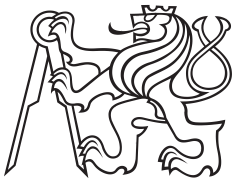
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Systém pro správu státnicových okruhů

Nikita Iryupin

Vedoucí: Ing. Jiří Šebek
Studijní program: Softwarové inženýrství a technologie
Květen 2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Iryupin** Jméno: **Nikita** Osobní číslo: **495544**

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 okruhů

Název bakalářské práce anglicky:

Application for management final exam topics Intended for Bachelor Field of study

Pokyny pro vypracování:

Bakalářská práce se bude zabývat procesem sestavování státnicových komisí. Tento proces je potřeba optimalizovat.
Systém bude muset umět:
zařazení daného člena (prof. docent, a zbylé pozice - doc. a prof. jsou nutné pro predsedovanie), schválení vědeckou radou
zaměření (pro kterou komisi je vhodný neboli obor), úvazek, kombinace různých pracovišť
Systém bude nasazen s reálnými daty.
Jedná se o pokračování předchozí práce a její zlepšení.
Provede se další vývoj (napojení na školní API) a testování reálného systému.
Práce bude schválena zákazníkem a dostatečně otestována.

Seznam doporučené literatury:

Walls C. Spring Boot in action. – Simon and Schuster, 2015.
Luo L. Software testing techniques //Institute for software research international Carnegie mellon university Pittsburgh, PA. – 2001. – T. 15232. – №. 1-19. – C. 19.
Aggarwal S. Modern web-development using reactjs //International Journal of Recent Research Aspects. – 2018. – T. 5. – №. 1. – C. 133-137.

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: **14.02.2022** Termín odevzdání bakalářské práce: **20.05.2022**

Platnost zadání bakalářské práce: **19.02.2024**

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

Děkuji Ing. Jiřímu Šebkovi za vedení mé bakalářské práce, za podporu během a za poskytnuté konzultace

Dále děkuji paní Kateřině Maršálkové - sekretářce z katedry počítačů za spolupráci, rychlou reakci a snahu systém zlepšit.

Prohlášení

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

V Praze, 20. května 2022

Abstrakt

Bakalářská práce se zabývá rozšířením a zlepšením predchozí práce - systému pro vytvoření státnicových komisí. Provedl se refactoring, proběhla upřesňující analýza a rešerše. Projekt je schválen zákazníkem a bude se používat příští semestr.

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

Vedoucí: Ing. Jiří Šebek
Praha, Resslova 307, E-431

Abstract

The bachelor thesis deals with the extension and improvement of the previous work - the system of management of final exam topics. Refactoring was performed, as is a detailed analysis. The project is approved by the customer and will be used next semester.

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

Title translation: Application for management final exam topics

Obsah

1 Úvod 1

1.1 Motivace 1

1.2 Cíle 1

2 Sběr požadavků 3

2.1 Funkční požadavky 3

2.2 Nefunkční požadavky 3

3 Rešerše 7

3.1 Reserse již existujících řešení 7

3.1.1 Informační systémy na VŠ a jejich funkcionality 7

3.1.2 Informační a studijní aplikace na ČVUT 8

3.1.3 Další řešení pro školní IS 8

3.1.4 Závěr 9

3.2 Rešerše technologií pro vývoj webových aplikací 9

3.2.1 Backend 9

3.2.1.1 Spring Boot 9

3.2.1.2 Django 10

3.2.1.3 CakePHP 10

3.2.1.4 Flask 11

3.2.1.5 Ruby on Rails 11

3.2.1.6 Laravel 12

3.2.1.7 Závěr 12

3.2.2 Frontend 12

3.2.2.1 React 13

3.2.2.2 Vue.js 13

3.2.2.3 Angular 14

3.2.2.4 jQuery 14

3.2.2.5 Závěr 14

3.2.3 Závěr analýzy 15

4 Analýza 19

4.1 Analýza původní práce 19

4.1.1 Možné změny datového modelu původní práce 19

4.1.2 Analýza funkčních požadavků 20

4.1.3 Analýza nefunkčních požadavků 22

4.2 Analýza kódu 22

4.3 Analýza změn a rozšíření 24

4.3.1 Studenti v systému 24

4.3.2 KosAPI a SSO API 24

5 Návrh 27

5.1 Entity 27

5.2 Mailing 27

5.3 Role v systému 27

5.4 Architektura 28

6 Testování 31

6.1 První fáze testování 31

6.2 Druhá fáze testování 32

7 Implementace 35

7.1 Úprava datového modelu 35

7.2 Testovací nasazení na lokálním prostředí 36

7.3 Vytvoření profilu pro lokální a produkční nasazení 36

7.3.1 React.js 36

7.3.2 Spring Boot 37

7.4 CORS 37

7.5 Nastavení správných oborů pro zkoušení 38

8 Nasazení **39**

8.1 Prostředí 39

8.2 Postup nasazení 39

9 Závěr **41**

9.1 Shrnutí 41

9.2 Perspektivy dalšího rozšíření . . . 41

A Literatura **43**

Obrázky

2.1 Původní Use Case Diagram 4

2.2 Aktualní Use Case Diagram 5

4.1 Datový model uživatele 20

4.2 Příklad kódu s proměnnou přímo v kódu 23

4.3 Nepoužitý kód 24

4.4 SSO vs Local [sso22] 25

5.1 Datový diagram 29

5.2 Posílání emailů přes SMTP server [Agg21] 29

7.1 Agilní vývoj [Gof22] 35

7.2 Nový model pro uživatele 36

Tabulky

3.1 Analyza backendů 16

3.2 Analyza frontendů 17

Kapitola 1

Úvod

1.1 Motivace

Podle výzkumu se v dnešní době velká část projektů nedostane ani do fáze reálného použití, nebo se daná aplikace používá jen krátkou dobu po dokončení vývoje.[tea22][RF07] Přesto část úspěšných aplikací je příliš náročná na údržbu kvůli chybám při implementaci. Žádný programátor nechce, aby jeho projekty dopadly stejně a kvůli tomu hodně úsilí zmizelo v propadlišti dějin, proto se bude autor v této práci zabývat dalším rozvojem již existující aplikace.

Aplikace, které se věnuje tato bakalářská práce, už je nasazená na serveru. Původně to byla bakalářská práce studenta Dmitrie Gritsaie z roku 2021[GD21]. Jedná se o systém pro správu státnicových okruhů, který funguje, ale může být vylepšen. Motivací pro dodělání této práce je, že pořád není ve stavu, který by uspokojil katedru počítačů kvůli možným vylepšením. Kvůli tomu původní aplikace nebyla použita a pečlivě otestována.

1.2 Cíle

Cílem této bakalářské práce je analýza a úprava hotového řešení. Analýza zahrnuje zkoumání kódu, architektury systému, funkčních a nefunkčních požadavků, odhalení co odpovídá původnímu záměru studenta Gritsaie, a co ne. Také bude uskutečněn rozbor změn a zlepšení, které chce realizovat autor. Některé z těchto změn vznikly na základě zpětné vazby o systému od jeho uživatelů. Její výsledky byly formulovány do nových funkčních a nefunkčních požadavků a jsou uvedené v následující kapitole 2

Též do nich patří hlavně rozšíření aplikace o napojení na SSO a KosAPI. Kromě analýzy autor provede refaktorování kódu. Zdrojový kód obsahuje

TODO a zakomentované části – ty bude potřeba odstranit nebo předělat. Schválení ze strany klienta proběhne až po testování funkční aplikace, což bude také součástí této práce.

Kapitola 2

Sběr požadavků

Jak bylo uvedeno v podkapitole 1.2, původní aplikace nebyla použita. Komise se doteď sestavují, jak uvádí student Gritsai, v Excelu. Zabírá to mnohem více času, než by mělo, právě kvůli ručnímu sestavení.

Následující podkapitoly obsahují funkční a nefunkční požadavky, které budou přidané do systému v rámci jeho zlepšení. (obrázek 2.2) Vznikly na základě testování a snahy použít systém pracovníkem vytvářejícím státnicové komise a paní sekretářkou.

Detailní ověření jednotlivých požadavků (obrázek 2.1) z práce studenta Gritsaie (jaké z nich byli realizované v původní aplikaci a jaké naopak ne) je provedeno v kapitole 4.

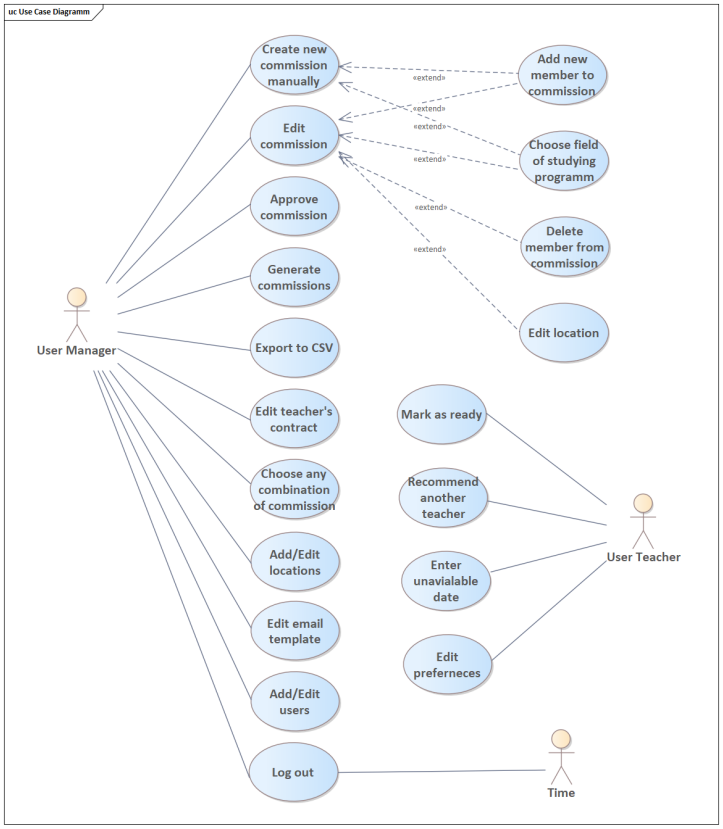
2.1 Funkční požadavky

- SSO přihlášení – systém bude umožňovat uživateli přihlášení přes SSO.
- Podrobné informace o členech komise – systém bude umožňovat uživateli vidět veškeré informace o vyučujících: tituly, role v komisi, obory, které zkouší, a zda dotyční již byli v komisi v minulém a předminulém semestru.
- Přidání/Změny nových oborů pro zkoušení – systém bude umožňovat uživateli přidat nebo změnit obory, které zkouší komise.

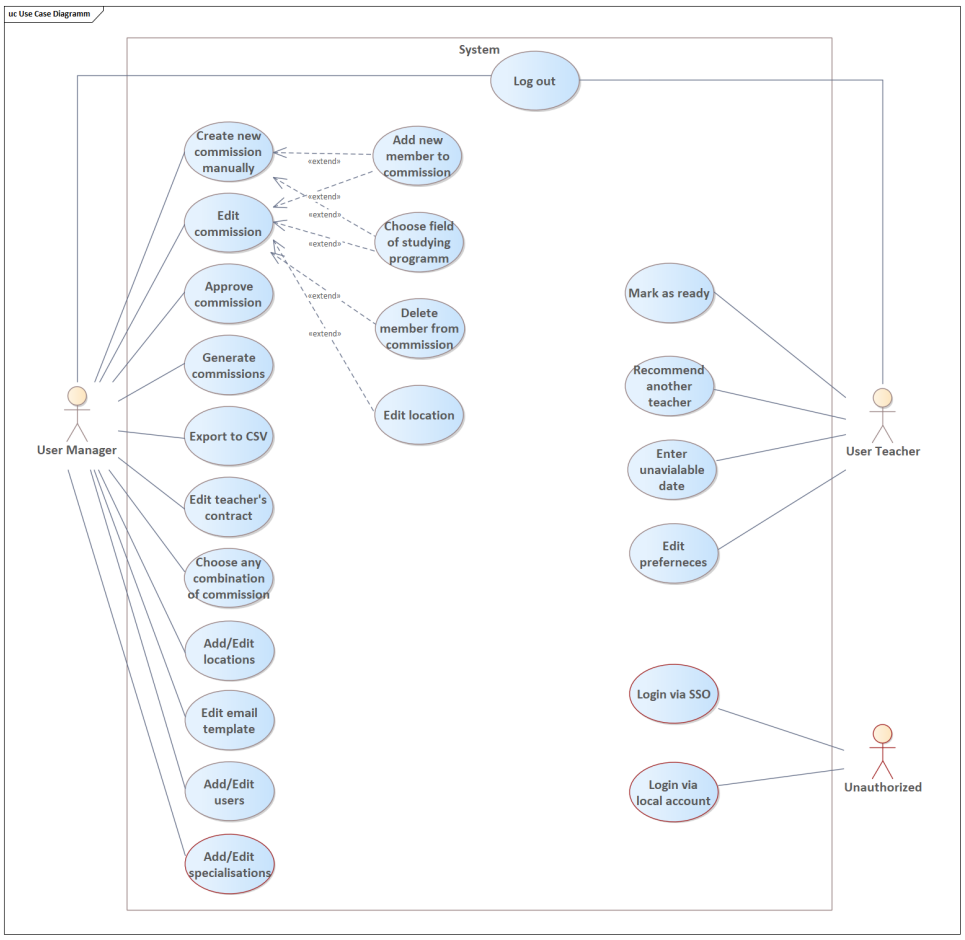
2.2 Nefunkční požadavky

- Přátelský a lehce pochopitelný design aplikace - všechny záložky budou umístěni logicky tak, aby je uživatel nehledal dlouho. Celkový design aplikace je vyroben tak, aby se noví uživatelé mohli snadno zorientovat v aplikaci.

- KOŠ API - systém bude využívat KOŠ API pro práci s databází a daty univerzity.
- Ochrana osobních údajů uživatelů - Všechna citlivá data uživatelů budou před uložením hashovaná a osolená.



Obrázek 2.1: Původní Use Case Diagram



Obrázek 2.2: Aktualní Use Case Diagram

Kapitola 3

Rešerše

Tato kapitola se zabývá rešerši již existujících řešení a rešerší technologií pro vývoj webových aplikací. Jelikož pan Gritsai ve své původní práci už základní rešerše udělal, autor ji jen rozšíří o statistiku a doplňující informace týkající se technologií použité v systému pro správu státnicových okruhů.

Navíc, v podkapitole 3.1 autor práce blíž prozkoumá informační systémy na největších vysokých školách v ČR a porovná jejich funkcionality. Také se podívá na hlavní aplikace ČVUT a jaké existují alternativní řešení pro informační školní systémy.

3.1 Reserse již existujících řešení

Hlavním cílem rešerše je zjistit, zda jedna z nich má nějaký způsob sestavení komise pro SZZ.

3.1.1 Informační systémy na VŠ a jejich funkcionality

Vysoke skoly v ČR využívají pro své cíle různé informační systémy.

- UK, VŠCHT, Policejní akademie ČR a další využívají především IS STUDIUM (SIS) společnosti ERUDIO [IJM14]. SIS je hlavním informačním systémem pro výše uvedené univerzity a umožňuje studentům vytvářet rozvrhy a sledovat své výsledky. Vyučující a pracovníci mohou vypisovat termíny zkoušek, dávat známky a také i vytvářet subkomise pro SZZ.
- VŠE, Mendelova univerzita v Brně, Škoda Auto Vysoká škola v Mladé Boleslavi, Vysoká škola obchodní v Praze, o.p.s., Česká zemědělská univerzita v Praze, NEWTON College, a.s., Vysoká škola podnikání

a práva využívají informační systém UIS poskytnutý společnosti IS4U [s.r22].

Konkrétně na VŠE se tento IS jmenuje Integrovaný studijní informační systém, ve zkratce InSIS. Seznam funkcionalit InSIS je popsán v oficiální dokumentaci. [dt22] Systém poskytuje úpravu přihlášek, monitorování přijímacího řízení pro absolventy, sledování harmonogramu akademického roku, rozvrhu a studijních výsledky, zapisování se na předměty a zkoušky a další. Avšak žádnou informaci o tom, zda InSIS má nějaké nástroje pro sestavení komise na SZZ, autor práce na oficiálních stránkách VŠE nenašel.

- Západočeská univerzitu v Plzni, Technická univerzita Liberec, Univerzita Pardubice, Univerzita Tomáše Bati ve Zlíně a několik dalších vysokých škol mají studijní systém IS/STAG vyvinutý na Západočeské univerzitě v Plzni, který je považován za nejpoužívanější podobný systém. Poskytuje univerzitám, které ho používají, kromě základních funkcionalit také možnost sestavovat komise na SZZ.

3.1.2 Informační a studijní aplikace na ČVUT

ČVUT využívá několik různých aplikací – KOS, CourseWare, Moodle a vlastní informační systém IS ČVUT. Poslední slouží pro sdílení informace pro studenty, zaměstnance a partnery.

- KOS je hlavním systémem pro evidenci studijních výsledků, zkoušek, rozvrhů a dalších školních záležitostí. Nicméně nemá žádné nástroje pro sestavení komise na SZZ.
- CourseWare a Moodle jsou nástroje pro výuku – obsahují hlavně studijní materiály. Moodle ČVUT využívá open source

Na rozdíl od několika IS jiných vysokých škol uvedených výše ČVUT nemá žádnou aplikaci pro sestavení komise na SZZ, pracovníci to musejí dělat "ručně".

3.1.3 Další řešení pro školní IS

Pro IS vysokých škol existuje možnost použít CMS (z anglického content management systém) neboli systém pro správu obsahu pro univerzity, např. WordPress a Drupal.

Dalším možným řešením je objednání IS u firmy se zaměřením na vytváření takových systémů, např. . V podkapitole 3.1.1 bylo uvedeno o systém firmy IS4U.

Ale obě varianty mají nevýhodu v tom, že my chceme mít vlastní řešení, které budeme moct upravovat a integrovat samostatně.

3.1.4 Závěr

Výsledkem řešení je, že univerzity mají různé informační systémy, ale jen některé z nich umožňují sestavení komisí na SZZ. ČVUT používá několik aplikací pro různé účely, avšak žádná z nich nemá funkcionality jako seskupení učitelů a externistů, což komplikuje už tak náročný proces SZZ.

3.2 Rešerše technologií pro vývoj webových aplikací

Tato podkapitola se věnuje rešerši technologií pro vývoj webových aplikací, jejich srovnání a výběru nejlepších pro vývoj velkých webových aplikací, kterým je i systém pro správu státnicových okruhů. Pro backend a frontend byly provedené nezávislé rešerše v samostatných podkapitolách.

3.2.1 Backend

Pro vývoj a rozvoj webových aplikací je nezbytný backend – je to vrstva odpovídající za práci s hardwarem, serverem, daty databáze apod. V dnešní době je velké množství různých frameworků a nástrojů pro práci s backendem: Spring Boot, CakePHP, Django, Ruby, Flask, Express JS, Laravel a jiné. Většina z nich jsou už dávno zastaralé, některé nevyhovují pro náš projekt jinými vlastnostmi. Avšak podrobně jsme se podívali na největší a nejpopulárnější z nich – zvážili jsme jejich silné a slabé stránky a vybrali jsme ten, který se nám nám hodí nejvíc.

3.2.1.1 Spring Boot

Spring Boot je open-source framework v jazyce Java.

Výhody Spring Boot jsou

- flexibilní a jednoduchá konfigurace
- interní analýza chyb
- vhodný pro mikroservisy a monolitní aplikace
- přímá podpora Tomcat, Jetty a Undertow

- popularita: více než polovina respondentů (62) využívá ve svých projektech SptingBoot. [s.r21] Popularita znamená početnou komunitu, což s sebou přináší velké množství studijních materiálů v podobě knih, tutoriálů, článků apod. V důsledku toho práce je mnohokrát lehčí, příjemnější a hlavně produktivnější.
- vlastní zkušenosti: autor práce používal framework ve vlastních projektech, i v rámci předmětů, vyučovaných během studia na oboru
- jednoduchá práce s DB díky nativní JDBC[GKS16] a JPA[KSN18].
- integrované testování

Mezi nevýhody patří, že aplikace používající Spring Boot jsou občas obsáhlé a kvůli tomu pomalé.

3.2.1.2 Django

Django je open-source framework v jazyce Python. Drží se MVC architektury.

Výhody:

- zvýšená bezpečnost
- aplikace napsané pomocí Django jsou rychlé
- zabudované nástroje autentizace a autentifikace, RSS feeds, administrování obsahu a jiné nezbytné pro webový vývoj
- univerzálnost: Django je používán pro tvorbu jak sociálních sítí tak i pro aplikace velkých byznys organizací
- Django je multiplatformní software

Nevýhody:

- pomalý vývoj frameworku
- je určen spíše pro velké monolitní aplikace

3.2.1.3 CakePHP

CakePHP je nejvýznamnější PHP webový open-source framework. Je založen na konceptech frameworku Ruby on Rails a následuje architekturu MVC.

Výhody:

- MVC architektura
- podpora AJAX, HTML formulářů, JavaScript View Helpers
- komponenty pro Cookie, Session, Email a Request Handling
- CRUD funkcionalita

Nevýhodou je, že začátečníci mohou narazit na problémy

3.2.1.4 Flask

Flask je open-source micro framework v jazyce Python.

Výhody:

- flexibilita a díky tomu jednoduchý vývoj jak malých, tak i velkých webových aplikací
- škálovatelnost pro různé projekty
- jednoduchý syntax

Nevýhody:

- pro práci s DB a velkými soubory je třeba ručně přidávat speciální rozšíření
- omezenost – přestože Flask je používán pro aplikace různých velikostí, občas nemá potřebné funkcionality
- je určen spíše pro menší projekty

3.2.1.5 Ruby on Rails

Webový framework v jazyce Ruby.

Výhody:

- MVC architektura
- jednoduché testování kódu
- rychlý vývoj aplikací
- ochrana osobních údajů uživatelů budoucích aplikací vytvořených na Ruby on Rails

Nevýhody:

- chybí podrobná a jasná dokumentace
- složitý pro začátečníky
- možné problémy při vývoji na OS windows kvůli potřebě instalovat rozšíření navíc

3.2.1.6 Laravel

Laravel je open-source framework v jazyce PHP založený na architektuře MVC.

Výhody:

- MVC architektura
- jednoduché pro testování
- velké množství doplňkových nástrojů

Nevýhodou je problémová kompatibilita – občas nové verze jsou nekompatibilní se starými.

3.2.1.7 Závěr

V následující tabulce jsou znázorněné slabé a silné stránky jednotlivých výše uvedených frameworků a byla přidána informace navíc.

Závěrem podle provedeného výzkumu je zřejmé, že nejvhodnějším backend frameworkem je Spring Boot, protože ne jenom samotný framework má hodně silných stran, ale samotná aplikace už je napsaná v Spring Boot.

3.2.2 Frontend

Žádná webová aplikace se neobejde bez kvalitního frontendu. Frontend je zodpovědný převážně za rozhraní, celkový vzhled a tím pádem i zkušenost uživatelů. Existuje velké množství frameworku pro vývoj rozhraní, v této podkapitole je uveden rozbor nejvýznamnějších z nich.

3.2.2.1 React

React je open-source knihovna pro jazyk JavaScript.

Výhody:

- jednoduchý pro osvojení – hlavně pro ty, co už mají předešlé zkušenosti s JavaScript. Navíc tento framework má podrobnou dokumentaci, velkou komunitu a hodně studijních materiálů [Ber17], díky své široké oblíbenosti
- popularita: podle průzkumy servisu StackOverflow React několik let za sebou byl nejpopulárnějším frontend frameworkem a dodnes se drží vedoucích pozic
- garantuje stabilitu
- znovupoužitelnost jednotlivých komponentů
- flexibilita
- vlastní virtuální Document Object Model, což zrychluje [Jav19] a zjednodušuje operace v UI aplikace

Nevýhodou je - složité koncepty JSX syntaxe, které nejsou určené pro začátečníky. JSX je speciální rozšíření JavaScript umožňující strukturovat komponenty

3.2.2.2 Vue.js

Vue.js je druhý nejpopulárnější open-source JavaScript framework

Výhody:

- virtuální DOM
- podrobná dokumentace a elegantní syntax
- jednoduchá integrace s jinými frameworky a knihovnami, podpora TypeScript

Nevýhody:

- malé komunity
- jazyková bariéra mezi různými pluginy a komponenty – většina z nich je napsána v Čínštině
- nedostatek stability komponentů

3.2.2.3 Angular

Angular je framework v jazyce TypeScript.

Výhody frameworku:

- MVC architektura
- tzv. two-way data binding, který zajišťuje okamžitou synchronizaci změn ve view a v model částech MVC architektury
- velká komunita a podpora

Nevýhody u Angularu jsou následující:

- aplikace v Angularu jsou poměrně těžké a kvůli tomu i pomalé
- je složitý pro osvojení

3.2.2.4 jQuery

jQuery je nejstarší z frameworků pro frontend, lze říci, že se jedná o knihovnu pro JavaScript.

Výhody:

- jednoduchost v naučení a používání
- kompatibilnost se všemi prohlížeči
- podpora DOM

Nevýhody:

- zastaralé DOM API
- dynamické jQuery aplikace jsou pomalejší narozdíl od jiných variant

3.2.2.5 Závěr

V následující tabulce jsou znázorněné slabé a silné stránky jednotlivých výše uvedených frameworků a byla přidána informace navíc. Závěrem dle provedeného výzkumu je to, že nejvhodnějším frameworkem pro práci s rozhraním je React kvuli široké podpoře a znovupoužitelnosti komponentu.

3.2.3 Závěr analýzy

Jak už bylo uvedeno v jednotlivých podkapitolách nejlepšími volbami z frameworků pro náš projekt jsou Spring Boot pro backend a React pro frontend [Hin18].

Ke stejnému závěru přišel i student Gritsai ve své původní práci, proto se nebude při implementaci měnit framework nebo jazyk programování.

Framework	Programovací jazyk	Výhody	Nevýhody	Známá využití
Spring Boot	Java	flexibilní konfigurace, interní analýza chyb, mikroservisy, popularita, zkušenosti, nativní JDBC a JPA, integrované testování	obtížně a pomalé aplikace	Trivago, Intuit, Udemy
Django	Python	bezpečnost, rychlost, zabudované nástroje, univerzálnost, je multiplatformní	pomalý vývoj, pro velké monolitní aplikace	Spotify, Dropbox, Pinterest, Instagram, NASA, The Washington Post
CakePHP	PHP	MVC, podpora AJAX, HTML formularu, JavaScript View Helpers, komponenty pro Cookie, Session, Email a Request Handling, CRUD	těžký pro začátečníky, starý framework	BMW, MIT, HYUNDAI, Followmy Tv
Flask	Python	flexibilita, jednoduchý vývoj, škálovatelnost, jednoduchý syntax	nutnost přidávat speciální rozšíření, omezenost, pro menší projekty	Microblog, Red Hat, Rackspace, Reddit
Ruby on rails	Ruby	Ruby MVC, jednoduché testování, rychlý vývoj, ochrana osobních údajů	nejasná dokumentace, těžký pro začátečníky, problémy při vývoje na OS windows	GitHub, Shopify, Airbnb, Twitter, Kickstarter
Laravel	PHP	MVC, test-friendly, doplňkové nástroje	problémová kompatibilita	Twitch, Disney

Tabulka 3.1: Analyza backendů

Framework	Programovací jazyk	Výhody	Nevýhody	Známá využití
React	JavaScript	jednoduchost, popularita, stabilita, znovupoužitelnost, flexibilita, virtuální DOM	složité koncepty JSX syntaxe	Facebook, Instagram, Netflix, Uber, Reddit
Vue.js	JavaScript	virtuální DOM, podrobná dokumentace, jednoduchá integrace	malé komunity, jazyková bariéra, slabá stabilita komponentů	Nintendo, Gitlab, Behance, Xiaomi, Adobe
Angular	TypeScript	MVC, two-way data binding, velká komunita	těžké a pomalé aplikace, složitý	BMW, Xbox, Forbes, Blender, LEGO
jQuery	JavaScript	jednoduchost, kompatibilnost, DOM	zastaralé DOM API, je pomalejší	Kickstarter, Pandora, Twitter, Microsoft

Tabulka 3.2: Analyza frontendů

Kapitola 4

Analýza

Tato kapitola se věnuje analýze stávajícího systému. V práci je rozebrána nejdříve původní práce, slabší části projektu, které, lze zlepšit a proč tomu tak je. Dále se ověří splnění požadavků a provede analýza kódu. Na závěr je zde shrnutí, jaké části je potřeba upravit systematicky a jaké jen v malém měřítku.

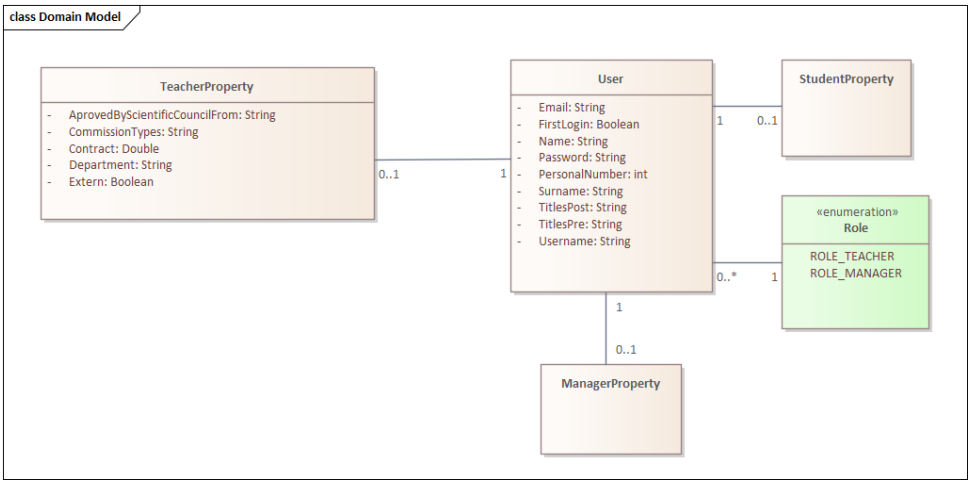
4.1 Analýza původní práce

4.1.1 Možné změny datového modelu původní práce

V původní aplikaci, která byla realizována studentem Gritsaiem, byla odhalena slabina v class diagramu a pak následně i v celém systému – použití různých řešení najednou (obrázek 4.1). V systému jsou použity celkem tři způsoby rozlišení rolí – manager, učitel, student je uděláno pomocí tzv. flagů (boolean hodnot), enumů s rolí a property (zvláštní entity pro každou roli). Na rolích závisí dostupné funkcionality a UI.

Autor práce chce, zaprvé, sjednotit systém, aby bylo použito jenom jedno řešení, a, zadruhé, vybrat to nejlepší z možných variant.

Použití flagů v tomto případě není antipatternem, ale není správné z důvodu existence obtíže sdílení doplňujících informací flag použít. Uveďme příklad, který jednoduše vysvětlí názor autora. Je-li vyučující externím specialistou, tak má ve svých vlastnostech atribut Extern = True. Avšak, tímto způsobem nejsme schopni pridat doplňující údaje, které chceme vědět.



Obrázek 4.1: Datový model uživatele

4.1.2 Analýza funkčních požadavků

Původní práce nastavila sadu funkčních a nefunkčních požadavků. Zde jsou body pro vylepšení jednotlivých funkčních požadavků. ***Poznámka: Rozdělení na MUST HAVE a NICE TO HAVE je z původní práce.***

MUST HAVE

FR1. Systém bude umožňovat uživateli vytvářet nové skupiny lidí.

Ano, aplikace umožňuje vytvářet skupiny – komise pro státnice.

FR2. Systém bude umožňovat uživateli editovat již existující komise.

Ano, editovat komise může jen administrátor.

FR3. Systém bude umožňovat uživateli schvalovat již existující komise.

Ano, může to dělat učitel, který do dané skupiny patří.

FR4. Systém bude umožňovat uživateli měnit stav existující komise.

Ano, může to udělat administrátor a komise má několik možných stavů (APPROVED, EDITABLE, DRAFT).

FR5. Systém bude umožňovat uživateli přidávat/měnit pracoviště v rámci existující komise.

Ano, všechno se nachází v menu editací.

FR6. Systém bude umožňovat uživateli nastavovat úvazek každému členovi komise.

Ne, bohužel tento požadavek není naimplementován.

FR7. Systém bude umožňovat uživateli výběr z vícero kombinací komisí.

Ano, aplikace sama může vytvářet náhodné komise, ale když generuje komisi se třemi členy, proces se dramaticky zpomalí (viz Rychlost).

FR8. Systém bude umožňovat uživateli volit obor/zaměření, pro nějž je komise vhodná.

Ano, je to v systému.

FR9. Systém bude schopen automaticky posílat e-maily různých typů.

Ne, po ověření se zjistilo, že systém neodesílá emaily. Toto je potřeba už během refactoringu a implementace debugovat.

FR10. Systém bude umožňovat členům komisí reagovat na výzvu.

Ano, systém to umožňuje.

FR11. Systém bude umožňovat členům komisí doporučit dalšího člena komise.

Ne, při vytváření skupiny nemůže učitel doporučit dalšího člena.

NICE TO HAVE

FR12. Systém umožňuje učitelům preferenci oboru, jehož zkoušejícím chce v rámci komise být.

Ano, to může udělat učitel v osobním nastavení.

FR13. Systém bude schopen exportovat data do Excelu v CSV podobě.

Ano a ne. Aplikace vytváří XSLX misto CSV (i když původně se mluvilo jen o CSV). Aplikace ten XSLX vytváří jen s prvním sloupcem tabulky a přidává info jen o některých uživateli.

FR14. Systém bude umožňovat uživateli zprostředkovat přístup do aplikace dalším uživatelům.

Ne, ale tento bod nemá ani praktické využití.

FR15. Systém bude umožňovat uživateli vytvářet a editovat e-mailové šablony.

Ano, administrátor to může udělat, ale pokud nebude fungovat email (viz FP9), nelze to otestovat

4.1.3 Analýza nefunkčních požadavků

NFR1. Uživatelsky přívětivé a jednoduché UI.

UI je funkční a pochopitelný pro nové uživatele (viz Testování)

NFR2. Rychlost.

Aplikaci trvá 3-5 minut vytvoření jedné skupiny se třemi členy. Je třeba prozkoumat, proč tomu tak je.

NFR3. Systém bude pracovat s reálnými daty z již existující databáze

Prozatím má aplikace svoji databázi. Je potřeba udělat zálohování dat při vytvoření komise, i když se budeme opírat o KosAPI.

NFR4. Funkčnost ve všech moderních prohlížečích.

Aplikace je funkční v Google Chrome a Firefox.

NFR5. Responsible design.

Ne, layout se rozpadá při šířce 768px, což je velikost průměrného tabletu.

NFR6. Lokalizace (CZ, EN) jako nice to have.

Ne, není.

4.2 Analýza kódu

Kód je správně rozdělen do různých tříd podle jejich významu. Java model je oddělen od controlleru, i když některé třídy používají proměnné přímo ve funkcích. (obrázek 4.2)

Kód obsahuje 8 komentářů s TODO popisem:

```
public void sendSimpleMessage(String[] emails, String subject, String text) {

    var message = new SimpleMailMessage();

    message.setFrom("noreply@baeldung.com");
    message.setTo(emails);
    message.setSubject(subject);
    message.setText(text);

    emailSender.send(message);
}
```

Obrázek 4.2: Příklad kódu s proměnnou přímo v kódu

1. ExamRepository.java, TODO pro testování funkce getAllByDate Samotná aplikace má jenom 2 JUnit testy, bude potřeba přidat další testy, např. na automatické vytvoření komisí.

2. CommisionSearchBox.js, TODO if field != all TODO change correct degree Tento TODO je realizovaný.

3. AutoGenerating.js, TODO pro vypnutí tlačítka po stisknutí

4. CommisionList.js, TODO pro obnovení stavu po změně komise

5. AppConfig.js, přidat globální proměnnou, která definuje prostředí (lokální nebo nasazení) Aplikace nemá konfigurační profily. Většina nastavení je přes komentování řádků kódu nebo inline proměnné. O profilech se jedná v kapitole 7.3

6. Exam.java, změnit jeden z atributů na jiný V modelu pro popis zkoušky místo degree (což je titul) se má používat FieldOfStudy, i když zkouška už má atribut fieldOfStudy, který je ve formě Stringu.

7. LocationRepository.java, 'query method' TODO se týká použití query pro získávání volných mistností pro komise. To se řeší na servisní úrovni.

8. UserService.java, "figure out with requirement: data.xl -> 2.sheet -> columns F and G" Komentář se týká data.xslx souboru, ze kterého se dotahuje info o učitelích, sloupec F obsahuje data, sloupec G obsahuje typ komise.

Kód také obsahuje spoustu kusů, které aplikace nevyužívá.(obrázek 4.3) Velká část kódu jsou pokusné verze různých funkcí, pozůstatky z debuggingu a staré atributy.

```
    //DO NOT USE THEM
//    @PostMapping
//    public Exam create(@RequestBody ExamTO examTO) {
//        return examService.save(examTO);
//    }

//    @PostMapping("/{id}")
//    public Exam update(@PathVariable Long id, @RequestBody ExamTO to) {
//        return examService.update(id, to);
//    }
```

Obrázek 4.3: Nepoužitý kód

4.3 Analýza změn a rozšíření

Původní práce pana Gritsaie obsahovala část "Budoucí práce", která popisovala další možný rozvoj a možné zlepšení.

4.3.1 Studenti v systému

Mezi změnami systému, nad kterými se přemýšlelo, patří přidání studentů do aplikací, aby mohli vidět kompletní informace o své závěrečné zkoušce včetně času, místností a údaje o předsedovi a ostatních členech komise, což není možné získat bez KosAPI. Ale nápad nebyl žádným způsobem projednán a zůstal jenom jako další dobrý způsob rozšíření systému. Existuje několik způsobů realizace:

1. Poslat e-maily studentům pres systém, když je daný student přidán do komise
2. Přístup do systému přes SSO API a oddělený UI pro studenty.

4.3.2 KosAPI a SSO API

V rámci svých změn a zlepšení autor práce chce rozšířit systém pro správu státnicových okruhů o dvě API – KOS API a SSO API.

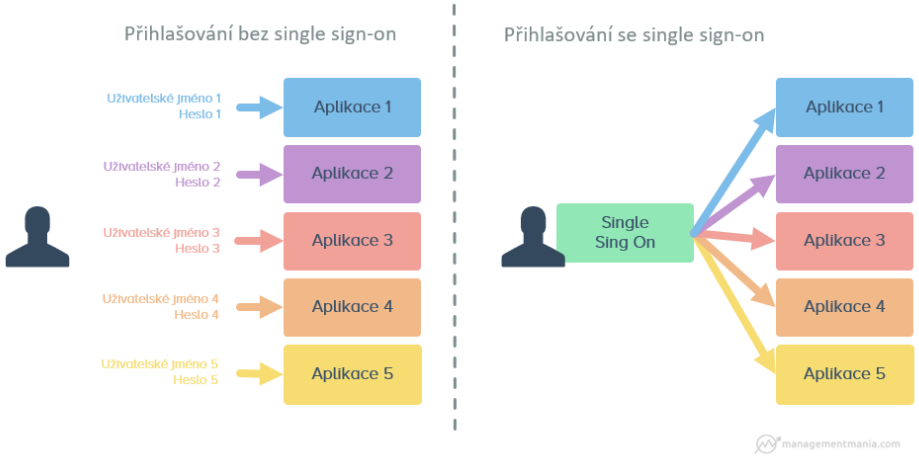
Technický provoz KOS API zajišťuje ICT oddělení Fakulty informačních technologií [Jir15]. Rozhraní pracuje s celouniverzitní databází – je speciálně navržen pro práci s těmito daty. API dává přístup k informacím o studentech, profesorech a cvičících.

SSO API [oIC19] dovoluje uživatelům dloudobě (až po dobu 10h) přihlášení do všech informačních systémů ČVUT. Na rozdíl od lokálního přihlášení (v

našem případě - pres JWT) se stačí přihlásit jednou do jakékoli ze služeb ČVUT a přihlášení bude udržováné do 10 hodin (obrázek 4.4).

Použití různých školních API umožňuje vyhnout se duplicitám dat a kódu, je příjemný pro uživatele a jednotný pro všechny systémy ČVUT.

Bohužel i přes snahu dostat do API, žádný přístup se nepodařilo získat. To měla být hlavní úprava aplikace, proto celá práce je zaměřena na zlepšení kódu a akceptaci zákazníka.



Obrázek 4.4: SSO vs Local [sso22]

Kapitola 5

Návrh

V této kapitole projdeme návrh naší aplikace. Samotný návrh byl vytvořen studentem Gritsaem v rámci jeho bakalářské práce. V kapitole 4 se řešilo několik zlepšení, ale API zůstane víceméně stejné (až na přidání nového rozhraní pro obory).

5.1 Entity

Díky existujícímu modelu, nebylo potřeba navrhovat nové entity a jejich propojení. Datový model a jeho části byly mírně zrefaktorované a upravené (obrázek 5.1). Největší změnou je úprava modelu uživatele, které je popsána v kapitole 7.1.

5.2 Mailing

V předchozí práci nebylo popsané, jak funguje mailing. Mailing se používá při posílání emailů členům komise. Pan Gritsai použil JavaMailSender a napojil ho přes SMTP Google účet (obrázek 5.2).

SMTP server pro mailer dostává požadavek s backendu a pak posílá zprávu podle zadaných údajů. Bohužel tento bod neměl otestovaný, a kvůli tomu samotný mailing nefungoval. Bylo potřeba nastavit Google App password.

5.3 Role v systému

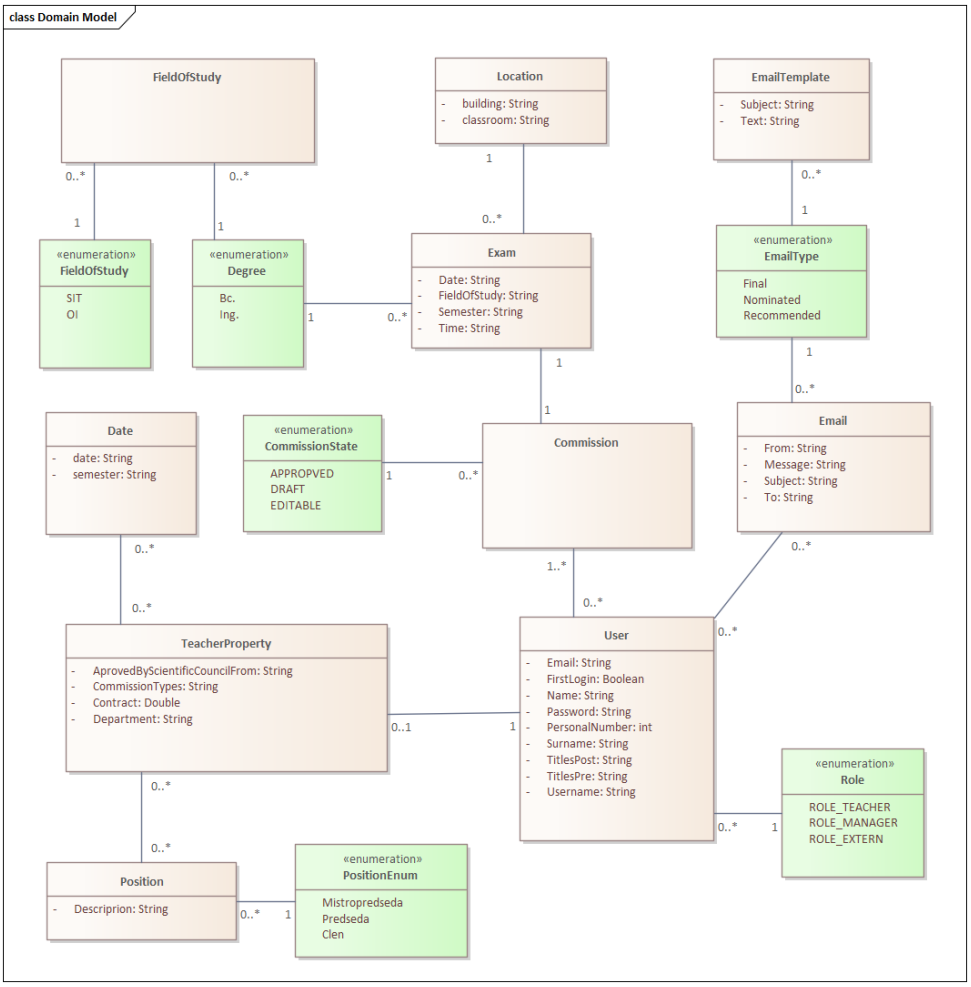
Rozdělení funkcionalit se také obnovilo. Učitel a manažer jako role v systému zůstaly, ale proběhlo pár změn:

1. Manažer dostal novou funkcionalitu (Add/Edit specialisation), která dovoluje měnit specializace bez úprav DB.
2. Učitel má Logout, který předtím v diagramu chyběl. V samotné aplikaci taková možnost byla.
3. Vznikla nové role - Unauthorized, a má dva use case - Login via SSO a Login via local account.

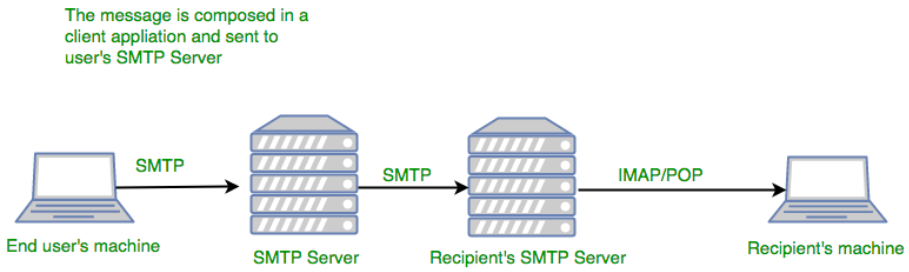
5.4 Architektura

Jedna z variant, jak takový systém lze postavit, je rozdělit části aplikace do mikroservis.[Tho15] Jedná se o rozdělení aplikace podle contextu. To je jeden ze způsobů už existující systém zrefaktorovat a vytvořit znovupoužitelné částí.[KH18]

V našem případě systém, i když má velké API - se nedá rozdělit. Spíš by šlo o integraci jiných služeb (už existujících) do naší aplikace (kapitola 2). Vytvoření nové služby, co by řešila např. profesory a cvičící, byla by zbytečná a redundantní, a proto systém zůstane monolitní.



Obrázek 5.1: Datový diagram



Obrázek 5.2: Posílání emailů přes SMTP server [Agg21]

Kapitola 6

Testování

V této kapitole je fáze testování, které proběhly s paní sekretářkou Kateřinou Maršálkovou, která poskytovala svůj feedback na nasazenou aplikaci.

6.1 První fáze testování

Testování probíhalo formou volného použití aplikace. Jedinym ukolem bylo vytvorit komisi a exportovat ji do XLSX. Po průchodu aplikací paní sekretářka poskytla tento feedback:

1. Názvy studijních programů jsou poměrně matoucí (a zřejmě jsou mezi nimi i programy, které nepatří pod naší katedru, ty je pro nás zcela zbytečné)
2. PhD. komise se vytváří úplně jiným způsobem (dává je dohromady pan prof. Železný)
3. U výběru vyučujících do komise je nezbytné vidět i jejich tituly
4. V návaznosti na to by bylo praktické moci vyučující rovnou dosazovat do jejich rolí v komisi - předseda, místopředseda, člen..
5. Ne všichni vyučující zkouší všechny obory. Někteří vyučující také zkouší pouze obecné okruhy, někteří pouze odborné
6. Jak do skládání komise zapracujete externisty, tajemníky a zástupce z katedry matematiky (ty nám pak na konkrétní data posílá katedra sama)
7. Bylo by také praktické, kdyby aplikace nějak zohlednila (třeba formou nějakého znaku či barvy), že byl dotyčný člen už v komisi v minulém semestru.
8. Oceňuji možnost poslat členům komise přímo skrze aplikaci email, to je velmi praktické

6.2 Druhá fáze testování

Druhá fáze taky proběhla formou QnA a byla potřebná pro zajištění informací z prvního kola testování. Následují jsou otázky a odpovědi ohledně aplikace.

Otázka. Můžete jasněji popsat, v čem byl ve skutečnosti problém v orientaci teď? Nemusí to být přesná odpověď, ale aplikace by neměla vám dělat problém

Chvíli mi dělalo problém najít, která záložka je na vytváření komise, nejprve jsem měla pocit, že to je úvodní stránka, ale tam je seznam komisí. Možná je to zvoleným layoutem, protože názvy záložek mi dávají smysl. Spíše to všechno trochu splývá a tím, že normálně pracuji u komisí s excelem, je to pro mě změna.

Otázka. Můžete určit, jaké programy jsou zbytečné?

Seznam programů a oborů pod naší katedrou:

Bc. - SIT, OI - Software, Mgr. - OI - Kybernetická bezpečnost, OI - Bioinformatika, OI - Umělá inteligence, OI - Softwarové inženýrství, OI - Datové vědy.

Otázka. Nevíte, chtěl by pan Železný vytvářet PhD komise přes naši aplikaci?

Zkuste se ho zeptat, má na to ale vlastní systém a nejsem si jistá, zda ho bude chtít měnit. Doktorandské státnice se navíc nyní spojují s Odbornou rozpravou.

Otázka. Můžu informace ohledně kdo a co zkouší, odněkud dostat?

Zeptám se, zda vám mohu poskytnout tuto tabulku. Je v nich bohužel poněkud zmatek - v tom by aplikace velmi pomohla, pokud by sama uměla najít, co kdo zkouší a člověk neproklikával x listů / řádků a hledal, zda vyučující daný obor vůbec zkouší.

Otázka. A co externisté, zástupci z katedry matematiky? Je nějaký seznam?

Externisty máme také v tabulkách na které se zeptám, obecně je s nimi ten problém, že jich máme k dispozici velmi málo. Zástupce z katedry matematiky navrhuje sama katedra matematiky ve chvíli kdy už máme stanovená data jednotlivých komisí. Jejich referent mi pak pošle seznam přidělených zástupců emailem na základě jejich časových možností.

Otázka. Potřeboval bych informace o tom, kdo zkoušel minulý semestr. Můžu tyto informace dostat?

V tabulce najdete složení komisí ze zimních SZZ. Vyučující, kteří se zúčastnili v posledních SZZ by tedy bylo dobré mít označeno nějakým značkem/barvou abych věděla, že je primárně nemám vybírat na následující SZZ termín.

Otázka. Budeme potřebovat ukládat nějaké doplňující info o tajemnicích či externistech? Například u externistů – z jaké firmy pocházejí?

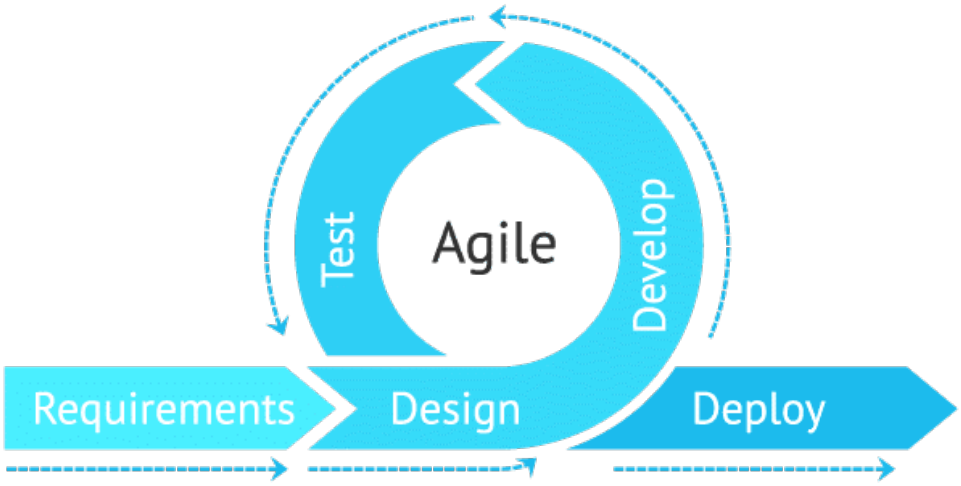
To je dobrá otázka - u tajemníků ne. U externistů a stejně tak i u všech vyučujících je potřeba uvádět pracoviště. (u externistů je to většinou třeba jiná univerzita, u našich vyučujících stačí číslo pracoviště např. u pana Šebka by to bylo 13142 - toto info naleznete v usermap po kliknutí na pracoviště.

Kapitola 7

Implementace

Je potřeba definovat, že se jedná o agilní vývoj. Agilní metodiky jsou skupiny metod původně určených pro vyvíjení softwaru založené na iterativním a inkrementálním vývoji. Umožňují rychlý vývoj softwaru a zároveň dokáží reagovat na změnu požadavků v průběhu vývojového cyklu. [ASRW17]

Vyvoj se střídá s testováním v agile (obrázek 7.1), a úpravy probíhá podle potřeb zákazníka. V tomto případě budou dvě fáze testování (nebo kontrolních otazek) a fáze vývoje.

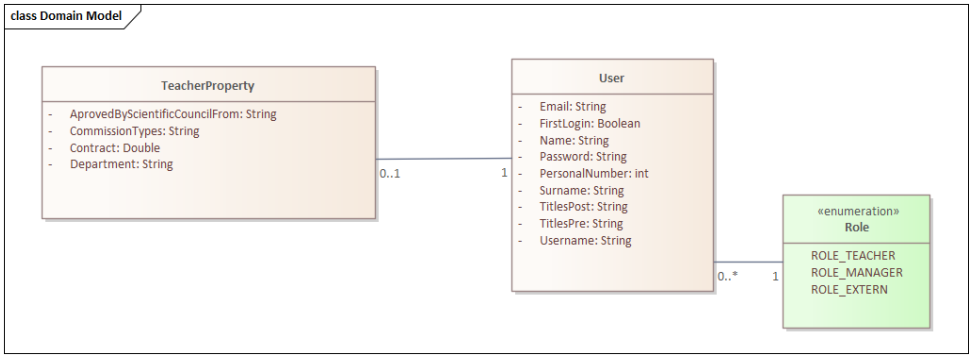


Obrázek 7.1: Agilní vývoj [Gof22]

7.1 Úprava datového modelu

Jak uz bylo zjištěno v podkapitole 4.1.1 - model, spojený s uživatelem či členem komise má v sobě několik způsobů popisu samotné entity. Bylo rozhodnuto místo flagů přidat další Enum s hodnotou Extern kvůli tomu, že externista

může mít taky doplňující data. Také se odstraní nepoužívané třídy, jako ManagerProperty a StudentProperty. TeacherProperty zůstane samostatně, protože stejně jako profesor či cvičící, i externista může být učitelem. (obrázek 7.2)



Obrázek 7.2: Nový model pro uživatele

7.2 Testovací nasazení na lokálním prostředí

Další důležitou částí vývoje je nasazení aplikace na lokální prostředí pro ověření provedených změn, udělaných během samotného vývoje. Ze začátku nebylo úplně jasné, proč aplikace po spuštění nedopada kvůli očekávání existující DB na localhost, předchozí autor nenechal žádný readme s postupem nasazení. Sepsání readme je součástí této části, bude popsán způsob nasazení a nastavení správného profilu (kapitola 7.3)

7.3 Vytvoření profilu pro lokální a produkční nasazení

Původně aplikace v kódu neměla žádné režimy pro nasazení do různých prostředí. Většina konfigurace byla popsána v kódu a přepínala se "zakomentováním" jednoho řádku "odkomentováním" druhého.

Aplikace měla konfiguraci i přes konfigurační soubor, ale po každé změně prostředí bylo potřeba přepsat hodnotu nebo použít komentování, jak to bylo už popsané. Proto byly v aplikaci zavedeny konfigurační soubory pro React.js a Spring Boot.

7.3.1 React.js

React používá .env soubory pro konfiguraci prostředí. Ted' to je použité pro určení adresy backendu.

V souboru .env.development je `REACT_APP_URL=http://localhost:8080`
V souboru .env.production je `REACT_APP_URL=http://fem.felk.cvut.cz:8080/fem`

7.3.2 Spring Boot

Spring Boot pro takové účely umožňuje použít .properties nebo .yml soubor. Byl preferován YML format kvůli lepší čitelnosti a skupinování proměnných. Konfigurace se rozdělila na 2 části - prod a dev.

V application.yml je nastavené, jaký má použít režim. (Kód 7.1)

```
spring:
    profiles:
        active: '@activatedProperties@'
```

Kód 7.1: YAML nastavení

A v pom.xml se řeší, který z profilu se používá (Kód 7.2)

```
<profiles>
        <profile>
            <id>dev</id>
            <properties>
                <activatedProperties>dev</activatedProperties>
            </properties>
            <activation>
                <activeByDefault>true</activeByDefault>
            </activation>
        </profile>
        <profile>
            <id>prod</id>
            <properties>
                <activatedProperties>prod</activatedProperties>
            </properties>
        </profile>
</profiles>
```

Kód 7.2: pom.xml nastavení

Pak v oddělených souborech application-dev.yml a application-prod.yml se řeší konfigurace pro jednotlivé prostředí.

7.4 CORS

CORS [Not16] nebo Cross-Origin Resource Sharing je technika využívající hlaviček protokolu HTTP, která umožňuje aplikacím běžícím ve webovém

prohlížeči přistoupit k jiné URL, který leží na jiné doméně. Její konfigurace je důležitá z pohledu backendu, protože jinak frontend nedokáže přestoupit k API. V samotné aplikaci to bylo naimplementované pomocí rozhraní s anotací. Pak každý kontroler implementoval rozhraní. Tento přístup má několik nevýhod:

1. Konfigurace je znovu měnitelná jenom v kódu, není měnitelná přes konfigurační soubor
2. Omezuje to možnost nastavit kontroler na různé endpointy – např tady na API je přístup jenom z localhost:3000

Existuje ale způsob, jak to nakonfigurovat jinak. Používat konfigurační třídy, které podporuje Spring Boot. To bylo realizováno pomocí dědění třídy WebMvcConfigurer. Tady se natahuje info přes konfiguraci na jakou URL je nastaven CORS. Taky je možné nastavit jiný mapping, aby bylo možné omezit část API nebo nastavit několik povolených CORS URL pro mikroservisy.

```
@Configuration
@EnableWebMvc
public class WebConfig implements WebMvcConfigurer \{

    @Value("\$\{fem.app.cors.origins\}")
    private String origins;

    @Override
    public void addCorsMappings(CorsRegistry registry) \{
        registry.addMapping("/**").allowedOrigins(origins);
    \}
\}
```

Kód 7.3: Nastavení CORS

7.5 Nastavení správných oborů pro zkoušení

Během testování se zjistilo, že přes systém se nebudou generovat státnicové komise pro doktorské studium. Stejně taky byly přesně popsané obory, které se budou zkoušet. Byly provedeny změny v enum a DB.

Kapitola 8

Nasazení

V této kapitole rozebereme jak bude probíhat nasazení a jaké změny se provedly.

8.1 Prostředí

Aplikace je nasazená a běží na serveru ČVUT (felk.cvut.cz). Na serveru je již nainstalovaný JDK 11, Tomcat server a PostgreSQL.

8.2 Postup nasazení

Z velké části zůstane postup, popsaný studentem Gritsaiem, stejný, ale pár změn se provede:

1. Na serveru musí být nainstalován Tomcat 9, Java 11 a PostgreSQL 11. Pro instalaci je potřeba vytvořit postgres uživatele a tomuto uživateli udělit práva na vytváření databází. Poté vytvořit databáze s názvem, který pak nastavíte v konfiguraci.
2. V souboru application.prod.yml nastavit jméno postgres uživatele s právy. Nastavit URL na frontend pro CORS a URL na databázi se správným názvem.
3. V souboru .env.production nastavit URL na backend.
4. Použít script v rootu projektu pro build a přesun souboru .war pro Tomcat.
5. Na serveru přesunout nový .war soubor z ./download do ./app.

Kapitola 9

Závěr

9.1 Shrnutí

Cílem této bakalářské práce bylo vykonání podrobného rozboru existujícího projektu spolu s navrhováním a realizací jeho zlepšení.

Byla provedena analýza existujícího Systému pro správu státnicových okruhů realizované studentem Gritsaiem v minulém roce a uskutečněný doplňkové rešerše. Jejich výsledky ukázaly, že projekt je v dobrém stavu. Avšak provedené změny ho rozšířily, proběhly 2 kola testování a implementace a přivedly k pozitivnímu celkovému výsledku.

Přestože realizace propojení na KosAPI a SSO API nebyla dokončena kvůli tomu, že nebyl získán přístup, o který bylo žádáno odpovědné osoby, systém bude použit již v následujícím semestru pro sestavení komise na SZZ.

9.2 Perspektivy dalšího rozšíření

Aplikace je připravena na využití a splnění svých cílů, jakými jsou jednoduché a pohodlné sestavení komisí na SZZ. Ale existují další možnosti její zdokonalení.

Jedním z nich je přidání studentů do systému, aby měli možnost vidět plnou informací o svých závěrečných zkouškách – členech komise, místnosti, času a data.

Příloha A

Literatura

[Agg21] Anshul Aggarwal. Sending email through java with ssl / tls authentication [online]. July 2021. URL: https://www.geeksforgeeks.org/sending-email-java-ssltls-authentication/.

[ASRW17] Pekka Abrahamsson, Outi Salo, Jussi Ronkainen, and Juhani Warsta. Agile software development methods: Review and analysis, 2017. URL: https://arxiv.org/abs/1709.08439, doi:10.48550/ARXIV.1709.08439.

[Ber17] M. Bertoli. *React Design Patterns and Best Practices*. Packt Publishing, 2017. URL: https://books.google.cz/books?id=z08oDwAAQBAJ.

[dt22] InSIS developer team. Insis - dokumentace a užívatelská příručka [online]. 2022. URL: https://insis.vse.cz/dok/help.pl?tab=2.

[GD21] Ing. Šebek Jiří Gritsai Dmitrii. Application for management final exam topics [online]. 2021. URL: https://dspace.cvut.cz/handle/10467/94895.

[GKS16] S. G. Ganesh, Hari Kiran, and Tushar Sharma. *Building Database Applications with JDBC*, pages 359–387. Apress, Berkeley, CA, 2016. doi:10.1007/978-1-4842-1836-5_12.

[Gof22] Addy Goff. What is agile methodology in project management? [online]. May 2022. URL: https://hive.com/blog/what-is-agile-project-management-methodology/.

[Hin18] J. Hinkula. *Hands-On Full Stack Development with Spring Boot 2.0 and React: Build modern and scalable full stack applications using the Java-based Spring Framework 5.0 and React*. Packt Publishing, 2018. URL: https://books.google.cz/books?id=vqdhDwAAQBAJ.

[IJM14] Ph.D. MBA Ing. Jaromír Marušinec. Nasazení is v vŠ [online]. October 2014. URL: https://www.eunis.cz/cz/informacni-systemy.

[Jav19] Arshad Javeed. Performance optimization techniques for reactjs. In *2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, pages 1–5, 2019. doi:10.1109/ICECCT.2019.8869134.

[Jir15] Jakub Jirůtka. Projekt kosapi - wiki [online]. 2015. URL: https://kosapi.fit.cvut.cz/projects/kosapi/wiki.

[KH18] Holger Knoche and Wilhelm Hasselbring. Using microservices for legacy software modernization. *IEEE Software*, 35(3):44–49, 2018. doi:10.1109/MS.2018.2141035.

[KSN18] M. Keith, M. Schincariol, and M. Nardone. *Pro JPA 2 in Java EE 8: An In-Depth Guide to Java Persistence APIs*. Apress, 2018. URL: https://books.google.cz/books?id=Mh5KDwAAQBAJ.

[Not16] W3C Working Group Note. Cors for developers [online]. October 2016. URL: https://www.pmi.org/learning/library/seven-causes-project-failure-initiate-recovery-7195.

[oIC19] CIC Department of Integration and Coordination. Single sign on service on ctu [online]. 2019. URL: https://ist.cvut.cz/en/our-services/single-sign-on/.

[RF07] Discenza R. and J. B. Forman. Seven causes of project failure: how to recognize them and how to initiate project recovery [online]. 2007. URL: https://www.pmi.org/learning/library/seven-causes-project-failure-initiate-recovery-7195.

[s.r21] JRebel s.r.o. Java developer productivity report [online]. 2021. URL: https://www.jrebel.com/resources/java-developer-productivity-report-2021.

[s.r22] IS4U s.r.o. Is4u - is pro vŠ [online]. 2013-2022. URL: https://www.is4u.cz/cs/produkty.

[sso22] Single sign on [online]. 2022. URL: https://managementmania.com/cs/single-sign-on.

[tea22] Project management statistics: Trends and common mistakes in 2022 [online]. 2022. URL: https://teamstage.io/project-management-statistics.

[Tho15] Johannes Thones. Microservices. *IEEE Software*, 32(1):116–116, 2015. doi:10.1109/MS.2015.11.