým žiadosť nie je potvrdená používateľ ju môže zrušiť. Ak ju rozvrhár schváli, zapíše ju do systému KOS. Systém toto rozhodnutie oznámi používateľovi.

Proces rezervovania miestností uzatvára prípad použitia č.8 B, kde sú rozpísané prípady, kedy treba rezervovanie miestnosti zapísať aj do iného systému.

PP č.3 Pridanie miestnosti

Umožnuje fakultnému rozvrhárovi pridať miestnosť do systému. V prípade že FIT ČVUT získa nové priestory, fakultný rozvrhár bude mať možnosť pri-

dať nové miestnosti do systému. Po prihlásení do systému vyberie možnosť `pridať miestnosť` a vyplní nasledujúce parametre miestnosti:

- názov
- kapacita
- typ
- katedra
- fakulta

Miestnosť bude následne možné vyhľadať a rezervovať pre používateľov.

PP č.4 Zablokovanie miestnosti

Rozvrhár po autentifikácií na webe využije prípad použitia č.1 B, pomocou ktorého si zobrazí miestnosť. Následne zvolí možnosť `Upraviť` alebo `Blokovať`. Fakultný rozvrhár môže blokovať akúkoľvek miestnosť, katedrový rozvrhár len miestnosti svojej katedry. Pri vyhľadaní miestnosti bude pri miestnosti zobrazená informácia o zablokovaní miestnosti.

PP č.5 Úprava miestnosti

Fakultný rozvrhár si po autentifikácií na webe zobrazí miestnosť pomocou prípadu použitia č.1 B. V príslušnej miestnosti zvolí možnosť upraviť miestnosť. Atribúty evidované u entity miestnosť sú: názov miestnosti, kapacita, príslušnosť ku katedre, príslušnosť k fakulte a typ miestnosti. Názov miestnosti je identifikátor, teda pri prípadnej zmene názvu miestnosti systém overí či ostala zachovaná unikátnosť. Pri type miestnosti bude upravujúcemu ponúknuté na výber z evidovaných typov miestností. V prípade že požadujúci typ nie je evidovaný, bude fakultnému rozvrhárovi povolené vytvoriť nový typ miestností. Každý typ miestností má pre seba špecifikované kapacity, ktoré u daného typu existujú. Výber katedry a fakulty bude rozvrhárovi ponúknutý z evidovaných fakúlt a katedier.

PP č.6 Zrušenie rezervácie

Fakultný rozvrhár môže kedykoľvek zrušiť akúkoľvek rezerváciu. K rezervácii sa dostane dvoma spôsobmi.

1. Zobraziť všetkých používateľov a rušiť rezervácie používateľa (použije UC 10 B) Vyberie si konkrétneho používateľa, zobrazí jeho rezervácie. Tie sú zobrazené v liste pod sebou a pri každej je tlačítko zrušiť.

2. Vyhľadať miestnosť (použije UC 1 B) Fakultný rozvrhár po rozkliknutí miestnosti vidí všetky rezervácie, ktoré sú s touto miestnosťou spojené. Tie sú zobrazené v liste pod sebou a pri každej je tlačítko zrušiť.

Možnosť rušiť cudzie rezervácie má len rozvrhár. Rezervácie vytvorené fakultným rozvrhárom môže rušiť iba on sám. Tieto rezervácie musí navyše manuálne rušiť v systéme `KOS`. Katedrový rozvrhár môže kedykoľvek zrušiť rezerváciu podla miestnosti a predmetu patriacich pod jeho katedru. Zrušenie rezervácie musia rozvrhári spraviť v systéme `KOS`. Používateľ môže kedykoľvek zrušiť svoju rezerváciu, avšak rozvrhár to musí vyhodiť zo systému `KOS`. Vo všetkých prípadoch príde používateľovi informácia o zrušení rezervácie.

PP č.7 Úprava rezervácie

Umožňuje používateľovi upraviť všetky údaje ktoré sú u entity rezervácia evidované. Úprava v sebe zahrňuje správu rezervácie ako:

- zmena času
- dátumu
- dôvod rezervácie atď.

Možnosť úpravy taktiež záleží na stave vytvorenej rezervácie.

Úprava rezervácie čakajúcej na schválenie Umožňuje upraviť rezerváciu používateľovi, ktorý ju vytvoril, až kým nie je rezervácia v stave `schválená` alebo `neschválená`.

Úprava schválenej rezervácie úprava možná len rozvrhárom katedry/fakulty.

PP č.8 Schválenie/Neschválenie rezervácie

Funkcia schválenia alebo neschválenia rezervácie pre rozvrhára, ktorú môžme rozdeliť do nasledujúcich prípadov.

- **Miestnosť je katedrová**
  - **Používateľ patrí rovnakej katedre ako miestnosť**
    Systém automaticky rezervuje miestnosť bez dalšieho súhlasu rozvrhára alebo zapisovania do iného systému a informuje používatela.

  - **Používateľ nepatrí rovnakej katedre ako miestnosť**
    Žiadosť o rezerváciu schvaľuje rozvrhár katedry, pod ktorú patrí miestnosť, s tým že musí rezervovanú miestnosť zapísať do systému `KOS`.

- **Miestnosť nie je katedrová**
  - **Miestnosť je celofakultná**
    * **Používateľ vytvára rezerváciu pre predmet**
      Žiadosť o rezerváciu schvaľuje katedrový rozvrhár katedry, o ktorej predmet sa jedná, s tým že musí rezervovanú miestnosť zapísať do systému `KOS`.
    * **Používateľ vytvára rezerváciu pre seba**
      · **Používateľ patrí katedre**
      Žiadosť o rezerváciu schvaľuje katedrový rozvrhár katedry, o ktorej je používateľ členom, s tým že musí rezervovanú miestnosť zapísať do systému `KOS`.
      · **Používateľ nepatrí katedre**
      Žiadosť o rezerváciu schvaľuje fakultný rozvrhár s tým že musí rezervovanú miestnosť zapísať do systému `KOS`.
  - **Miestnosť je mimofakultná**
    Žiadosť o rezerváciu schvaľuje fakultný rozvrhár s tým že musí rezervovanú miestnosť zapísať do systému `KOS`.

PP č.9 Zobrazenie svojich rezervácií

Používateľ si po prihlásení do systému môže pozrieť svoje schválené a neschválené rezervácie.

Používateľ si z hlavného menu vyberie stránku `Zobraziť moje rezervácie`. Systém mu vypíše zoznam všetkých rezervácií. To či je rezervácia `schválená`, `zamietnutá` alebo ešte `čaká na schválenie` rozvrhárom systém odlíši graficky.

PP č.10 Zobrazenie používateľov

Fakultný rozvrhár bude mať po prihlásení do systému možnosť zobraziť si všetkých používateľov systému, ich profily a aktuálne rezervácie.

Fakultný rozvrhár si z hlavného menu vyberie stránku `Zobraziť všetkých používateľov`. Systém vypíše zoznam všetkých používateľov, po vybratí konkrétneho používateľa vidí fakultný rozvrhár kedy a aké miestnosti má používateľ aktuálne rezervované.

PP č.11 Blokovanie používateľov

Fakultný rozvrhár môže blokovať všetkých používateľov. Katedrový rozvrhár môže blokovať používateľov patriacich pod jeho katedru.

Rozvrhár využije prípad použitia č 10 B, Zobrazenie používateľov. Pri zobrazení konkrétneho používateľa mu systém ponúkne možnosť blokovať používateľa. Akonáhle rozvrhár túto možnosť zvolí, blokovaný používateľ stráca možnosť rezervovať si miestnosť až do kým ho fakultný rozvrhár neodblokuje. Pri zablokovaných používateľoch systém fakultnému rozvrhárovi ponúka možnosť odblokovať.

PP č.12 Úprava práv používateľov

Fakultný rozvrhár sa pri príchode na web autentifikuje. Zobrazí si používateľa podľa identifikátoru. Následne bude fakultnému rozvrhárovi ponúknutá možnosť pridania alebo odobratia funkcie fakultný rozvrhár pre danú osobu.

PP č.13 Zobraziť dostupnosť miestností

Umožňuje používateľovi vyhľadať, ktoré miestnosti sú v danom období voľné. Používateľ si buď vyberá konkrétnu miestnosť, alebo si pri vyberaní pomôže zobrazením miestností podľa typu.

Pri každej miestnosti systém zobrazí parametre miestnosti, teda názov kapacitu a katedru. Tieto údaje pomôžu používateľovi lepšie si zo zoznamu vybrať, ktoré miestnosti sú pre neho zaujímavé. Po označení miestnosti používateľom systém vyhľadá a graficky vyznačí časy, v ktorých sú tieto miestnosti dostupné a obsadené v rámci týždňa. Používateľ teda presne vie, v ktorých časoch si môže miestnosti rezervovať.

Dodatok C

Obrázky

V tejto kapitole je uvedený zoznam obrázkov ktoré istým spôsobom súvisia s touto bakalárskou prácou. Je však potrebné upozorniť, že nasledujúce obrázky nie sú dielom autora, ale dielom kolegov pracujúcich na webovej časti aplikácie.

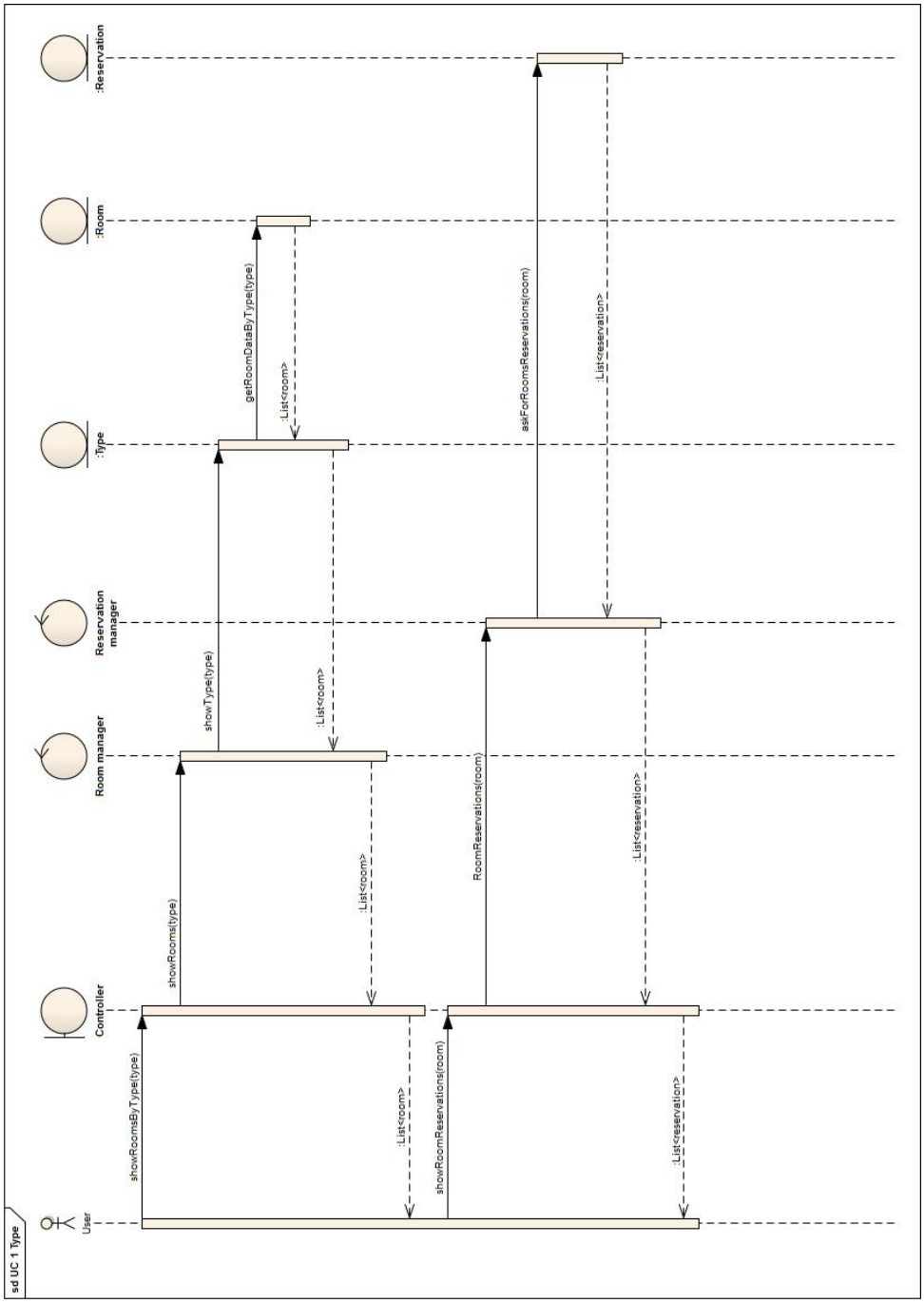
Sekvenčné diagramy

Ku každému so sekvenčných diagramov je uvedené REST rozrhanie.

Zobrazenie miestnosti

```
GET url_webovej_časti/api/rooms
GET url_webovej_časti/api/rooms?roomType
GET url_webovej_časti/api/rooms?roomName
```

Vrátené dáta sú objekt `miestnosť`.



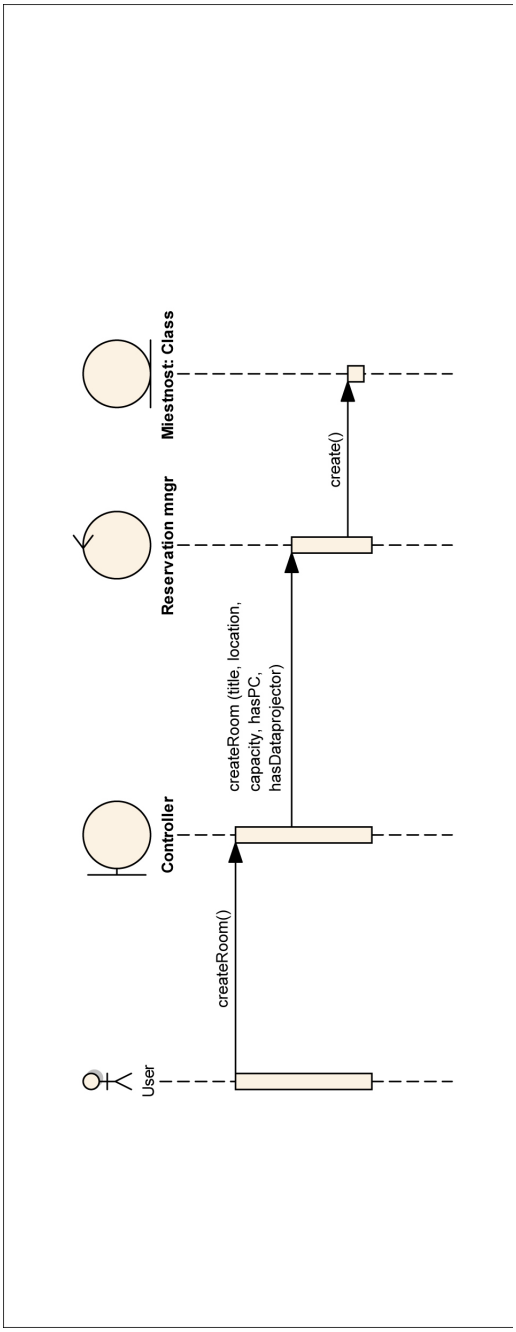
Obr. C.1: Sekvenčný diagram pre prípad použitia č.1 zobrazenie miestnosti

.

Pridanie miestnosti

`POST url_webovej_časti/api/rooms`

Posielané dáta sú objekt `miestnosť`.

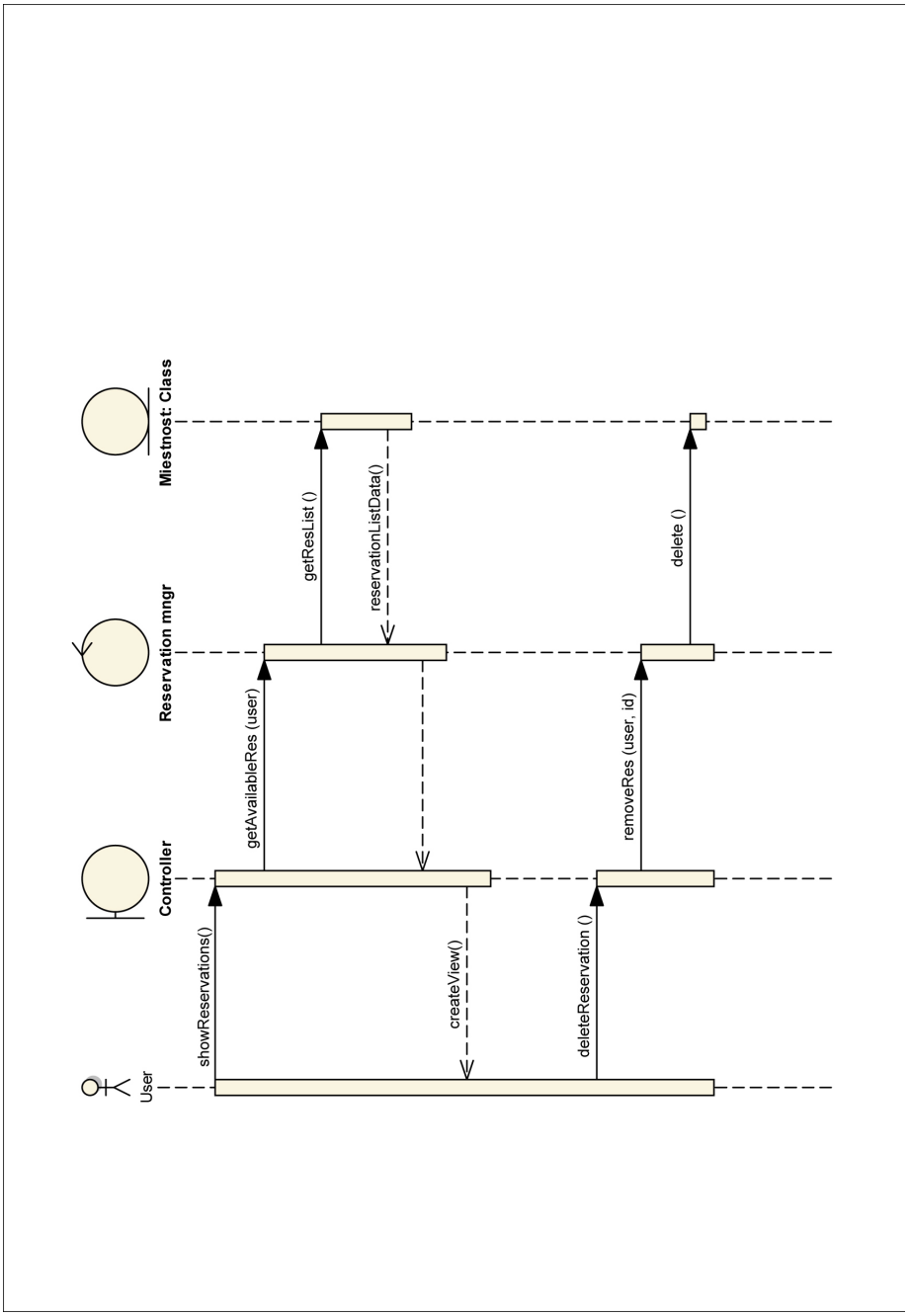


Obr. C.2: Sekvenčný diagram pre prípad použitia č.3 pridanie miestnosti

Zrušenie rezervácie

`DELETE url_webovej_časti/api/events`

Posielané dáta sú objekt `rezervácia`.

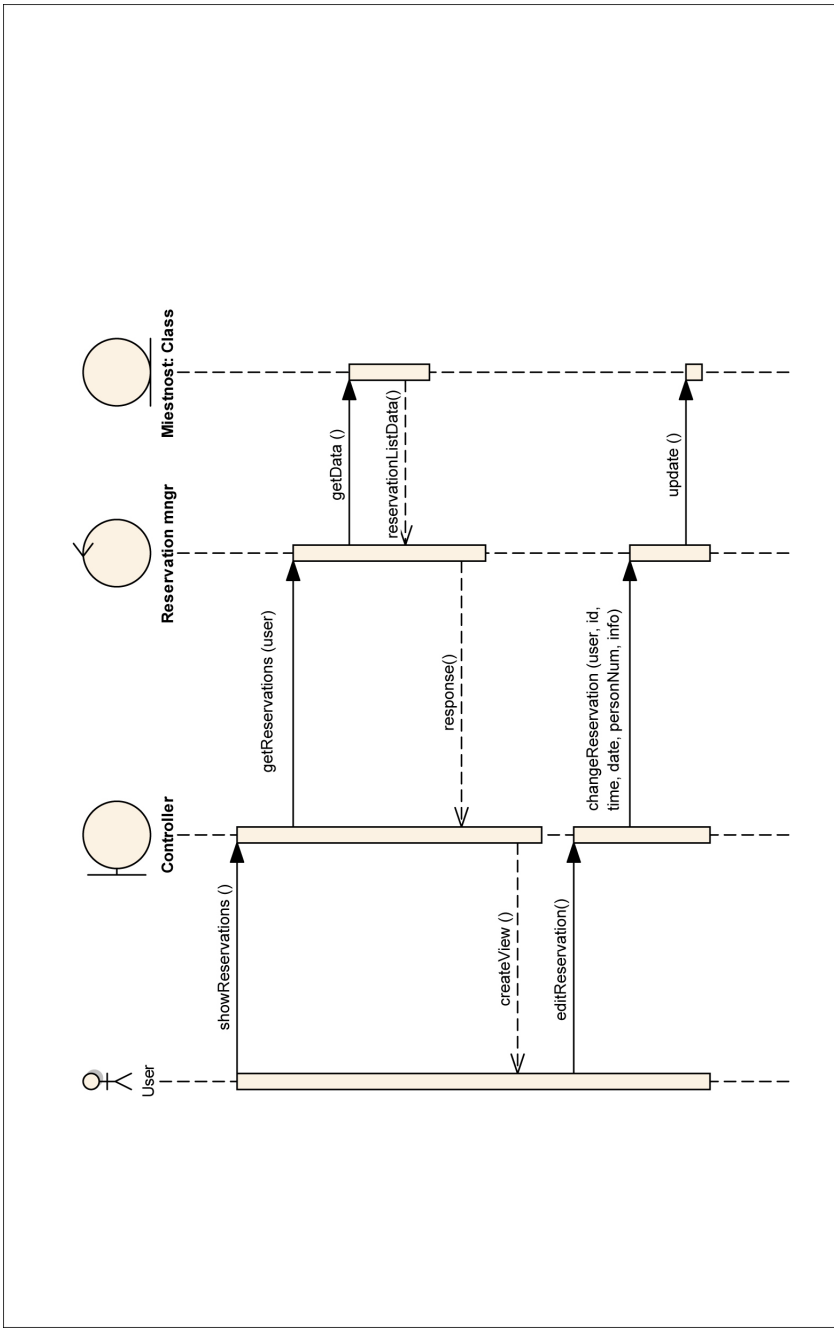


Obr. C.3: Sekvenčný diagram pre prípad použitia č.6 zrušenie rezervácie

Úprava rezervácie

`PUT url_webovej_časti/api/events`

Posielané dáta sú objekt `rezervácia`.

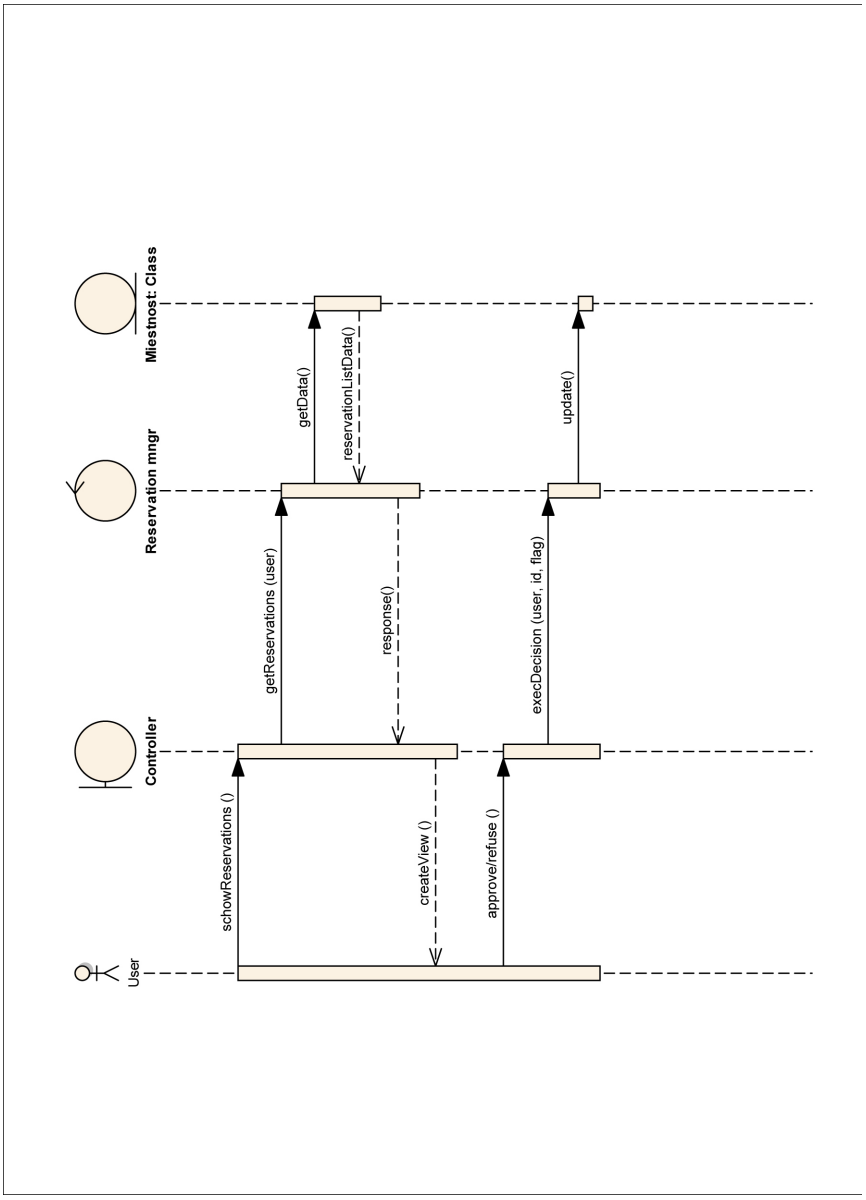


Obr. C.4: Sekvenčný diagram pre prípad použitia č.7 úprava rezervácie

Schválenie/neschválenie rezervácie

`PUT url_webovej_časti/api/events`

Posielané dáta sú objekt `rezervácia`.

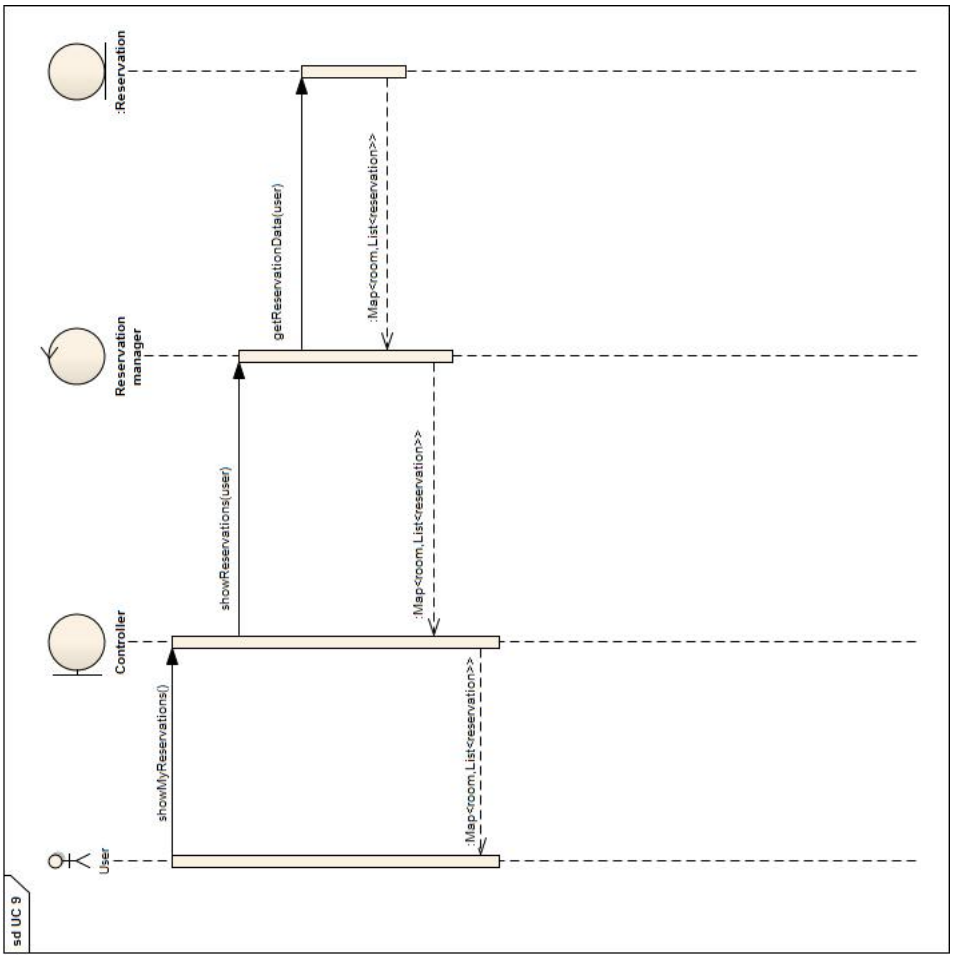


Obr. C.5: Sekvenčný diagram pre prípad použitia č.8 schválenie/neschválenie rezervácie

Zobrazenie svojich rezervácií

`GET url_webovej_časti/api/events/{username}?dateFrom&dateTo`

Parametrami sú uername, dátum začiatku a dátum konca.

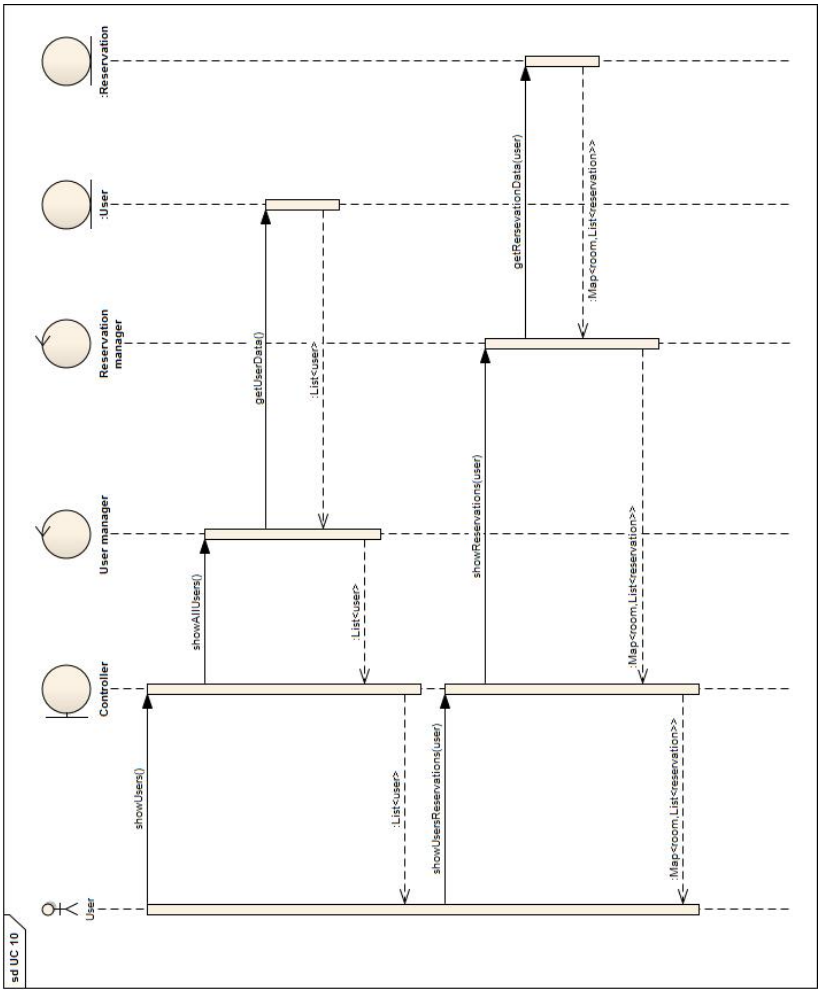


Obr. C.6: Sekvenčný diagram pre prípad použitia č.9 zobrazenie svojich rezervácií

Zobrazenie používateľov

```
GET url_webovej_časti/api/users
GET url_webovej_časti/api/users/{username}
```

Vrátené dáta sú objekt `používateľ`.

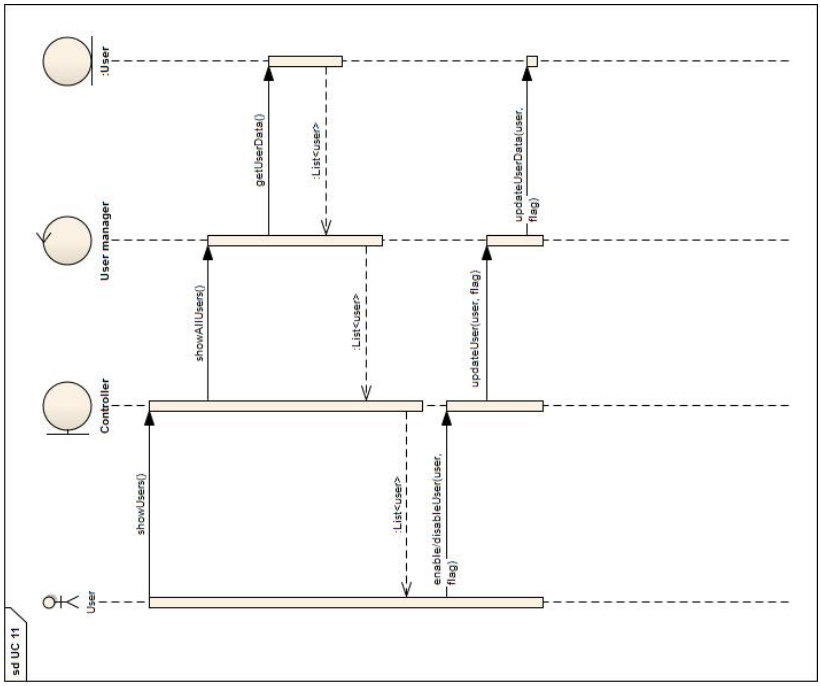


Obr. C.7: Sekvenčný diagram pre prípad použitia č.10 zobrazenie používateľov

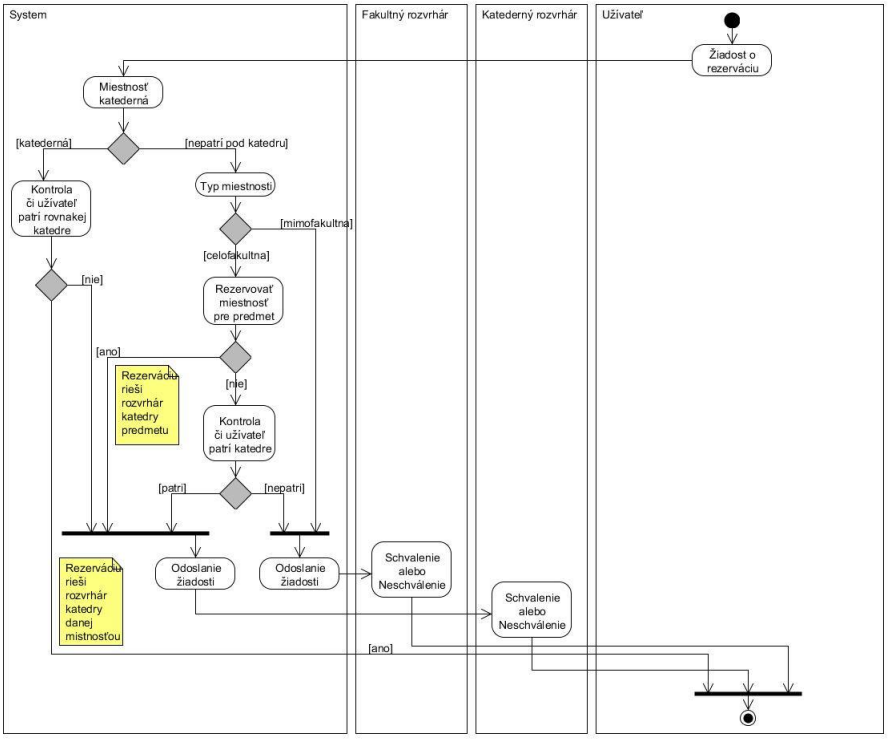
Blokovanie používateľov

```
PUT url_webovej_časti/api/users
```

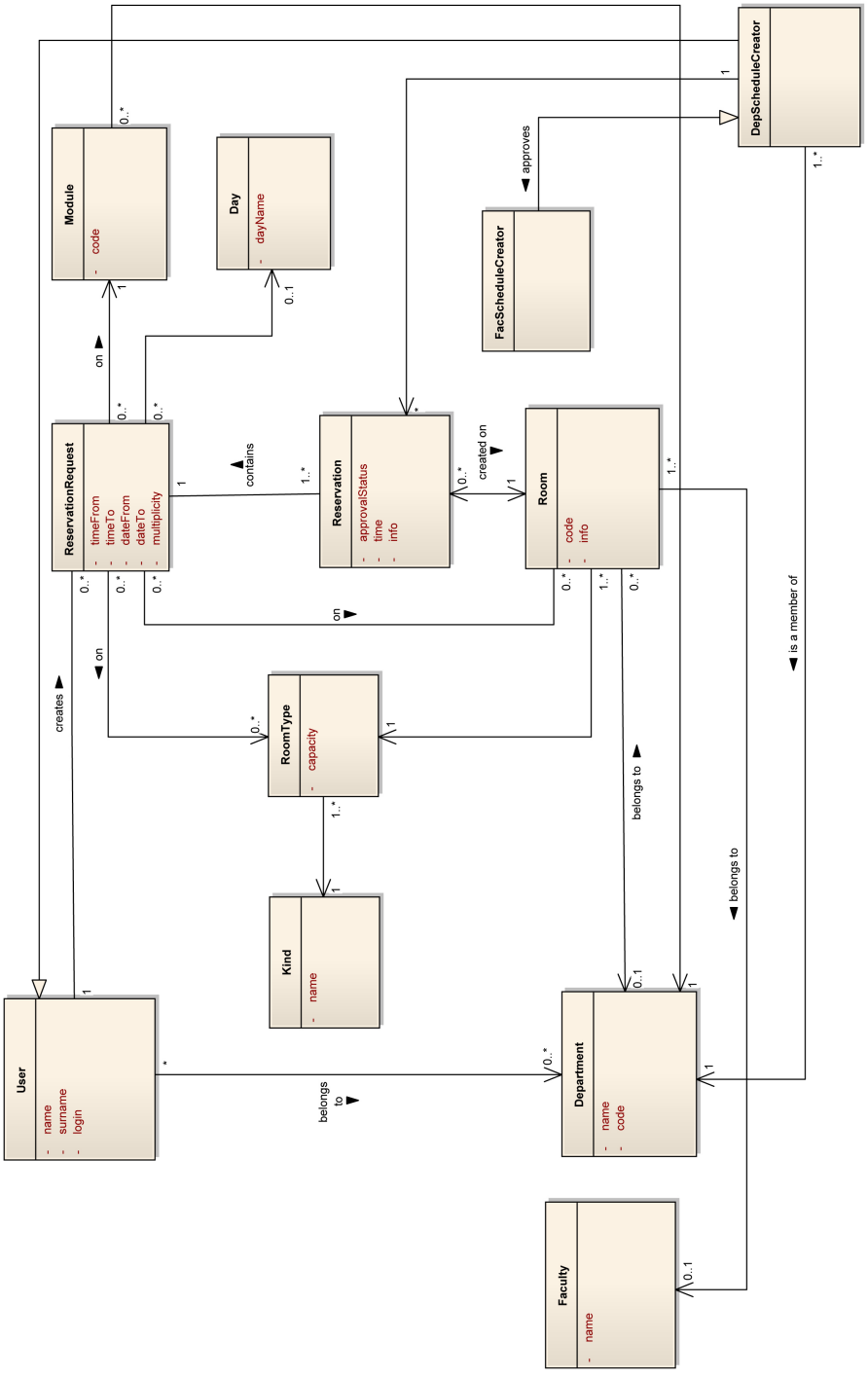
Posielané dáta sú objekt `používateľ`.



Obr. C.8: Sekvenčný diagram pre prípad použitia č.11 blokovanie používateľov



Obr. C.9: Aktivity diagram pre prípad použitia č.2 rezervácia miestností



Obr. C.10: Doménový model

.tex

Dodatok D

Obsah priloženého CD

```
readme.txt ..... the file with CD contents description
impl ..... source codes
deploy ..... instalation guide
thesis ..... the directory of LaTeX source codes of the thesis
text ..... the thesis text directory
    thesis.pdf ..... the thesis text in PDF format
```