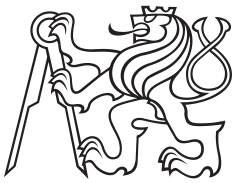


České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Systém pro správu státnicových okruhů

Dmitrii Gritsai

Vedoucí: Ing. Jiří Šebek
Květen 2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Gritsai** Jméno: **Dmitrii** Osobní číslo: **466284**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Systém pro správu státnicových okruhu

Název bakalářské práce anglicky:

Application for management final exam topics

Pokyny pro vypracování:

Cílem bakalářské práce je vytvořit systém, který usnadní agendu okolo statnic (specificky sestavování státnicových komisí) na katedře počítačů.
Funkce systému:
-zařazení daného člena (prof. docent, a zbylé pozice - doc. a prof. jsou nutné pro predsedovanie) do komise,
-komise musí obsahovat profesora nebo docenta v pozici předsedy
-schválení člena komise vědeckou radou
Systém musí obsahovat informace:
-zaměření člena komise (pro kterou komisi je vhodný - například SIT, OI),
-úvazek člena
Systém bude nasazen s realnými daty. Hlavním požadavkem je user friendly UI, proto bude práce zamerená i na část protoypu.
Obsahem práce bude:
-sběr požadavků
-analýza, návrh
-prototypování
-imlementace
-testování
-nasazení

Seznam doporučené literatury:

Omilusik, Dale. "Spring boot." U.S. Patent No. 4,660,299. 28 Apr. 1987.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Jiří Šebek, kabinet výuky informatiky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **21.02.2021** Termín odevzdání bakalářské práce: **21.05.2021**

Platnost zadání bakalářské práce: **19.02.2023**

Ing. Jiří Šebek podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. _____
Datum převzetí zadání

_____
Podpis studenta

Poděkování

Chtěl bych poděkovat vedoucímu své práce Ing. Jiří Šebkovi za jeho cenné rady, čas a trpělivost a zkušenosti, které jsem získal v rámci práce na tomto projektu. Rád bych také poděkoval své rodině, kamarádům a kolegům za neustálou podporu při studiu.

Prohlášení

I declare that this work is all my own work and I have cited all sources I have used in the bibliography.

Prague, května 21, 2021

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu.

V Praze, 21. May 2021

Abstrakt

Tato bakalářská práce se zabývá návrhem a vývojem webové aplikace pro vytváření státnicových komisí, která umožní uživatelům rychle a bez problému vytvářet skupiny učitelů pro státní závěrečné zkoušky.

Klíčová slova: Java, Spring Boot, React.js, webová aplikace, statní zkoušky

Vedoucí: Ing. Jiří Šebek

Abstract

This bachelor thesis deals with the design and development of a web application to create state commissions, which will allow users to quickly and easily create groups of teachers for final state examinations.

Keywords: Java, Spring Boot, React.js, web application, final exams

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

Tato práce se zabývá analýzou, návrhem a následným vývojem webové aplikace pro vytváření státnicových komisí.

1.1 Motivace

Státnice (neboli státní závěrečné zkoušky) jsou nezbytnou součástí vysokoškolského vzdělání. Jejich cílem je ověření znalostí získaných studentem během celé doby studia. U státnic zasedají členové zkušební komise (skupina učitelů, především z fakulty, na které probíhají konkrétní státnice). Sestavení těchto zasedacích komisí je poměrně složitá kombinatorická úloha. Sekretariát fakulty se jí zabývá docela dlouhou dobu, protože různé možnosti složení zkušební komise kombinuje „na papíře“. Zástupce sekretariátu FELu proto přišlo s požadavkem vymyslet a implementovat systém, který by složení komise usnadnil.

1.2 Cíle

Cílem této práce je navrhnout a implementovat informační systém, který umožní sekretariátu rychle a bez problému vytvářet skupiny učitelů pro státní závěrečné zkoušky.

Projekt je rozdělen do několika kapitol:

1. Sběr a analýza požadavků. Tato kapitola popisuje průběh sběru a specifikaci požadavků.
2. Návrh řešení. Dále bude popsán navržený prototyp a získaná zpětná vazba po testování sekretariátem.
    a. Implementace prototypu.
    b. Testování prototypy sekretariátem.
    c. Úpravy po testování. Pravděpodobné je, že než budou všechna přání sekretariátu splněna, tento cyklus se zopakuje několikrát.
3. Po testování prototypu začne samostatná implementace aplikace. Bude potřeba vyvinout backendovou a frontendovou část

4. Nasazení. Nemalou část zabere nasazení celého systému na již existující servery a finální testování funkčnosti aplikace.

Kapitola 2
Rešerše

Tato kapitola se zabývá analýzou na trhu existujících řešení, která cílí na tvorbu plnohodnotných webových aplikací. Doplněna je o technologie na implementaci frontendových a backendových části aplikace.

2.1 Analýza již existujících řešeni

Bylo provedeno hledání aplikací podobných té, která se dle zadání bude vyvíjet. Neočekávalo se, že se podaří najít aplikaci, která obsahuje podobnou funkcionalitu [3.3], jako ta, která se implementuje. Výsledná aplikace musela splňovat minimálně následující požadavky:

1. Možnost vytvářet náhodné skupiny lidi z předdefinovaného seznamu.
2. Možnost přidávat různé parametry pro vytváření skupiny – například, že by ve skupině byl přítomen člověk disponující nějakou vlastností.

Ve většině výsledcích vyhledávání se objevily jednoduché aplikace (především pro mobilní zařízení) pro vytváření skupin. Takové aplikace však pro účel našeho projektu nepostačují.

2.2 Analýza technologií pro vývoj webových aplikací

V této podkapitole jsou zkoumány oblíbené technologie, pomocí nichž se v současnosti staví webové aplikace. Je rozdělena na dvě části: analýzu backendových technologií a analýzu frontendových technologií.

2.2.1 Backend

Protože systém představuje kompletní plnohodnotnou webovou aplikaci, je potřeba si zvolit vhodný nástroj pro jeho realizaci. Vzniká tak otázka, jakou technologii pro vývoj webové aplikace použít. V dnešní době jsou nejpopulárnější následující backend frameworky: Laravel, Django, Spring Boot a další [1]. V následující podkapitole jsou uvedeny některé vlastnosti vyjmenovaných technologií.

Laravel

Laravel je PHP framework určený pro vývoj webových aplikací. Laravel je více méně nový framework (poprvé vydaný v roce 2011). Laravel nabízí vývojářům celý ekosystém. Co se týče vývoje na PHP, Laravel obsahuje téměř vše, co je potřeba pro vytvoření robustní backendové aplikace: PHPUnit je framework pro testování, Artisan je rozhraní příkazového řádku pro migraci databáze a vytváření modelů. S pomocí vestavěného serveru Homestead lze vytvářet virtuální prostředí, pracovat se směrováním, zpracovávat middleware a také pohodlně propojovat pohledy se šablonami Blade [4].

Na frameworku Laravel běží populární webové portály, jako jsou Deltanet Travel, Neighborhood Lender a MyRank.

Django

Django je open source web framework napsaný v Pythonu, který se volně drží architektury Model-view-controller. Mezi programátory se Python považuje za jazyk, pro který je potřeba nízká úroveň vstupních znalostí.

Silné stránky Django:

- Framework podporuje návrhový vzor MVC.
- Je to rychlý framework, není zatížený nadbytečnými funkcemi.
- Projekty na Django jsou kompaktní v kódu.
- Django je projekt fungující napříč platformami. Skvěle funguje na různých operačních systémech. Kromě toho podporuje interakci s různými databázemi.
- Aplikace postavené na Django jsou dobře rozšiřovatelné.

Na frameworku Django běží populární webové portály, jako jsou Instagram, Pinterest a Coursera.

Spring Boot

Spring Boot [2] je vývojový backendový framework založený na Java, který se používá jak pro vývoj mikroservisu, tak i pro rozsáhlejší projekty. Tento framework usnadňuje vytváření aplikací založených na Spring.

Výhodami tohoto frameworku jsou:

- Umožňuje vytvářet samostatné Spring aplikace s minimálním úsilím a zjednodušuje proces jejich konfigurace. Aplikace na Spring Boot lze lehce spustit pomocí příkazu java-jar.Umožňuje vytvářet samostatné Spring aplikace s minimálním úsilím a zjednodušuje proces jejich konfigurace. Aplikace na Spring Boot lze lehce spustit pomocí příkazu java-jar.
- Má interní analyzátor chyb, který je umožňuje snadno odhalit už v průběhu stavby(buildu) aplikace.

- Podporuje aplikační servery Tomcat a Jetty. Tudíž programátor se nemusí zabývat spuštěním aplikačního serveru. Framework to zařídí sám.
- Poskytuje názorné „startovací“ závislosti zjednodušující konfiguraci.
- Absolutně žádné generování kódu a žádný požadavek na konfiguraci XML.
- Usnadňuje použití principu softwarového patternu známého jako „Convention over Configuration“ [3].

Jednou z největších výhod ve prospěch Spring Bootu je skutečnost, že na fakultě používáme Javu se Spring Bootem v mnoha předmětech. Navíc autor práce má s tímto frameworkem vlastní zkušenosti. Proto bylo rozhodnuto pro vývoj použít Spring Boot framework.

Na frameworku Spring Boot běží populární webové portály, jako jsou Trivago, Via Varejo a Intuit.

2.2.2 Frontend

Pro implementaci frontendové části aplikace se autor rozhodl použít JavaScript framework. Na základě nejrůznějších porovnání a výzkumů [5] byl učiněn závěr, že bude vybrán jeden ze tří populárních frameworků: Vue.js, Angular.js a React.js.

Angular

Angular je nejstarší z výše uvedených frameworků, má dobrou podporu. Neyýhodu Angularu autor spatřuje v tom, že pro jeho použití je potřeba se naučit nový jazyk – TypeScript. A to, protože veškerá dokumentace k Angularu je napsaná v něm.

React

React si podle výzkumů [6] získal široké přijetí na trhu. Na trhu práce je po odbornících znalých Reactu sháňka a budoucnost tohoto frameworku vypadá dobře. React může být dobrou volbou pro někoho, kdo začíná s frontendovými frameworky na JavaScriptu. Schopnost bezproblémové integrace s jinými frameworky poskytuje velkou výhodu pro ty, kteří chtějí ve svém kódu určitou flexibilitu.

Vue

Vue je v seznamu nejpoužívanějších frameworků nejnovější. V posledních několika letech si zároveň vede jako silný konkurent pro Angular a React. Tento fakt pravděpodobně hraje roli u mnoha čínských gigantů, jako jsou Alibaba a Baidu, kteří si jako primární framework JavaScriptu vybírají právě Vue.

Všechny tři frameworky jsou vhodné pro implementaci frontendu. Autor práce se rozhodnul pro React.js. Framework React.js má velmi podrobnou dokumentaci a rozsáhlý počet uživatelů, což může výrazně urychlit proces učení a ušetřit čas na vývoj aplikace. Během práce s tímto frameworkem autor projektu navíc získá spoustu zkušeností, které pro něj budou přínosem také v budoucnosti. React.js je známý i díky svým výhodným vlastnostem.

React Komponenty

React disponuje komponenty, které se umí samy spravovat. Komponenty fungují na základě dědičnost a zapouzdření, což jim umožňuje vytvářet komplexní UI.

React Stavy

Stavy (neboli "state") jsou silnou vlastností, díky níž lze snadno předávat data prostřednictvím aplikace a udržovat stav mimo DOM. Vytváření interaktivních uživatelských rozhraní je proto snadné. Stačí navrhnout jednoduchá zobrazení pro každý stav a React bude při změně dat efektivně aktualizovat a zobrazovat ty správné komponenty.

React lze také zobrazit na serveru pomocí Node.js a napájet mobilní aplikací s pomocí React Native.

React Redux

Redux je stavový kontejner pro aplikace JavaScriptu. Pomáhá psát aplikace, které se chovají konzistentně, běží v různých prostředích (klient, server a nativní) a lze je snadno otestovat. Kromě toho poskytuje výhody pro vývojáře, jako je například editace živého kódu v kombinaci s časovým ladicím programem. Redux lze použít společně s Reactem nebo s jakoukoliv jinou frontend knihovnou. Je malý (2 kB, včetně závislostí), ale má k dispozici velký ekosystém doplňků. [7]

2.2.3 Závěr

Tato kapitola se zabývala rešerší výběru vhodného nástroje na implementaci systému. Ve výsledku byla zvolena následující kombinace: za backendovou část bude zodpovědný Spring Boot framework pro jazyk Java, za frontendovou - React.js framework pro JavaScript.

Kapitola 3

Analýza

V této kapitole je popsán stávající stav a účel aplikace. Svou pozornost pak autor směruje k definování funkčních a nefunkčních požadavků na systém.

3.1 Stávající stav

Současná situace je taková, že sekretariát tráví spoustu času vymýšlením podoby a sestavováním komisí a zjišťováním, který učitel je zrovna volný a který nemůže být k dispozici. Nejde přitom o hodiny, ale o dny. Po sestavení komise sekretariát musí nějakým způsobem (nejčastěji e-mailem) zvoleným členům příslušných komisí oznámit nutnost jejich přítomnosti u závěrečné zkoušky.

Po několika bezúspěšných žádostech o vytvoření systému v rámci fakulty sekretariát převzal iniciativu do vlastních rukou a o návrh systému požádal vedoucího autorovy práce, který projekt zadal studentům jako semestrální práci.

3.2 Účel aplikace

Aplikace je určena především k usnadnění práce sekretariátu. V podstatě jde o plnohodnotnou webovou aplikaci, která bude v ideálním scénáři přístupná po přihlášení v SSO. Systém bude představovat "online" interaktivní aplikaci, která bude zohledňovat funkční požadavky, které jsou popsány dále.

3.3 Požadavky

V podkapitole "Funkční požadavky" jsou vyjmenované požadavky sekretariátu na aplikaci [23]. Požadavky jsou rozdělené na podkategorie MUST HAVE a NICE TO HAVE. Některé požadavky jsou vlastním návrhem autora.

3.3.1 Funkční požadavky

MUST HAVE

1. Systém bude umožňovat uživateli vytvářet nové skupiny lidí.
2. Systém bude umožňovat uživateli editovat již existující komise.
3. Systém bude umožňovat uživateli schvalovat již existující komise.
4. Systém bude umožňovat uživateli měnit stav existující komise.
5. Systém bude umožňovat uživateli přidávat/měnit pracoviště v rámci existující komise.
6. Systém bude umožňovat uživateli nastavovat úvazek každému členu komise.
7. Systém bude umožňovat uživateli výběr z vícero kombinací komisí.
8. Systém bude umožňovat uživateli volit obor/zaměření, pro nějž je komise je vhodná.
9. Systém bude schopen automaticky posílat e-maily různých typů.
10. Systém bude umožňovat členům komisí reagovat na výzvu.
11. Systém bude umožňovat členům komisí doporučit dalšího člena komise.

NICE TO HAVE

1. Systém umožňuje učitelům preferenci oboru, jehož zkoušejícím chce v rámci komise být.
2. Systém bude schopen export data z Excelu v CSV podobě.
3. Systém bude umožňovat uživateli zprostředkovat přístup do aplikace dalším uživatelům.
4. Systém bude umožňovat uživateli vytvářet a editovat e-mailové šablony.

3.3.2 Nefunkční požadavky

1. Uživatelsky přívětivé a jednoduché UI.
2. Rychlost.
3. Systém bude pracovat s reálnými data z již existující databáze.
4. Funkčnost ve všech moderních prohlížečích.
5. Responsible design.
6. Lokalizace (CZ, EN) jako nice to have.

3.4 Závěr

V této kapitole jsou popsány současný stav a účel aplikace a definované funkční a nefunkční požadavky. Na začátku vývoje aplikace byl autor připraven na možné požadavky na změny, ale k těm nakonec nedošlo.

Kapitola 4
Návrh

Tato kapitola obsahuje koncept navrženého systému. Dále obsahuje popis základních entit systému. V dalších částech kapitoly se autor zaměřuje na proces vytváření komisí, kvůli němuž se aplikace vyvíjí, a popisuje jednotlivé role uživatelů v systému. Všechny diagramy jsou vytvořené v jazyce UML[8] pomoci nástroje Sparx Enterprise Architect [6.4].

4.1 Entity

Diagram tříd [4.1] zohledňuje balíček `model`, který obsahuje základní entity systému.

4.1.1 User

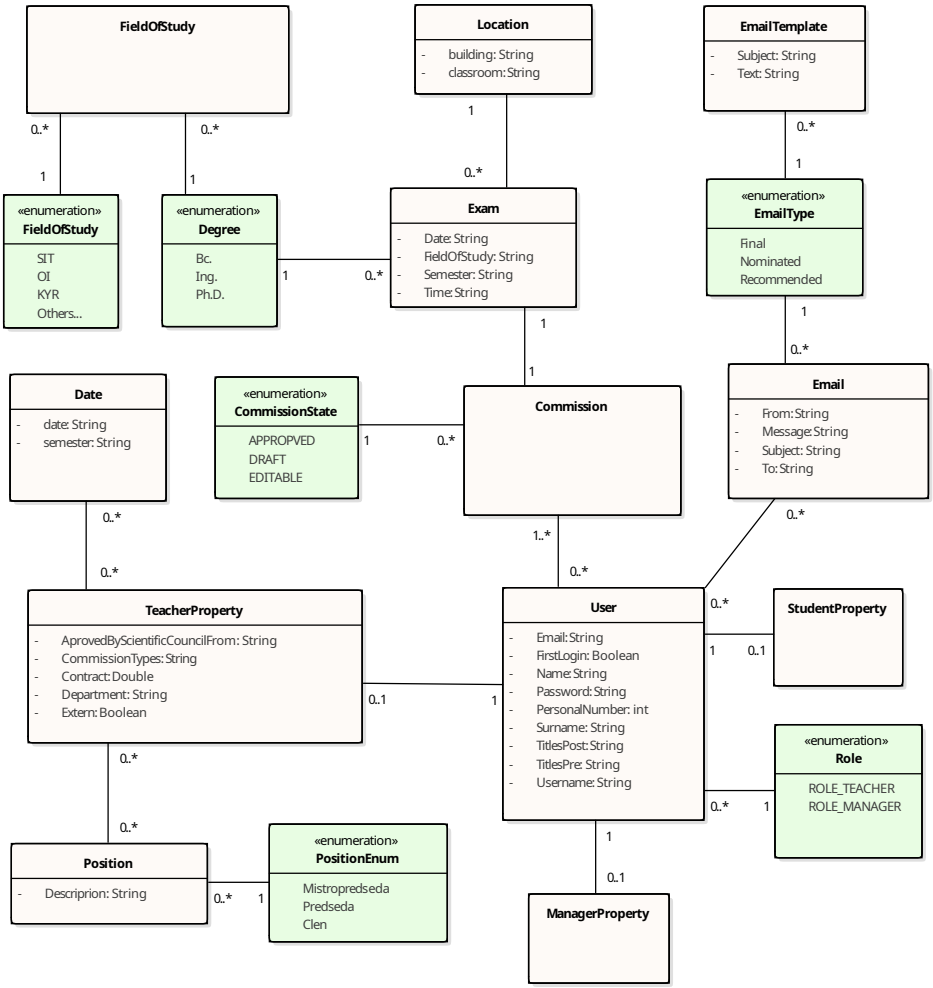
`User` je jednou z hlavních a největších entit v systému. Tato entita obsahuje atributy popisující člověka. `User` v systému má dvě role: `Teacher` a `Manager` [4.3]. Uživatelé Manager a Teacher mají odlišné atributy, a proto bylo rozhodnuto unikátní atributy přesunout do samostatných tříd `ManagerProperty` a `TeacherProperty`. Existuje také třída `StudentProperty`, která by měla reprezentovat studenta, ale nepoužije se. V budoucnosti může být použita při případném rozšíření funkcionality systému o možnost přiřazení studentů k jednotlivým zkouškám a komisím. Student tento systém nebude nijak používat, a proto ho zmiňujeme pouze jako entitu, a ne jako uživatele.

4.1.2 Commission

`Commission` vypadá v aplikaci jako hlavní entita, ale nemá unikátní atributy a v podstatě je jenom souhrnem jiných entit. Má atributy `teachers`, `exam`, `location` a `state`. State je enum který může nabýt až tří hodnot: Approved, Editable a Draft. Co znamenají tyto stavy a kde se používají, se popisuje v části věnované vytváření komisí [4.2].

4.1.3 E-mail

Jde o jednoduchou třídu reprezentující e-mail. E-maily se budou rozesílat na žádost uživatele. Pro odesílání e-mailů byl vytvořen speciální účet na Gmail. Pro vytvářeni a posílaní e-mailů přímo z aplikace byl použit `JavaMailSender` z balíčku `org.springframework.mail.javamail`. E-mail má atribut `emailType` a tento atribut má tři hodnoty: Final, Recommended a Nominated. E-maily různých typů se budou odlišovat pouze šablonou zprávy a předmětem zprávy. Tyto šablony lze bez problémů editovat.



Obrázek 4.1: Class diagram

4.2 Proces generování komisí

Diagram představující celý proces vytváření komise je obsazen v přiloze [C.4].

4.2.1 Popis procesu vytváření komise z hlediska uživatele:

1. Manager si zvolí potřebné parametry pro vytvoření komise: datum a čas začátku zkoušky, obor a místnost.

2. Systém podle vstupních parametrů nakombinuje seznamy učitelů. Systém zohlední časovou dostupnost učitelů, jejich preferenci a úvazky. Pro výběr učitele systém zohledňuje také to, jestli je oprávněný v komisi zasedat.

3. Systém vytvoří komisi a přiřadí ji stav DRAFT. Tento stav je určený pouze pro interní použití.

4. Systém zobrazí seznam komisí s různými kombinacemi učitelů.

5. Manager zvolí komisi, kterou chce vytvořit kliknutím na tlačítko Create. Zvolená komise poté přejde do stavu EDITED a zbylé (se stavem DRAFT) se vymažou. Vytvořenou komisi manager může najít po kliknutí na tlačítko `Commissionlist` v headeru stránky.

6. Manager může komisi editovat, pokud nebyla převedena do stavu APPROVED. Pokud už komise v tom stavu je, editovat ji nelze.

4.2.2 Proces vytváření komisí z hlediska systému

1. Na REST controller, který poslouchá na koncovém bodu `/util/gen/count` nebo na `/commission/create` přijde POST požadavek obsahující objekt `CreatorTO`. Tento objekt obsahuje veškeré potřebné informace pro vytvoření komise: čas, datum, studijní program, identifikátor místnosti a seznam učitelů.

2. V případě požadavku na automatické vytvoření komise kontrolér vyvolá metodu `generateCommission` ze třídy `CommissionMaker`. Tato metoda vybere z databáze pouze učitele, kteří jsou v daný den k dispozici (nepřisedají jiné komisi a nemají ten den obsazený jinou aktivitou). Aplikace poté nakombinuje seznamy komisí, které budou ihned filtrovány. Jeden filtr vymaže seznamy, v kterých není učitel, který může v komisi zasedat v roli předsedy. V případě požadavku na manuální vytvoření už není potřeba nic kombinovat, protože seznam učitelů je již obsahem objektu `CreatorTO`. Kontrolér rovnou vyvolá metodu `saveManual` z třídy `CommissionService` a přidá objekt `CreatorTO` do nej.

3. V obou případech třídy-servisy uloží novou komisi do databáze. Během ukládání komise do databáze servisy vytvoří příslušné `Exam` objekty.

4. Po vytvoření a uložení komise, metody vrací novou komisi.

Sekvenční diagramy, popisující proces vytváření komisi jsou v příloze teto práce. [C.1] [C.2]

4.3 Role v systému

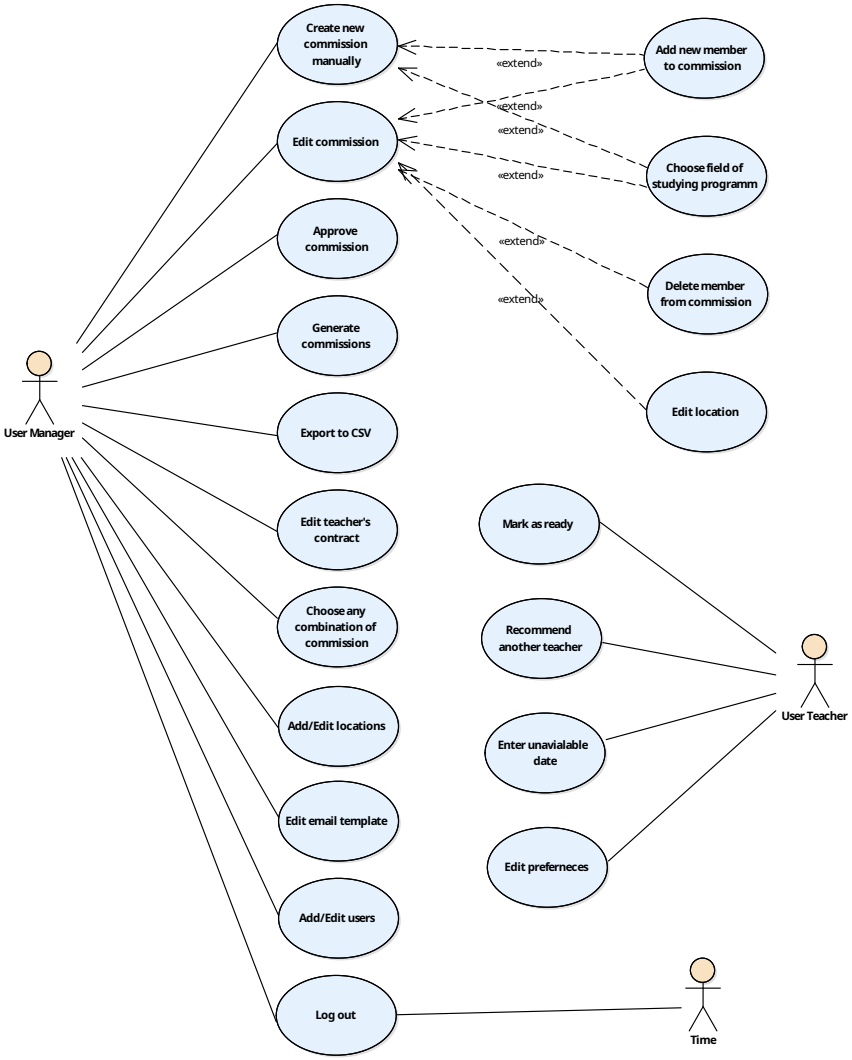
Funkcionalita systému je rozdělena na dvě části - manažerskou a učitelskou, a bude využívána dvěma typy uživatelů: Manager a Učitel.

Manažer

Manažer představuje základního uživatele (očekávání je že to bude pracovník sekretariátu FELu).

Učitel

Učitelem v systému nazýváme člověka, který byl vyzván k zasedání u státnicových komisi.



Obrázek 4.2: Use Case diagram

Na obrázku [4.2] jsou znázorněny případy užití systému rozdělené podle role.

4.4 Závěr

V této kapitole se autor zaměřil na návrh aplikace pro vytváření státnicových komisí. Byl popsán účel hlavních entit systému a následně postup sestavování komisí a popsaní rolí v systému. V další kapitole autor popíše prototyp aplikace a podrobněji se rozepíše o jejím testování.

Kapitola 5
Prototypování

Tato kapitola je rozdělena na dvě podkapitoly - Prototyp a Testování prototypu. Podkapitola Prototyp obsahuje souhrnné informace o prototypu a o tom, jaké funkce náleží jednotlivým uživatelům. Podkapitola Testování popisuje proces testování prototypu a jeho výsledky.

5.1 Prototyp

Samotná aplikace se dělí na několik částí. První část je manažerská, která je přístupná pracovníkům sekretariátu, druhá část je učitelská, která bude přístupná pouze učitelům, kteří byli vyzváni k účasti v komisi státní závěrečné zkoušky.

5.1.1 Manažerská část

V této části systému se realizuje podstatnější část funkčních požadavků na systém. Za důležité autor práce považuje následující:

- Vytváření a modifikace státnicových komisí;
- Automatické vytváření komisí podle zvolených parametrů;
- Informování předsedů komisí o jejich účasti;
- Nastavení úvazku;
- Editace šablon e-mailů;
- Editace uživatelských oprávnění;
- Vytváření nových uživatelů;

5.1.2 Učitelská část

Tato část aplikace představuje rozhraní pro učitele, který byl vyzván k zasedání ve státnicové komisi. Učitelům jsou dostupné zcela jiné scénáře pro práci se systémem než uživatelům v roli manažerů.

- Reakce na výzvy o členství ve státnicové komisi;
- Návrh náhradního učitele v případě nemožnosti účasti;
- Oznámení systému o vlastních časových možnostech s pomocí jednoduchého rozhraní kalendáře;

V příloze jsou screenshoty hlavni stránky uživatele v roli Manager [C.5] a Učitel [C.6].

5.2 Testování prototypu

Cílem testovaní prototypů není ověřit funkčnost jednotlivých části systému, protože prototyp představuje pouze několik desítek wireframů s předepsanými daty a tlačítky umožňující přechod mezi wireframy. Cílem testování je získat UX (User Experience) uživatele a dozvědět se o jeho celkovém dojmu z aplikace. Účastníci testování projdou dva scénáře. Po skončení testování odpoví otázky v jednoduchém dotazníku. Zároveň se budou moct rozepsat o vlastních názorech a myšlenkách.

5.2.1 Postup testování

Účastníkovi bude popsán hlavní problém, který bude řešit pomocí vyvinuté aplikace. Účastník dostane seznam úkolů (neboli scénářů), které bude muset splnit. Účastník poté vyplní krátký a jednoduchý dotazník.

Scénář role Manager:

1. Vytvořit novou skupinu ručním přidáváním učitelů.
2. Využít automatické vytváření skupin a vybrat libovolný počet nabídnutých skupin.
3. Poslat e-mail skupině.
4. Požádat o vytvoření CSV.

Scénář role Teacher:

1. Reagovat na výzvu k zasedání ve státnicové komisi.
2. Označit v kalendáři dny, ve kterých se učitel nebude moct komise zúčastnit.

Účastníci testování budou seznámeni s omezeními prototypů. Například, že prototyp neumožňuje psát do input fieldu, a že při zařazení jednoho konkrétního učitele do komise, se zařadí jiný, předem definovaný učitel.

Je třeba zmínit, že účastníci testování nepatří do cílové skupiny uživatelů. Stalo se to kvůli špatné komunikaci s iniciátory, kteří jsou zároveň cílovou

skupinou systému. Z toho plyne, že výsledky z testování nemusí být úplně relevantní.

5.2.2 Screener

Screening znamená výběr respondentů na základě potřebných charakteristik (demografie, znalosti, chování, zvyklosti, postoje atd.) na základě konkrétních potřeb projektu. Vždy je totiž potřeba testovat s relevantní cílovou skupinou [10].

Za cílovou skupinu je považovaný dospělý člověk se zaměstnáním ve vzdělávací oblasti (primárně z ČVUT FEL), který má alespoň základní znalosti o používání počítače a webového prohlížeče.

5.2.3 Výsledky testování

Scénář role Manager

Účastník číslo 1, 26 let, umělec. Testující zvládl bez větších potíží veškeré kroky. Objevilo se však několik nejasností, které je potřeba zohlednit. Testující byl zmatený a nechápal, co znamenají zelená a žlutá kolečka u jmen učitelů v sekci `Commissions List` [C.5], jestli je na ně možné například kliknout. Účastníkovi přišlo divné, že není možné zrušit filtraci v sekci `Commissions List`. Účastníkovi nepřipadlo pohodlné a intuitivní ani rozhraní manuálního vytváření komisi.

Účastník číslo 2, 22 let, student. Testující zvládl bez větších potíží veškeré kroky. Během plnění třetího kroku ho překvapilo, že mu nebyla nabídnuta možnost editace e-mailu, který se zamýšlel odeslat.

Scénář role Učitel

Účastník číslo 2, 22 let, student. Testující zvládl bez potíží veškeré kroky. Avšak, v prvním kroku, v případě možnosti, že se učitel nemůže komise zúčastnit, uživatel klikne na určené tlačítko a systém mu nabídne dialogové okno se jmény možných náhradníků. Tato vlastnost systému se účastníkovi testu nelíbila. Účastník testu by měl raději možnost napsat omluvu nebo důvod, proč se zúčastnit nemůže, a to rovnou v aplikaci a bez toho, aniž by musel psát samostatný e-mail.

5.3 Závěr testování

Oba testující měli celkově z aplikace dobrý dojem a jednoduše splnili všechny scénáře. Ocenili jednoduchost vzhledu a minimalismus v designu, a to, že výsledný vzhled aplikace nehýří příliš velkým množstvím barev. Nic je tak nerušilo od splnění úkolu. Celkem toto testování je považováno za úspěšné za výjimkou toho, ze respondenty nepatřili do cílové skupiny. Zpětná vazba,

získaná během testováni bude proanalyzovaná a připadne vylepšení vzhledu budou implementovány.

Kapitola 6
Implementace

Tato kapitola popisuje architekturu aplikace. Zabývá se serverovou a klientskou částí aplikace. [11] Také zohledňuje proces implementace webové aplikace a problémy, které během ní vzniknou. Na konci se krátce věnuje zálohování dat na serveru.

6.1 Serverová část

Jak bylo zmíněno na konci druhé kapitoly, pro implementaci backendové (serverové) části autor zvolil framework Spring Boot. Serverem je program zpracovávající HTTP požadavky přicházející od webového klienta. Obecně lze architekturu Spring Boot aplikace rozdělit na čtyři vrstvy, a to následujícím způsobem: Prezentační vrstva, Business vrstva, Persistence vrstva a Databázová vrstva.

6.1.1 Prezentační vrstva

Tato vrstva zpracovává HTTP požadavky přicházející společně s požadavky JSON, parametry překládá do objektu a následně předává do byznys vrstvy. Také má na starosti autentizaci uživatelů. V podstatě lze říci, že webový klient komunikuje s prezentační vrstvou, kterou v aplikaci představuje balíček s názvem `controllers`.

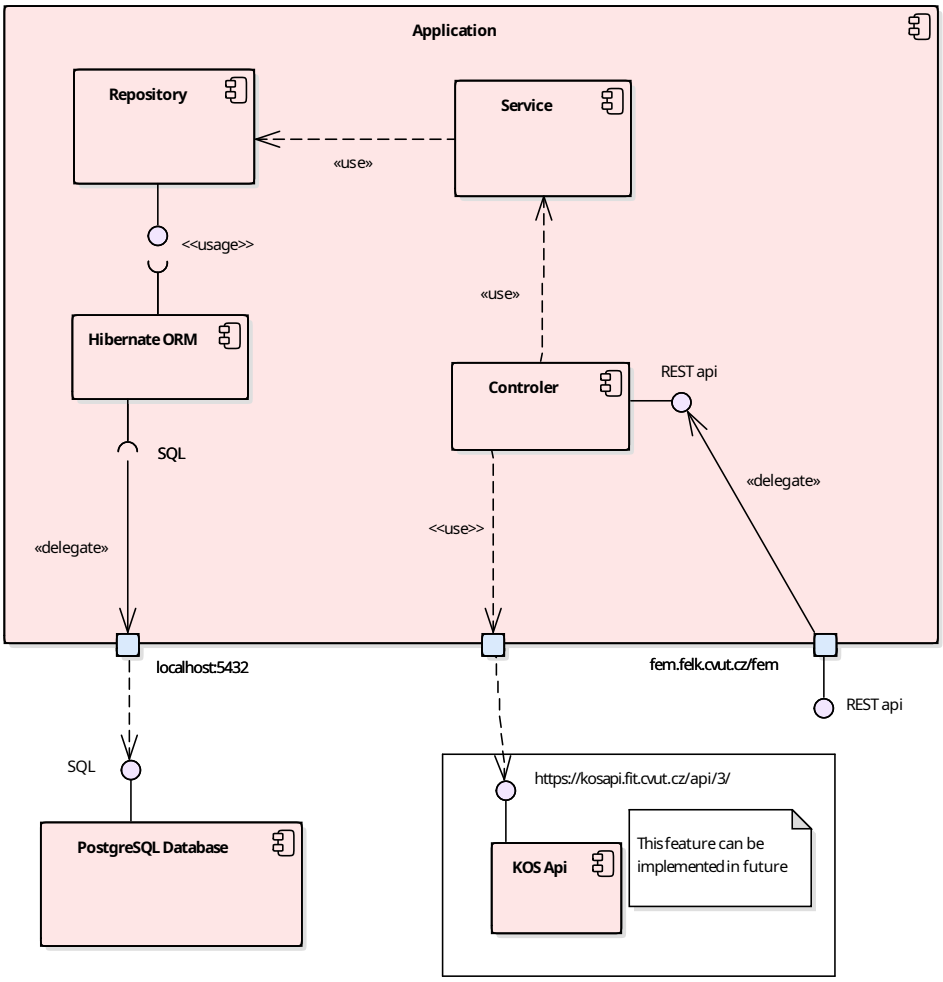
6.1.2 Business vrstva

Tato vrstva obsahuje veškerou logiku aplikace. Rovněž provádí autorizaci a validaci. V aplikaci ji představuje balíček s názvem `services`. Metody z toho balíčku většinou vrací buď samotné objekty, nebo data zabalené do `response` entity.

6.1.3 Persistence vrstva

Persistence vrstva zahrnuje metody pro komunikaci s databází. Ve zdrojovém kódu ji představuje balíček s názvem `repository`. Každý interface z tohoto balíčku dědí `org.springframework.data.jpa.repository.JpaRepository`.

Použití toho interface umožňuje zredukovat velké množství kódu, který je potřebný pro implementaci DAO (data access object) vrstvy, tudíž ve zdrojovém kódu `DAO` balíček není. Základní rozhraní obsahuje veškeré CRUD operace. V některých třídách ale nestačily základní CRUD operace, a proto bylo potřeba vytvořit vlastní SQL dotazy.



Obrázek 6.1: Diagram komponent

Obrázek [C.3] ukazuje architekturu aplikace. V příloze je možně najít rozšířenou verzi tohoto diagramu.

6.2 Struktura kódu

Spring Boot nevyžaduje žádnou konkrétní strukturu kódu. Existuje však několik postupů, kterými se řídí vývojáři Spring Boot. Nejlepší způsob, jak uspořádat kód webových aplikací, je obecně použít přístup strukturovaný

podle funkcí. Také existuje strukturovaní podle vrstev. Autorem bylo zvoleno strukturování podle vrstev důvodu, že je na strukturování zvyklý.

Zdrojový kód k výše zmíněným balíčkům, obsahuje následující: `Config`, `Model`, `TO`, `utils`.

6.2.1 Config

Balíček config obsahuje konfigurační soubory. Je zdrojem definic beanu pro kontext aplikace. Jsou tam konfigurace pro samotnou aplikaci, konfigurace pro odesílání e-mailů a také bezpečnostní konfigurace.

6.2.2 Model

Model obsahuje všechny entity systému [4.1]. Zahrnuje také subpackage obsahující enumy, které jsou použité ve třídách modelu.

6.2.3 TO

TO (angl. transfer object) balíček obsahuje třídy-objekty určené pro transfer dat mezi vrstvami. Důvod použití takových struktur je v tom, že ne vždy je potřeba posílat celou entitu, ale jen její část. V aplikaci, tam kde je to možné, jsou `TO` entity použity pro posílaní dat mezi serverovou a klientskou části. Jinak se posílají obecné entity.

6.2.4 Utils

Balíček `utils` obsahuje pomocné třídy, které nepatří do balíčku `model`. Patří mezi ně třída `CommissionMaker` která se stará o vytváření komisí, třídy pracující s CSV a XLS soubory a třídy, které řeší JWT autorizaci.

6.3 Klientská část

Na konci druhé kapitoly [2.2.3] bylo rozhodnuto, že pro implementaci frontendové části se použije framework React JS. Autor této práce se v praxi nesetkal s frameworkem ReactJS a se samotným JavaScriptem měl možnost pracovat jen částečné. Vytváření UI rozhraní proto pro něj bylo první zkušeností. Využil při tom své intuice a časové a lokální možnosti. Node.js je sada knihoven pro JavaScript, která umožňuje jeho použití mimo prohlížeč. Zaměřuje se primárně na vytváření jednoduchých a snadno sestavitelných síťových klientů a serverů. Během instalace Node.js se automaticky instaluje `npm`. Npm je správcem balíčků pro Node.js.

6.3.1 Struktura kódu

Bylo rozhodnuto zachovat výchozí strukturu `React app` a pouze ji rozšířit o vlastní psané komponenty. Zdrojový kód se nachází ve složce `src/main/frontend`.

Složka `frontend` je rootem React aplikace. Složka `src` dále byla rozdělena na podsložky následujícím způsobem:

- `components` obsahuje komponenty, které byly použity při vykreslování obrazovky. Samozřejmě, mnoho komponent je použito opakovaně a na několika místech zároveň, a proto byly přemístěny do zvláštních komponent;
- `pages`. Soubory ve složce pages reprezentují cele webové stránky;
- `styles` obsahuje soubory kaskádových stylů .css;
- `utils`. Obsahuje soubor s konstantami a soubor zajišťující komunikaci se serverem. Pro komunikaci se serverem byla použita knihovna `axios`.

6.3.2 Stylovaní

Jako CSS framework byl použit framework Material Design a jeho implementace v React.js s názvem Material UI [20]. Material Design je designový jazyk vyvinutý společností Google v roce 2014. Material Design rozšiřuje motivy „karet“, a více využívá mřížkových rozvržení, responzivní animace a přechody, polstrování a hloubkové efekty, jako jsou osvětlení a stíny [21]. Během stylovaní frontendu autor se inspiroval barevným rozvržením webu FELu: bílošedé pozadí s modrými prvky.

6.3.3 Filtrovaní výsledků

Na několika stránkách lze filtrovat výsledky. Na stránce `CommissonList` a `TeacherOverview` je možné vyfiltrovat komise podle následujících parametrů:

- Typ studijního programu (Bc, Ing, PhD);
- Studijní program (všechny aktuálně vyučované programy);
- Semestr nebo datum;

Na stránce manuálního vytváření komisí `ManualCreating` lze vytvořit komisi na základě parametrů popsaných výše. Podle těchto parametrů se potom v seznamu zobrazí vhodný učitel. Na všech stránkách, kde je možné zobrazit výpis seznamu učitelů, je implementováno filtrování podle jména, příjmení a felusername.

6.4 Seznam použitých nástrojů

Autor neřešil žádnou analýzu nástrojů, protože to nebylo nutné. Se všemi nástroji, které byly použity při vývoji se autor seznámil během svého studia.

JetBrains Intellij IDEA

JetBrains Intellij IDEA. IntelliJ IDEA je komerční vývojové prostředí (IDE) pro programování v jazycích Java, Groovy a dalších [12]. Kód (jak Java, tak i Javascript) autor psal v této IDE.

DBeaver

DBeaver je klientská softwarová aplikace SQL a nástroj pro správu databáze. Pro relační databáze používá rozhraní JDBC pro programování aplikací k interakci s databázemi pomocí ovladače JDBC. Pro ostatní databáze používá proprietární ovladače databáze [13]. Pomocí toho to nástroje autor nahlížel a případně opravoval databázi během vývoje aplikace.

Sparx Enterprise Architect

Enterprise Architect od společnosti Sparx Systems je kompletní CASE nástroj pro systémovou analýzu a návrh, který kompletně pokrývá cyklus vývoje systému, tzn. od zadání požadavků přes analýzu stavu, návrhu modelů, testování a údržbu, to vše s využitím diagramů v UML [14]. Veškeré diagramy, které byly nakreslené autorem, byly vytvořené pomocí Enterprise Architect.

Google Chrome

Browser Google Chrome Jeden z nejpopulárnějších internetových prohlížečů. Byl použit pro pozorovaní a debuggování frontendové části aplikace.

Postman

Postman je multiplatformní aplikace, která zvládá širokou škálu kroků od návrhu, přes testování, až po monitoring HTTP API [15]. Pomoci tohoto nástroje byly testovány vytvořené REST kontroléry na začátku vývoje, kdy ještě nebyla implementována frontendová část.

Pencil evolus

Pencil evolusje open-source nástroj pro prototypování GUI rozhraní. Pencil Evolus podporuje velký počet různých kolekcí tvarů určených nejen k prototypování desktopových rozhraní, ale také rozhraní mobilních platforem [16]. Za jeho hlavní výhodu autor považuje to, že Pencil Evolus podporuje exportování prototypu jako je HTML stránka s obrázky. Tento prvek umožnil autorovi editovat povahu prototypu po jeho exportování.

6.5 Autentizace a autorizace

Ideální by bylo, kdyby bylo přihlášení umožněno pomocí SSO [17]. Jedná se o stávající a dlouholetý funkční způsob přihlašování na ČVUT [9]. Výhod

použití SSO je několik:

- Není potřeba implementace přihlašovacího systému.
- Není nic jednoduššího, než využít toho, co již plně funguje.
- ČVUT SSO je propojený s dalšími systémy ČVUT, zejména s databází uživatelů a rozvrhem místností. Tato interakce by byla užitečná.

Autor v současnosti nemá žádná práva tohoto způsobu přihlášení využít a používá proto způsob, který není navázán na systémy ČVUT. Proto bylo rozhodnuto implementovat autorizaci uživatele pomoci JWT.

6.5.1 JWT

JSON Web Token (JWT) představuje způsob pro bezpečnou výměnu informací mezi dvěma stranami. JWT je JSON objekt, který se skládá z hlavičky (header), dat (payload) a podpisu (signature). JWT je definován podle specifikace **RFC 7519** [18]. Cílem JWT je možnost ověření autenticity dat [19]. Pro přihlášeni byla zvolena metoda autorizace pomocí JWT.

Použití

V běžném případě máme uživatele, aplikaci (aplikační server) a autentizační server. V rámci vytvářené aplikace ale roli autentizačního serveru přebírá samotná aplikace.

1. Uživatel se přihlásí (autentizuje) na serveru prostřednictvím loginu a hesla.
2. Server ověří identitu uživatele a vytvoří JWT, který poskytne uživateli odpověď na autentizační požadavek.
3. Uživatel použije JWT při komunikaci s aplikací. Každý http požadavek bude v hlavičce obsahovat token, který server bude validovat. Pokud s požadavkem přijde nevalidní token, uživatel bude přesměrován na stránku s přihlašovacím formulářem.

6.6 Změny provedené během vývoje oproti původnímu plánu

Změna počtu typu uživatelů

Na začátku projektu se plánovalo, že systém bude pracovat se třemi typy uživatelů: Admin, Manager a Teacher. Ale v průběhu vývoje bylo rozhodnuto zmenšit počet rolí na dvě: Manager a Teacher. Veškeré funkce Admina nově zahrnuje role Manager.

Změna typu přihlášeni

Na začátku projektu se k přihlášeni do aplikace plánovalo využití školního účtu (jako přístupové údaje do systému KOS). Ale během implementace bylo rozhodnuto implementovat autorizaci pomoci JWT tokenu.

6.7 Závěr

Tato kapitola obsahovala přehled architektury aplikace. Autor práce podrobně popsal strukturu kódu backendové a frontendové části aplikace. Dále byly vyjmenovány nastroje, které autor používal během implementace projektu. Na konci kapitoly je popsán způsob autorizace a vyjmenovaný změny v aplikace oproti původnímu návrhu.

Kapitola 7
Nasazení

Kapitola obsahuje proces instalace systému do zařízení fakulty. Obsahuje postup nasazeni aplikace na server FEL. V autor zmiňuje o zálohováni dat na vzdáleném serveru.

7.1 Prostředí

Začátkem dubna 2021 autor práce získal přístupové údaje na server ČVUT (`felk.cvut.cz`). Pro pohodlnou instalaci aplikace bylo na tomto serveru potřeba nainstalovat JDK (11 verzi) a aplikační server Apache Tomcat. Autor nainstaloval Tomcat verzi 9. Na serveru byl již nainstalovaný PostgreSQL.

7.1.1 Tomcat

Tomcat umožňuje deploy `.war` souboru. Aby bylo možné projekt (frontend a backend) zabalit do jednoho `.war` souboru, byly použity následující pluginy: `spring-boot-maven-plugin`, `maven-compiler-plugin`, `frontend-maven-plugin` a `maven-resources-plugin` (přesná konfigurace se nachází v souboru `pom.xml`).

Tomcat požaduje, aby se `.war` soubor nacházel ve složce `/opt/tomcat/webapps/`. Autor pro zjednodušení vytvořil symlink na `/home/student/app/fem.war`, aby se kvůli každému kopírování do `webapp` složky nemuselo používat rootovské heslo. Výchozí konfigurace Tomcatu se změnila pouze částečně: byl přidán uživatel s rolí Manager, avšak ani jednou nemusel být použit.

7.1.2 PostgreSQL

Na serveru byl předem nainstalován PostgreSQL verze 11. Běží na výchozím portu 5432. Autor manuálně vytvořil databázi s názvem `fem` a tím konfigurace databáze ukončil. Údaje potřebné pro připojení aplikace k databázi se nacházejí v souboru `application.properties`.

7.2 Postup nasazení

1. **Příprava prostředí.** Na serveru musí být nainstalován Tomcat, Java a PostgreSQL. Momentálně se používá Tomcat verze 9, Java 11 a Postgres verze 11. Pro instalaci je potreba vytvořit postgres uživatele se jménem **`student`** a tomu to uživateli udělit práva na vytváření databázi. Poté vytvořit databáze s názvem **`fem`**.
2. **Případná úprava zdrojového kódu.** Důležité je nezapomenout na soubor `application.properties`, ve kterém se konfiguruje většina důležitých proměnných používaných na backendu. Podobny soubor s důležitými konfiguracemi existuje i pro frontend s názvem `packages.json`.
3. **Build aplikace.** Byl vytvořen jednoduchý bash script, který postupně vyplňuje build frontendu, backendu a zasílá vytvořený `.war` soubor na server `student@fem.cvut.cz` (po buildu stačí jen zadat heslo). Script se nachází v rootovském adresáři zdrojového kódů (ve stejném adresáři, v němž je `pom.xml`). [7.2]
4. **Přesun souboru.** Na serveru přesunout novy `.war` soubor z `~/download` do `~/app`. Na serveru byl vytvořen symlink na `/home/student/app/fem.war`, aby se nemuselo kvůli každému kopoírování do `webapp` složky používat rootovské heslo. Počítá se s tím, že Tomcat je spuštěný. Pokud ne, je potřeba ho spustit pod rootem příkazem `/opt/tomcat/bin/start.sh`.

V následujícím listingu je ukázán skript, který vyplňuje build aplikace a posílá `fem.war` soubor na server.

```
#!/bin/bash
rm -rf src/main/frontend/build
cd src/main/frontend
npm run build
cd ../../../
mvn clean package
scp -r target/fem.war
   student@fem.felk.cvut.cz:/home/student/download
```

Listing 7.1: Script for build application

7.3 Zálohování

Pro jednoduché zálohování databáze byl přidán malý bash skript. Skript se nachází na `~/app/backup_db.sh`. Skript používá program `pg_dump` a ukládá databázi do `~/app/db_fem_dump/`. Pro obnovení databáze stačí použít program `psql` a to následujícím způsobem:

```
psql fem < infile
```

Listing 7.2: Command for restore database

Kde **fem** je nazev databaze, a **infile** je výstupní soubor programu `pg_dump`.

V následujícím listingu je ukázán skript na zálohovaní databáze:

```
#!/bin/bash

DATE_FORMAT="$(date +'%d.%m.%Y-%T')"
SCRIPT_DIR="$( cd "$( dirname "${BASH_SOURCE[0]}" )" &>
   /dev/null && pwd )"
full_path=${SCRIPT_DIR}/db_fem_dump/dump-${DATE_FORMAT}
echo "Creating database backup using pg_dump on: "
echo ${full_path}
pg_dump fem > "${full_path}"
```

Listing 7.3: Script for make database dump

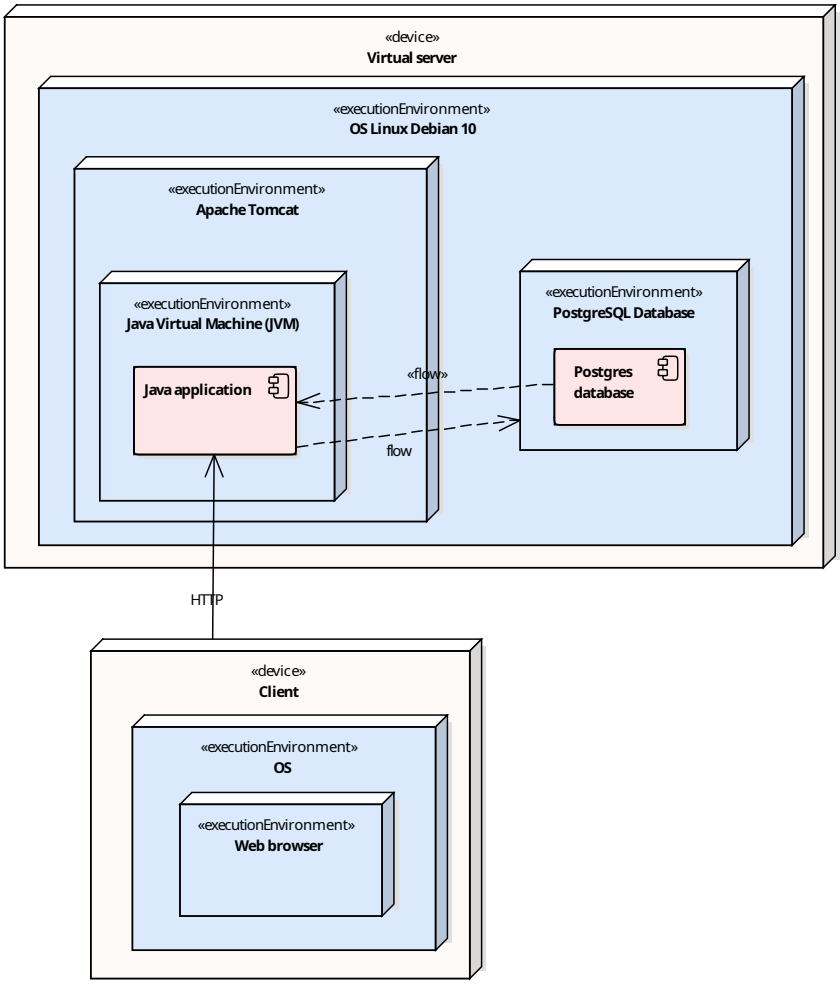
7.4 Architektura

Pro danou aplikaci autorem byla zvolena monolitická architektura.

Monolitická architektura je upřednostňována pro vývoj velmi malých, jednoduchých a lehkých aplikací. Protože monolitická architektura je považována za tradiční způsob vývoje aplikací, je vždy lepší mít stejné znalosti. Architektura Microservice je dobrá pro vývoj složitých aplikací. [22]

Monolitická architektura je považována za konvenční metodu vývoje aplikací. Aplikace v monolitické architektuře je vyvinuta jako jediný balíček. Vývoj normální aplikace začíná modulární vrstvenou architekturou.

I když má monolitická architektura logickou vrstvenou architekturu, konečné aplikace budou zabaleny do jediného monolitu a budou takto nasazeny. Monolitické aplikace mají pouze jednu kódovou základnu.



Obrázek 7.1: Diagram nasazení

Deployment diagram [7.1] znázorňuje architekturu nasazení aplikace na server FEL.

Kapitola 8
Závěr

8.1 Shrnutí

Cílem této práce bylo navrhnout, implementovat a nasadit aplikaci pro sestavování státnicových komisí. Během práce nad projektem autor prošel všemi částmi a kroky vytváření webové aplikace. V roli zákazníka projektu vystupoval sekretariát Fakulty Elektronické ČVUT. Na začátku byly definovaný systémové požadavky. Poté na základě definovaných požadavků autor navrhnul klikací prototyp aplikace. Následně bylo provedeno uživatelské testování prototypu, byly analyzovány příslušné výsledky a některé náměty účastníků byly přímo zahrnuty do aplikace. Po testování započala implementace. Implementace byla dlouhá a občas náročnější. Backendová a frontendová část se vyvíjely souběžně, což zároveň umožnilo testování nových funkcí. S blížící se poslední fází implementace se autor začal pokoušet nasazovat aplikaci na server.

Zadaní se považuje za splněné a projekt je téměř hotový. Bez zvláštních problémů probíhala implementace backendu. Obtížnější částí byla implementace frontendu, a to především z důvodu, že komplexní UI za použití React.js frameworku autor programoval poprvé. Bylo nutné získat mnoho nových znalostí. K věcem, které se nepodařily, by autor přiřadil odhad času na splnění jednotlivých úkolů. Tyká se to především frontendové části a důvodem je, že React.js je pro autora nová technologie.

Hlavní přínos práce autor spatřuje ve významném osobním pokroku v programovací činnosti a v získání nových znalostí v oblasti programovaní fullstack aplikace, které bude s velkou pravděpodobností moct využít v praxi v nejbližší budoucnosti.

8.2 Výhled do budoucna

Implementovaná aplikace funguje a splňuje funkční a nefunkční požadavky, lze ji ale vylepšit. Například implementovat přihlášení pomocí jiných metod, třeba s využitím školního účtu. Aplikaci bych také šlo propojit s databází FEL. Určitě by bylo možné synchronizovat databázi uživatelů, místností a studijních programu s databází vyvinuté aplikace. Dalším prvkem aplikace

by mohla být funkce přiřazování studentů ke zkušebním komisím.

Příloha A

Literatura

[1] Codeburst. Top 5 backend frameworks https://codeburst.io/top-5-backend-frameworks-2021-f3a73a18aa5f, 2020, [online].

[2] Prasad Reddy, K. Siva. *Beginning Spring Boot 2: Applications and Microservices with the Spring Framework*, Apress, 2017, ISBN 9781484229316.

[3] Convention over configuration https://facilethings.com/blog/en/convention-over-configuration, [online].

[4] Laravel LLC. Laravel framework https://laravel.com/, 2021, [online].

[5] Codeinwp. Porovnání frontend frameworku https://www.codeinwp.com/blog/angular-vs-vue-vs-react/, 2021, [online].

[6] State of Js. Front-end Frameworks comparison https://2020.stateofjs.com/en-US/technologies/front-end-frameworks/, 2020, [online].

[7] Dan Abramov and Andrew Clark. React Redux https://redux.js.org/introduction/getting-started, [online].

[8] Dan Pilone, Neil Pitman. *UML 2.0 in a Nutshell*, O'Reilly Media, Inc., 2005, ISBN 0596007957, 9780596007959

[9] České Vysoké Účetní Technické. SAML SSO https://ist.cvut.cz/nase-sluzby/single-sign-on/, 2019, [online].

[10] Uživatelské testování: výběr respondentů https://medium.com/house-of-%C5%99ez%C3%A1%C4%8D/u%C5%BEivatelsk%C3%A9-testov%C3%A1n%C3%AD-2-v%C3%BDb%C4%9Br-respondent%C5%AF-c7d34476c7c1

[11] Fowler, M. *Patterns of Enterprise Application Architecture*, Pearson Education, 2012, ISBN 9780133065213.

[12] JetBrains s.r.o. Intellij IDEA https://www.jetbrains.com/idea/, [online].

[13] DBeaver Community. DBeaver Universal Database Tool https://dbeaver.io/, 2021, [online].

[14] Sparx Systems. Enterprise Architect https://sparxsystems.com/, 2013 [online].

[15] Postman. Postman API Client https://www.postman.com/, 2021, [online].

[16] Pencil project. Pencil Evolus https://pencil.evolus.vn/, 2019,[online].

[17] Ado Kukic. *The Definitive Guide to Single Sign On*, OAuth, 2019.

[18] RFC 7519 Specification https://datatracker.ietf.org/doc/html/rfc7519, 2015, [online].

[19] Zoom.cz. JSON Web Tokens(JWT), https://zooom.cz/json-web-tokens-jwt/, 2019, [online].

[20] Material-UI. Material-UI the world's most popular React UI framework https://material-ui.com/, 2014 [online].

[21] Google, LLC. Material design https://material.io/design, 2014, [online].

[22] O. Al-Debagy, P. Martinek. *A Comparative Review of Microservices and Monolithic Architectures* 2018 IEEE 18th International Symposium on Computational Intelligence and Informatics (CINTI), 2018, pp. 000149-000154, doi: 10.1109/CINTI.2018.8928192.

[23] Klaus Pohl. *Requirements Engineering Fundamentals, 2nd Edition: A Study Guide for the Certified Professional for Requirements Engineering Exam - Foundation Level - IREB compliant*, Rocky Nook, Inc., 2016, ISBN 1937538842, 9781937538842.

Příloha B

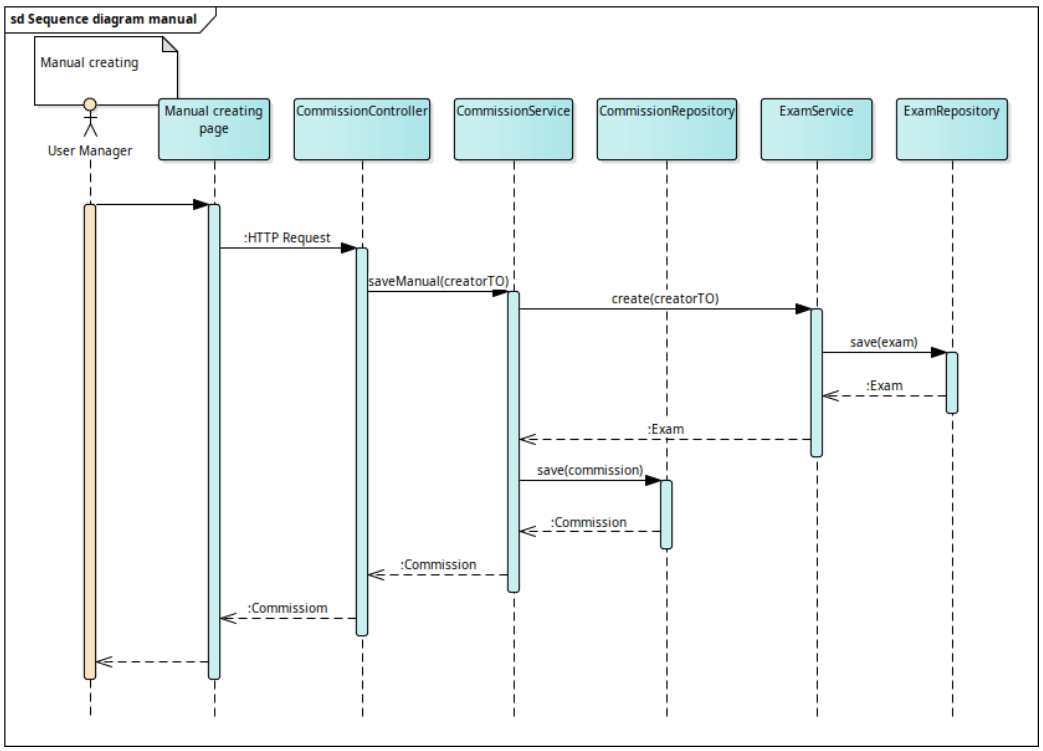
Seznam zkratek

ČVUT	České Vysoké učení technické v Praze
FEL	Fakulta elektrotechnická
KOS	Studijní informační systém
SSO	Single Sign-On
JWT	Jason Web Token
CASE	Computer Aided Software Engineering
JDK	Java Development Kit
DB	Database
SQL	Structured Query Language
JDBC	Java Database Connectivity
NPM	Node Package Manager
CSV	Comma-separated values
XLS	přípona souborů specifikace Office Open XML vytvořených v aplikaci Microsoft Excel
HTTP	Hypertext Transfer Protocol
API	Application Programming Interface
GUI	Graphical User Interface
REST	Representational State Transfer
CRUD	Create, Read, Update, Delete
DAO	Data Access Object
MVC	Model, View, Controller
UML	Unified Modeling Language

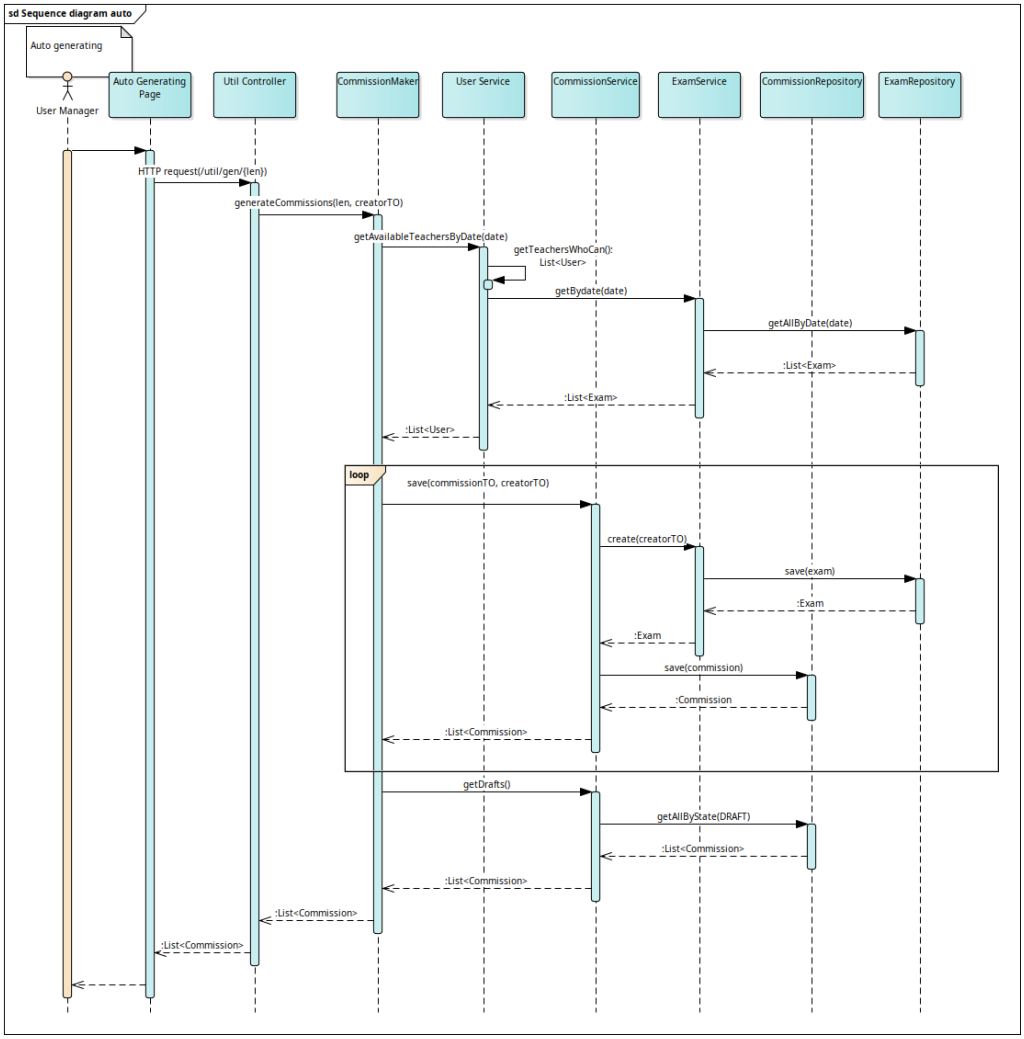
Příloha C

Diagramy a obrázky

C.1 Sekvenční diagramy

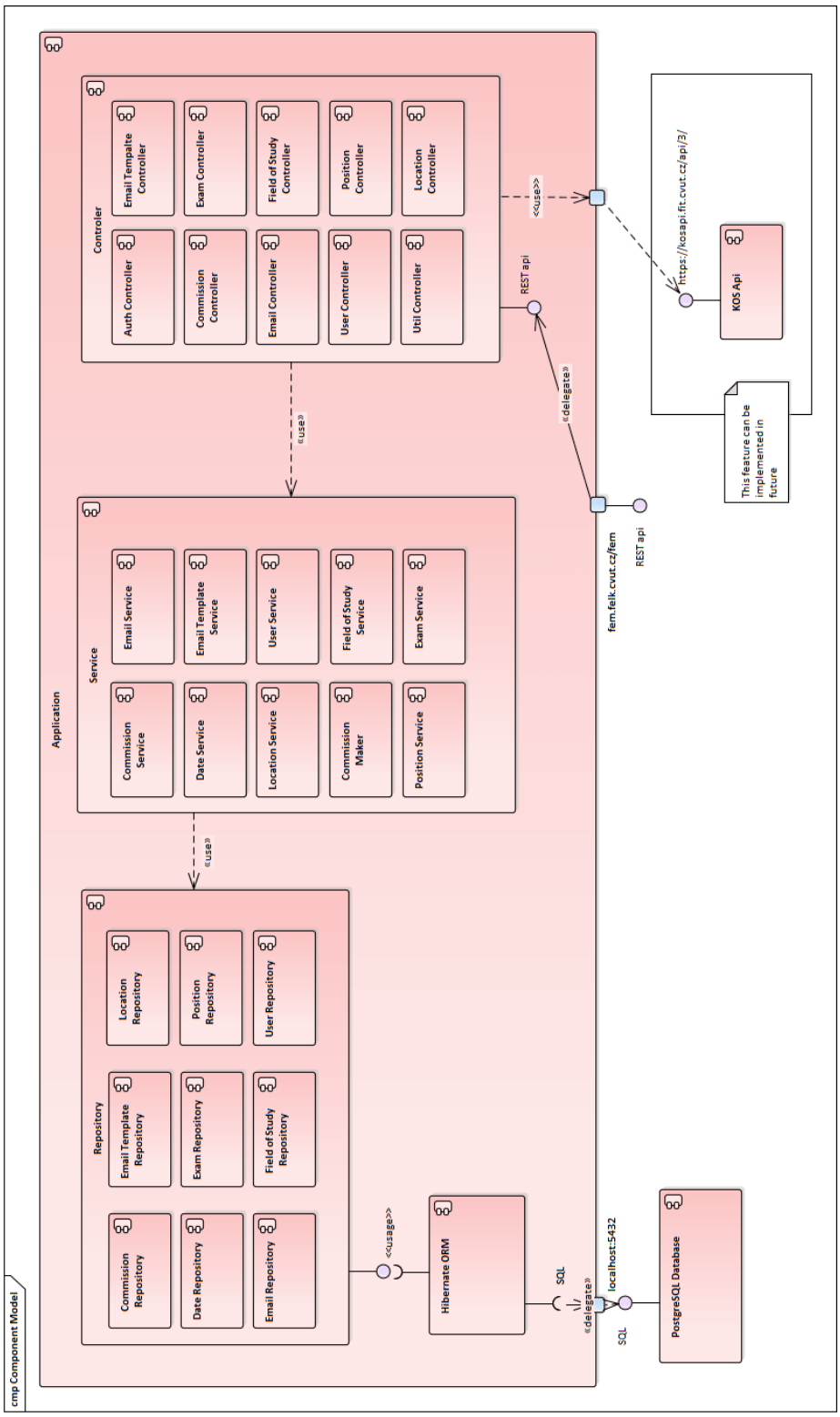


Obrázek C.1: Sekvenční diagram. Manuální vytvářeni komisi



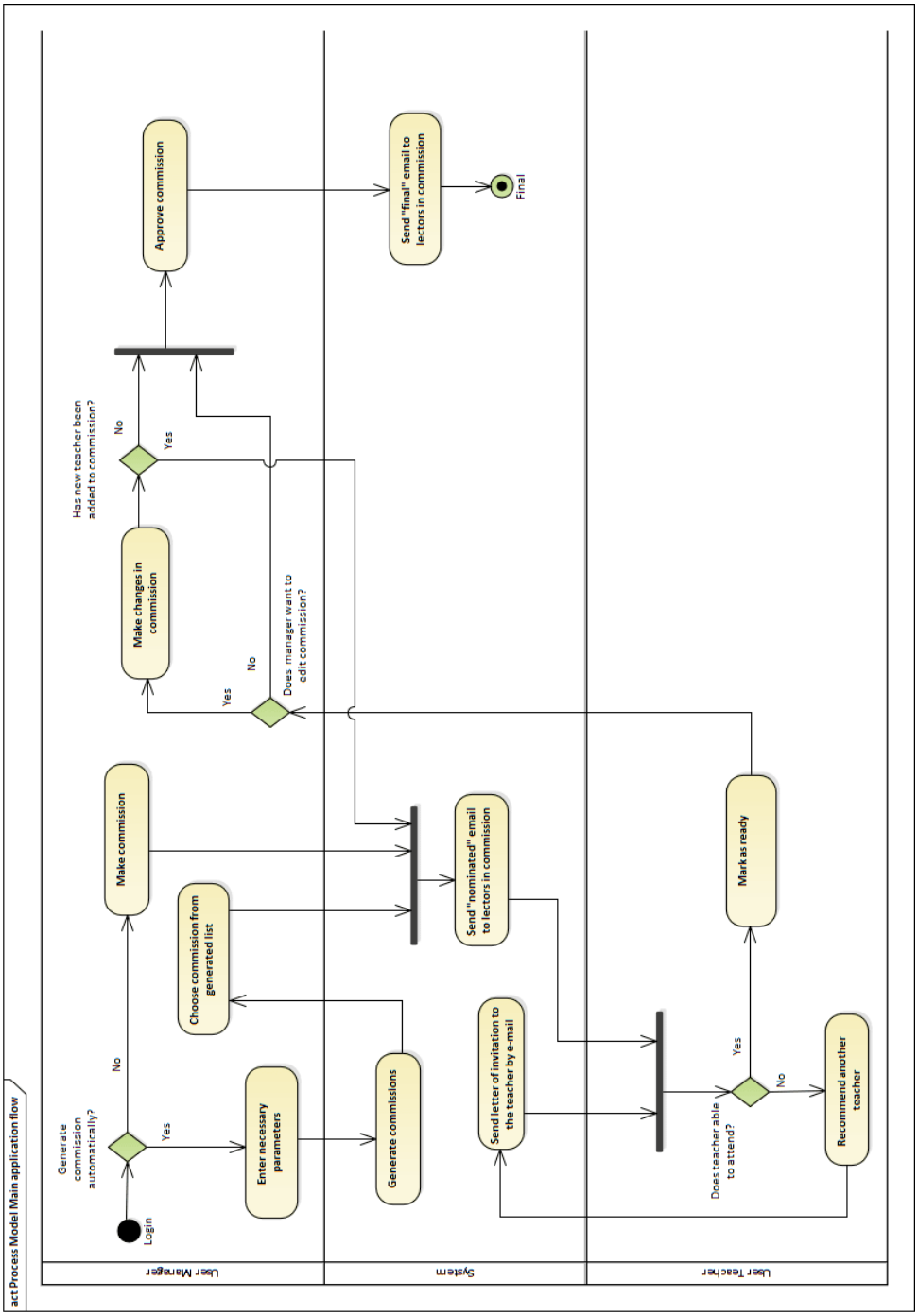
Obrázek C.2: Sekvenční diagram. Automatické vytváření komisi

C.2 Diagram komponent



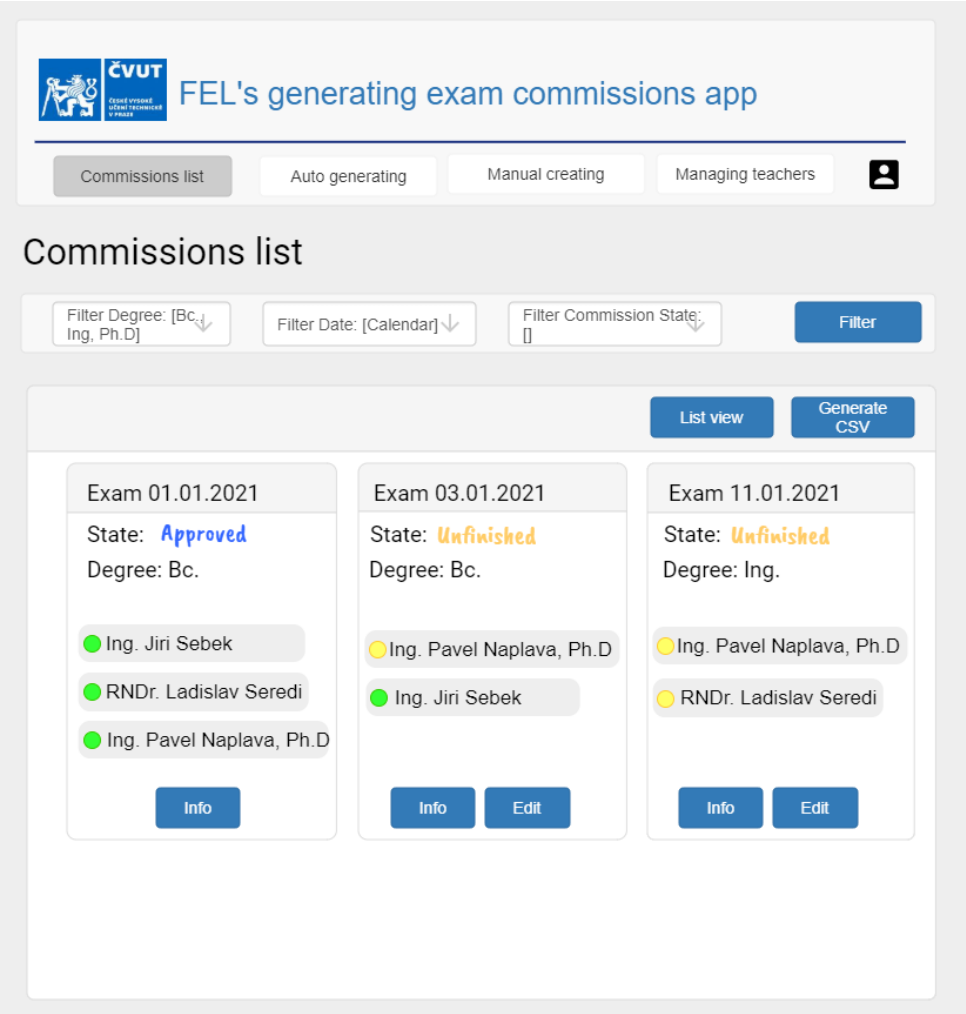
Obrázek C.3: Komponent diagram

C.3 Procesní diagram

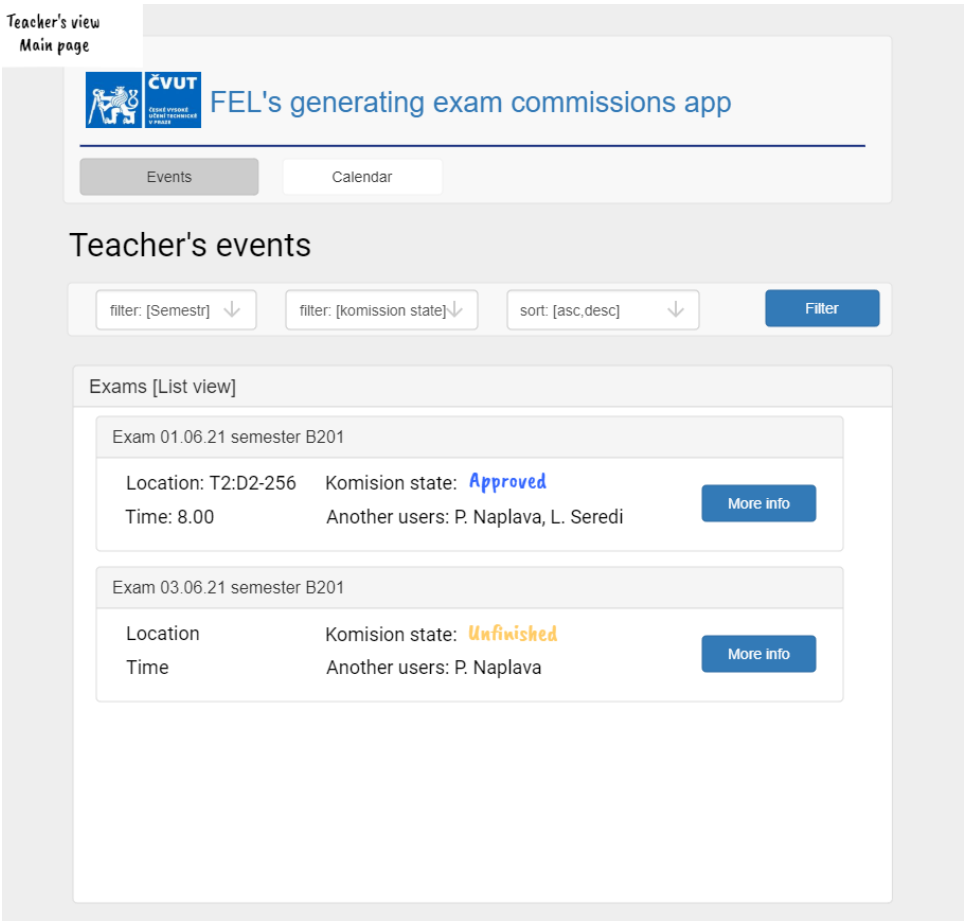


Obrázek C.4: Procesní diagram

C.4 Prototyp



Obrázek C.5: Prototype. Commissions List page



Obrázek C.6: Prototype. Teacher's main page