FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Informační systém pro správu studijních projektů
Student:	Sergey Dunaevskiy
Vedoucí:	Ing. Marek Suchánek
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2019/20

Pokyny pro vypracování

Na základě analýzy a zkušeností navrhněte a implementujte informační systém, který bude podporovat aktivity kolem studijních projektů (například složitější úkoly, týmové projekty, semestrální práce) během celého jejich životního cyklu od specifikace hodnocení a témat po jejich dokončení a archivaci.
- S využitím konceptuálního modelování zanalyzujte oblast studijních projektů na FIT ČVUT v obecné rovině (tj. bez omezení se na konkrétní předmět) pro stanovení požadavků na systém.
- Proveďte stručnou rešerši dostupných systémů na FIT ČVUT pro správu projektů.
- Navrhněte snadno rozšiřitelný informační systém jako webovou aplikaci umožňující vyučujícím a studentům snadnou organizaci, sledování a hodnocení studijních projektů. Při návrhu zvažte integraci s existujícími službami na ČVUT.
- Implementujte systém dle návrhu, řádně jej otestujte a zdokumentujte. Výběr zvolených technologií pro implementaci zdůvodněte.
- Zhodnoťte výsledný systém a navrhněte další možný rozvoj systému.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 25. listopadu 2018

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Informační systém pro správu studijních projektů

Sergey Dunaevskiy

Katedra softwarového inženýrství

Vedoucí práce: Ing. Marek Suchánek

17. června 2019

Poděkování

Děkuji Ing. Marku Suchánkovi za cenné rady a pomoc při vytváření této bakalářské práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 17. června 2019 .....

České vysoké učení technické v Praze

Fakulta informačních technologií

Odkaz na tuto práci

DUNAEVSKIY, Sergey. *Informační systém pro správu studijních projektů.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2019. Dostupný také z WWW: ⟨`https://github.com/dunaevskiy/thesis-bachelor`⟩.

Abstrakt

Tato bakalářská práce se zabývá tvorbou nového informačního systému pro správu a hodnocení studentských projektů. Návrh systému vychází z analýzy již existujích obdobných služeb na Fakultě informačních technologií, České vysoké učení technické v Praze. V rámci práce je vytvořena specifikace systému, zdůvodněn výběr nástrojů a popsáno jejich konkrétní využití. Implementačí část obsahuje architekturu a způsoby realizace serverové a klientské části služby.

Klíčová slova informační systém, studentské projekty, open-source, javascript, typescript

Abstract

This bachelor's thesis deals with the creation of a new information system for the management and evaluation of student projects. The system design is based on an analysis of similar services at Faculty of Information Technology, Czech Technical University in Prague. Within the thesis a system specification was created, the choice of tools was justified and their usage was described. An implementational section contains a system architecture and forms of realisation of server and client part of the service.

Keywords information system, student projects, open-source, javascript, typescript

Obsah

Seznam zdrojových kódů

Seznam obrázků

Úvod

Vysokoškolské prostředí je jedna z mnoha oblastí, kde se v současné době velice často vyžaduje pokročilá správa strukturovaných celků v podobě jednorázových semestrálních úkolů, semestrálních prací, soukromých a týmových projektů, bakalářských a diplomových prací.

Ve snaze naplnit danou potřebu jednotlivé výzkumné a akademické instituce poskytují svým studentům a zaměstnancům centrální uzavřené systémy orientované spíš na menší celky (předměty, katedry), protože velice často vyžadují specifické typy obsahů. Ve finanční oblasti jsou potřeba například interaktivní grafy časového průběhu, v informačních technologiích ukázky zdrojového kódu a popisy systémů s pomocí jazyků konceptuálního modelování. Dané služby mohou představovat vlastní implementace systémů správy nebo komerční software s uzavřenými zdrojovými kódy. Z hlediska OSS (open-source software) řešení neexistuje žádná služba, jež by se plně orientovala na vedení a správu studentských projektů.

Vzhledem k dané situaci se dá předpokládat, že vznik moderního OSS nástroje pro správu takových celků, jenž by obsahoval jisté společné rysy, ale zároveň byl snadno přizpůsobitelný potřebám, by mohl být přínosem pro některé akademické instituce.

Cíl práce

Cílem této práce je analýza stávajících způsobů vedení studentských projektů zaměřená na Fakultu informačních technologií Českého vysokého učení technického v Praze (dále jen FIT ČVUT) s následným návrhem a implementací OSS informačního systému. Nová služba by měla být univerzálnější a víc přizpůsobena uživatelským požadavkům v oblasti řízení studentských prací.

Pro výchozí analýzu budou vybrány některé služby na fakultě, které dle subjektivního hodnocení mají největší podíl na správě projektů či nabývají užitečných funkcí. Na základě získaných poznatků bude navržena obecná specifikace nového informačního systému, která bude sloužit osnovou pro implementaci serverové a klientské části aplikace. Důraz bude kladen především na uživatelsky pohodlné rozhraní a dostatečné funkcionální možnosti. Pro dosažení daného cíle budou parciálně otestovány vybrané nástroje, technologie a implementační řešení v některých aspektech, aby se docílilo optimálního výběru.

Konečným výsledkem bude prototyp aplikace pro správu studentských prací. V případě úspěšné a dostatečné integrace s existujícími službami na FIT ČVUT bude zvažováno nabídnutí pro využití na dané fakultě.

Struktura práce

Kapitola 1 se věnuje obecné analýze portálů a služeb FIT ČVUT, které slouží pro správu projektů a plnění semestrálních úkolů. U jednotlivých služeb jsou popsány jejich silné a slabé stránky, které následně budou sloužit pro návrh nového IS (informační systém).

Kapitola 2 specifikuje obecné a funkční požadavky na nový IS a podrobně analyzuje vybrané primární procesy s pomocí konceptuálního modelování a jazyka UML (Unified Modeling Language).

Kapitola 3 definuje základní nástroje a technologie, s pomocí nichž se bude realizovat implementace IS, a povrchově popisuje návrh architektury.

Kapitola 4 popisuje tvorbu serverové RESTful[1] aplikace. Postupně jsou analyzovány, navrhovány a implementovány vybrané části aplikace. Pro tyto účely jsou v některých aspektech parciálně testovány webové frameworky, databáze a možná implementační řešení s následným výběrem nejvhodnějšího řešení.

Kapitola 5 je obdobná čtvrté kapitole, ale už se věnuje především tvorbě klientské webové aplikace. Kromě implementace aplikace kapitola zahrnuje psychologické vnímání uživatelského rozhraní a tvorbu designu.

Kapitola 6 zkoumá vliv legislativy na oblast webových aplikací, jaké nároky jsou kladeny v současné době a co je potřeba split v případě implementace IS pro akademické instituce.

Kapitola 7 využívá hotové funkcionality serverové a klientské části aplikace a popisuje způsob automatického a manuálního testování. Závěr kapitoly se věnuje možnému průběhu nasazení IS na produkční server.

Na závěr je zhodnocen výsledek a dosažení předem stanovených cílů a splnění zadání bakalářské práce.

V přílohách a na přiloženém médiu jsou uvedeny zdrojové kódy obou částí IS, krátké dokumentace pro vývojáře i pro uživatelé a jiné dodatečné materiály.

[1] REST (Representational State Transfer)

Kapitola 1

Analýza stávajících způsobů správy projektů

Kapitola se věnuje analýze vybraných stávajících portálů a aplikací pro správu projektů a plnění úkolů na FIT ČVUT. Analýza je provedena pouze z pohledu běžného studenta, jenž nemá přístup k funkcionalitě vyžadující speciální oprávnění. U každého portálu je ke konci vždy uveden stručný přehled pozitivních a negativních prvků, které by se mohly hodit pro návrh nového informačního systému.

1.1 SwinPro

Autor: Tým SwinPro
Analyzovaná verze: Dev-Release 1.1.4-M2 nebo FIT-1.1.5[2]
Datum analýzy: 15. 2. 2019
URL: `https://project.fit.cvut.cz/`

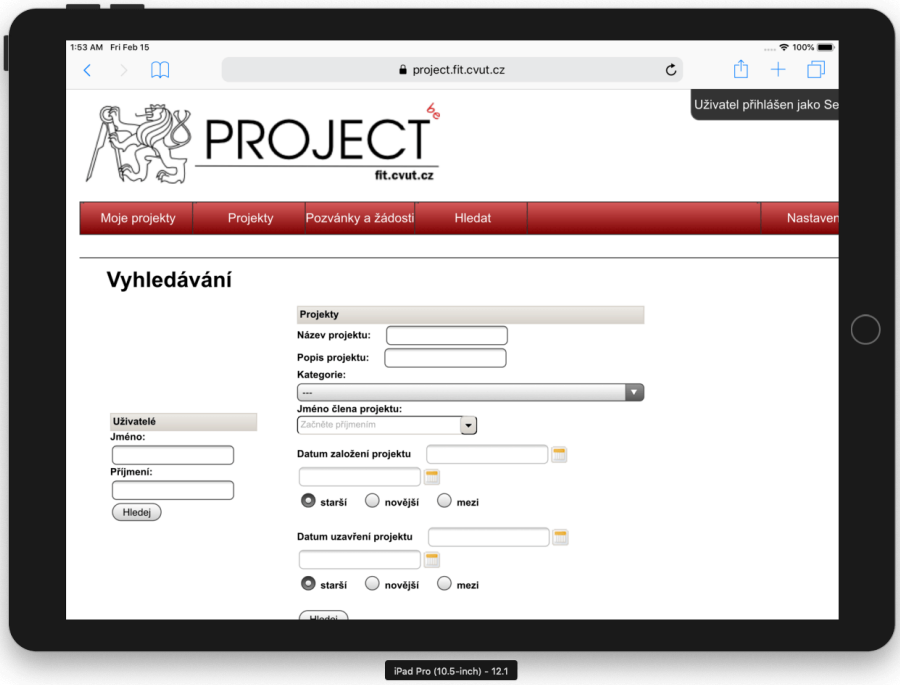
SwinPro je v současné době centrálním systémem pro správu projektů na FIT ČVUT pro některé předměty. Dle nabídky kategorií a zastoupení jednotlivých projektů v těchto kategoriích se dá usoudit, že je zaměřen především na obor softwarového inženýrství, konkrétně na předměty Softwarové inženýrství a Softwarový projekt. Zastoupení jiných předmětů není tak výrazné.

Uživatelé zde mohou založit projekt, jenž je rovnou spojen s konkrétním vyučujícím na fakultě, a zvolit jeho veřejnou dostupnost. Během vytváření je možné automaticky založit i podpůrné nástroje pro bug tracking, správu dodavatelského řetězce, wiki a průběžnou integraci. Dané nástroje nelze přidat později.

[2] Dle hlavní stránky [1] je nasazená verze Dev-Release 1.1.4-M2, dle vývojářské stránky [2] FIT-1.1.5.

Interní rozhraní projektu slouží především jako rozcestník pro různé nástroje a správu členů daného projektu. Studentům je umožněno měnit pouze základní informace, například název nebo popis. Veškerý vývoj projektu probíhá vně SwinPro v rámci podpůrných nástrojů nebo jiných, nesouvisejících aplikací. Neexistuje žádný uživatelsky pohodlný přehled stavu projektu, zda se shánějí členové nebo je projekt delší dobu neaktualizován.

Podle informací uvedených ve vývojářské sekci poslední plnohodnotný release byl označen verzí FIT-1.1.5 [2] a nasazen 10. 2. 2013 [3], od té doby následovala pouze řada oprav a modifikací. Poslední zaznamenaná aktualizace proběhla před téměř dvěma roky [4]. Je evidentní, že se portál pořádně neudržuje, spousta vnějších i vnitřních odkazů vede na neexistující stránky a obsahuje zastaralé informace a typografické chyby.



Obrázek 1.1: Ukázka vnitřní stránky portálu SwinPro

SwinPro není uživatelsky pohodlnou aplikací, jeho webové rozhraní (viz obrázek 1.1) není adaptováno pro současná zařízení. Při zobrazování stránek na menších obrazovkách se prvky rozhraní kriticky překrývají a není možné se dostat k překryté části. V dokumentaci nelze zobrazovat uvedené obrázky přímo v prohlížeči, jsou automatický staženy do zařízení.

Pozitivní prvky

- Každému projektu je umožněno automaticky přidávat individuální podpůrné nástroje.

Negativní prvky

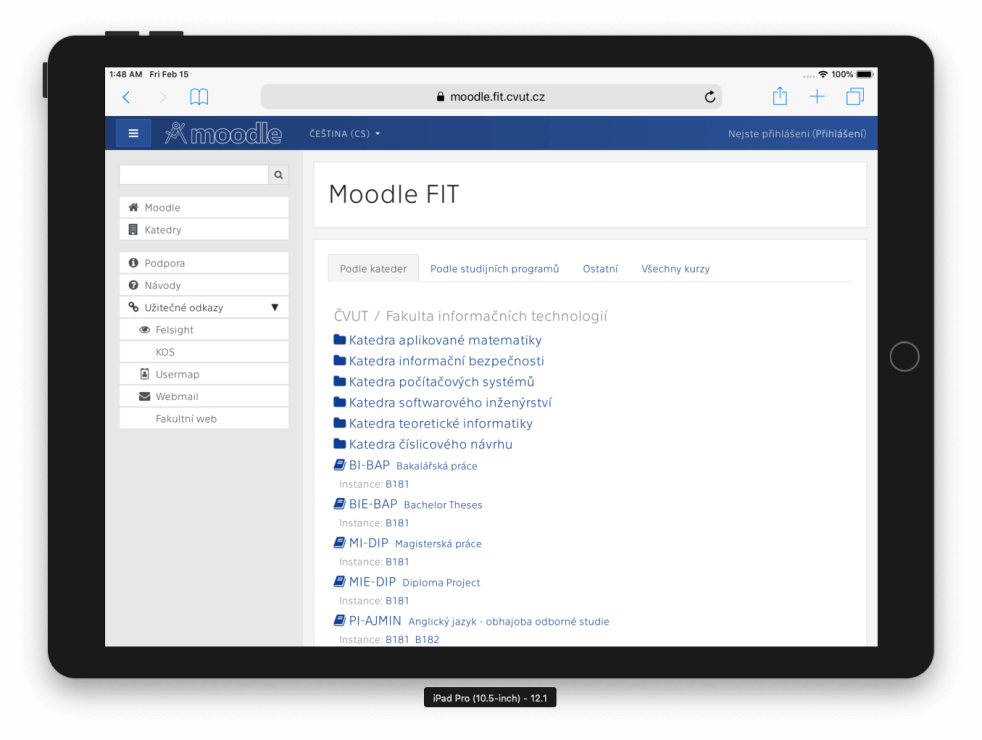
- Chaotické uživatelské rozhraní, které velice často neposkytuje přehlednou nabídku funkcionality.
- Neúplné a zastaralé informace napříč celým projektem vyvolávají pocit opuštěnosti portálu.
- Nelze zakládat studentské projekty, které by nebyly vedeny konkrétním vyučujícím.

1.2 Moodle FIT

Autor: Centrum znalostního managementu FEL ČVUT
Analyzovaná verze: Lassard 17
Datum analýzy: 15. 2. 2019
URL: `http://moodle.fit.cvut.cz/`

Moodle FIT [5] je webová aplikace založená na systému Moodle [6] a spravovaná Centrem znalostního managementu FEL ČVUT [7]. Primárně plní roli poskytovatele základních informací o průběhu a hodnocení jednotlivých předmětů v konkrétních semestrech. V roce 2018 nahradila předchozí systém s podobnou funkcionalitou.

Kromě správy předmětů Moodle FIT rovněž dává možnost vytvářet online testy a definovat úkoly zvlášť pro každý předmět. Úkol je zobrazován v přehledu daného předmětu a definuje zadání a nejzazší termín odevzdání. Po splnění úkolu, což znamená nahrání jednoho či více souborů, se čeká na hodnocení se strany vedoucího. Po celou dobu nesplněného úkolu se na boční straně stránek zobrazuje upozornění se zbývajícím časem pro odevzdání. Získaný počet bodů za úkol je hned zobrazen v příslušné sekci přehledu předmětu. Může být doprovázen komentářem od vedoucího.



Obrázek 1.2: Ukázka vnitřní stránky portálu Moodle FIT

Uživatelské rozhraní (viz obrázek 1.2) je adaptováno pro různé velikosti obrazovek, avšak realizovaná funkcionalita má často nepohodlné rozhraní. Vyhledávání na webu je velice omezené, není možné se dostat ke svým absolvovaným předmětům a tudíž i k odevzdaným úkolům.

Pozitivní prvky

- Zobrazování upozornění na nesplněné úkoly a zbývající čas.
- Komentáře vedoucího k odevzdanému úkolu.

Negativní prvky

- Veškerá komunikace ohledně úkolu se provádí pouze prostřednictvím posílání souborů s následnou známkou a komentářem.
- Chybějící funkcionalita pro náhled a vyhledávání starších úkolů, jež se nachází v archivované podobě.
- Mnohdy neintuitivní uživatelské rozhraní.

1.3 DBS portál

Autor: kolektiv autorů (bakalářské práce, diplomové práce)
Analyzovaná verze: 4.21.0
Datum analýzy: 14. 2. 2019
URL: `https://dbs.fit.cvut.cz/`

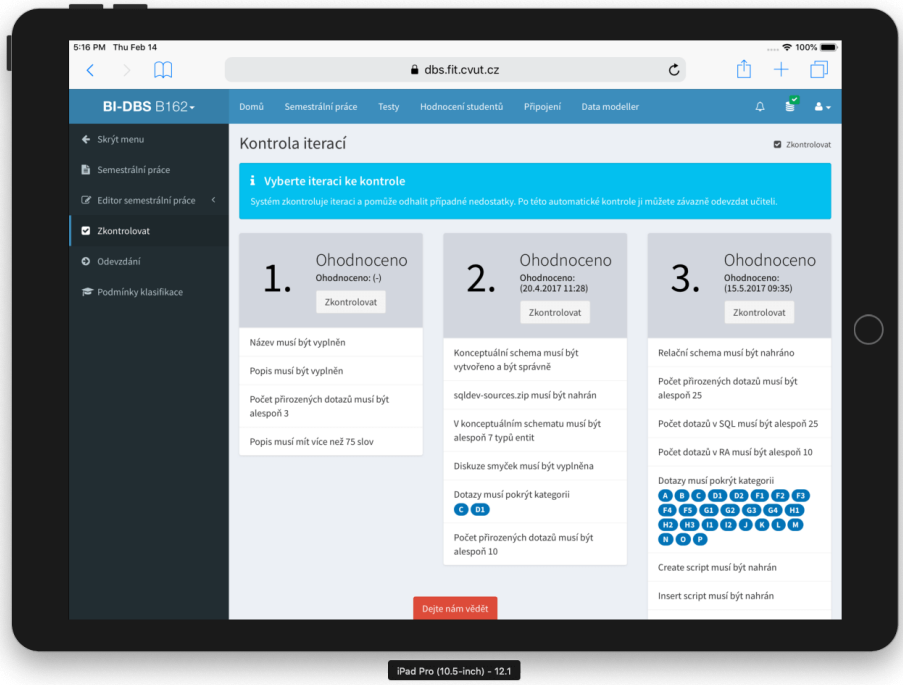
Portál předmětu databázových systémů vyvíjený od roku 2014 [8]. Ze studentského hlediska se jedná především o nástroj pro organizaci systematického odevzdávání průběžných částí semestrální práce, jež se píše během průchodu předmětem DBS (Databázové systémy). Uplatnění nachází rovněž během semestrálního testu a závěrečné zkoušky, po zbytek času je však pro běžného studenta téměř zbytečný.

Organizace odevzdávání představuje systém dělení práce na tři navzájem závislé iterace, které definují jednotlivá kritéria pro jejich splnění. Každá následující iterace rozšiřuje či upravuje předchozí požadavky, mezi které se řadí datum odevzdání a plnění úkolů. Studenti zpracovávají práci do předem definovaných forem, například popis do textového pole, obrázek konceptuálního modelu do pole pro nahrávání souborů a další obdobná uživatelská rozhraní.

Po splnění kritérií jednotlivých iterací studenti mohou odevzdat aktuální stav práce jako celek pro hodnocení vedoucím práce. Některé části se hodnotí automaticky, něco však vyžaduje manuální kontrolu. Vedoucí práce rozhoduje o konečným počtu bodů za iteraci, jenž se zobrazuje studentovi na stránce celkového přehledu.

Webové rozhraní (viz obrázek 1.3) je plně realizováno pro všechny běžné typy obrazovek a prohlížečů. Vychází z populární šablony ovládacího panelu AdminLTE [9].

Daný webový portál je využíván jako nástroj pro plnohodnotnou správu předmětu. Mimo správu semestrální práce nabývá možnosti data modelleru (modelování ERD – entity-relationship diagram), připojení ke školním či jiným databázím, přehledu anonymních výsledků studentů, vytváření průběžných semestrálních testů a závěrečné zkoušky s maximálně možnou automatizací kontroly výsledků. Veškerá tato funkcionalita není detailně analyzována z důvodu překročení rozsahu zájmu vytvářeného informačního systému.



Obrázek 1.3: Ukázka vnitřní stránky portálu DBS

Pozitivní prvky

- Existence stránky přehledu usnadňuje orientaci.
- Semestrální práce je dělena na iterace a následně na úkoly. Z hlediska studenta je to přehledné rozdělení práce.
- Některé úkoly je možné hodnotit automaticky.

Negativní prvky

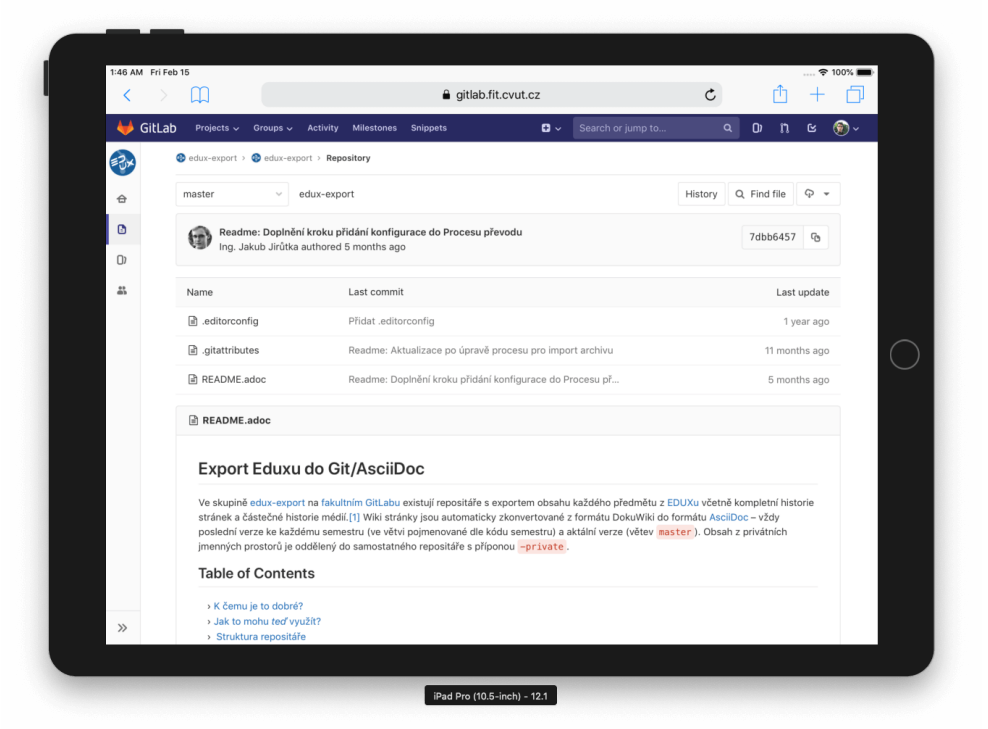
- Portál je zaměřen striktně na předmět databázových systémů. Není možné ho používat pro jakýkoliv jiný předmět bez kardinální změny implementace.
- Vedoucí práce nemá možnost okomentovat jednotlivé části práce. Realizovaná funkcionalita předpokládá pouze ohodnocení určitým počtem bodů.

1.4 GitLab

Autor: GitLab Inc.
Analyzovaná verze: Community Edition 11.3.14
Datum analýzy: 15. 2. 2019
URL: `https://gitlab.fit.cvut.cz/`

Fakultní nasazení aplikace GitLab poskytující v první řadě serverové úložiště pro Git repozitáře, systém sledování chyb, nastavení CI (continuous integration) a CD (continuous delivery) a případné vytváření příručky typu wiki. Využívá se pro neveřejné studentské a fakultní projekty spravované VCS (version control system) Git.

Na základě GitLab je vytvořena webová služba Course Pages [10], která, stejně jako Moodle FIT, spravuje základní informace o předmětech. Mimo materiálů k jednotlivým předmětů se zde publikují zadání k semestrálním úkolům, jež studenti následně zpracovávají a odevzdávají buď fyzicky, nebo s pomocí jedné z komunikačních platforem.



Obrázek 1.4: Ukázka vnitřní stránky GitLab

GitLab je služba, jež kolem sebe shromáždila obrovskou komunitu lidí [11]. Uživatelské rozhraní (viz obrázek 1.4) je udržováno se současnými standardy.

Pozitivní prvky

- Ohebný nástroj vyvíjený obrovskou komunitou mimo fakultu.

Negativní prvky

- Pro používaní je nutné umět VCS Git.

1.5 Komunikační platformy

Pojmem komunikační platformy se v daném případě označuje skupina prostředků pro komunikaci, jež přímo není určena k udržování projektů, ale využívá se pro tento účel. Mezi takové paří Roundcube[3], Slack, Discord a další aplikace podobné struktury.

Roundcube se bere jako primární způsob komunikace mezi studenty a fakultou. Úkoly a semestrální práce zadané na cvičeních se většinou odevzdávají přes e-mail a výsledky se oznamují na jednom z dalších cvičení, či přímo do aplikace pro správu klasifikace daného předmětu.

Pozitivní prvky

- Intuitivní způsob komunikace, většinou není nutno se vyznat v základních možnostech používané služby.

Negativní prvky

- V základní formě (bez rozšíření) pouze textová forma materiálů s případnými odkazy na vnější zdroje nebo soubory.
- Nepřehlednost předaných dat z hlediska verzování, komentování a hodnocení.
- Ztráta dat v dlouhodobém časovém úseku (automatické mazání zpráv, placená služba uchovávání většího množství dat apod.).

1.6 Fyzická komunikace

Klasická komunikace ověřená mnohaletými akademickými institucemi. V současné době se většinou obecně vyplácí pouze v případech, kdy výsledkem práce je jistý ma-

[3] Webmail

nuálně vytvořený objekt, jenž nelze převést do digitální podoby se zachováním jeho významu (například hardwarová součástka, fyzický model návrhu budovy).

Na FIT ČVUT se tohoto způsobu odevzdávání využívá zejména v případech, kdy digitální podoba klade omezení na opravování práce. Jedná se například o opravu chyb ve výpočetních matematických příkladech, uvádění dodatečných vizualizací k určitým částem práce apod.

Pozitivní prvky

- Kontrola odevzdané práce vedoucím projektu není závislá na jeho přístupu k internetu.

Negativní prvky

- Nutnost se fyzicky dostavit na určité místo pro předání materiálů.
- Je nutné disponovat příslušnými zařízeními pro tisk.
- Doba od odevzdání po obdržení ohodnocené práce může být nepřiměřeně velká, pokud se nepoužívá jiný komunikační prostředek.

Kapitola 2

Specifikace informačního systému

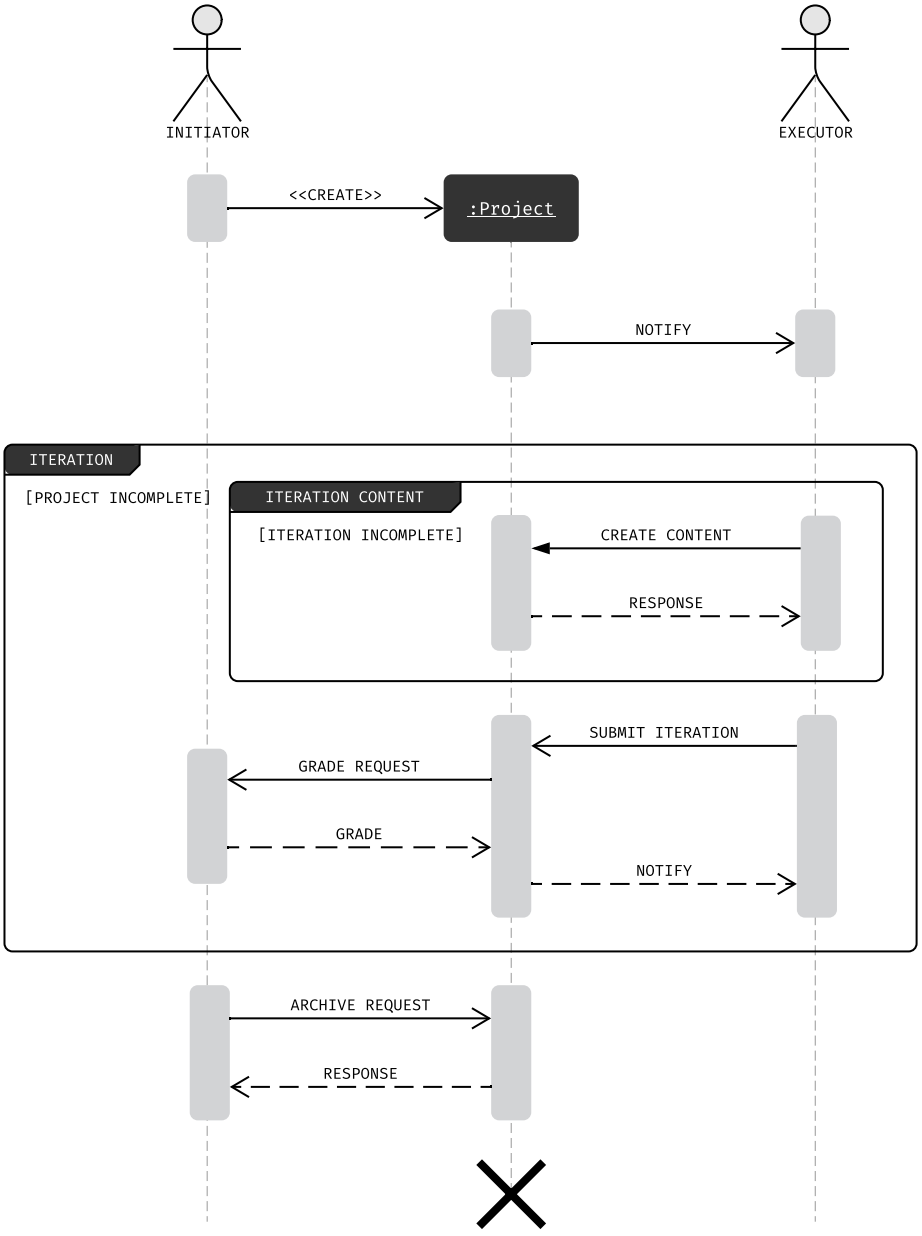
S využitím výše uvedené analýzy způsobů správy semestrálních projektů a seznamů pozitivních a negativních prvků je potřeba navrhnout informační systém, jenž by se mohl stát potenciální náhradou větší části služeb.

Pokud zobecníme správu jednoho projektu a vizualizujeme jeho životní cyklus (viz obrázek 2.1), to bude především komunikace dvou subjektů – vyučujícího a studenta[4] –, jež postupně tvoří obsah projektu.

Vyučující zakládá/přiřazuje/nabízí projekt studentům na fakultě. Průběh projektu je strukturovaný do jedné, či více iterací, přičemž obsahuje sadu úkolů, které je potřeba splnit. Úkoly zpravidla patří pod určité iterace, aby se jednoznačně definovala zadaná práce. Po vytvoření projektu je o tom student informován a začíná vypracovávat jednotlivé úkoly a tím tvořit obsah projektu. V situaci, kdy je současná iterace splněna, student ji může odevzdat ke kontrole vedoucímu projektu, jenž na základě hotové práce přiděluje určitý počet bodů. Daný postup tvoření a hodnocení obsahu se opakuje do doby, než jsou všechny iterace vyčerpány. V tuto chvíli se projekt považuje za dokončený a vedoucí ho archivuje.

Podobného způsobu správy se má držet i budoucí IS, nebude však omezován pouze na vedoucí z řad vyučujících. Projekty bude možné zakládat a řídit běžným studentům, tím bude do jisté míry podporována jejich samostatná činnost.

[4] V grafu na obrázku 2.1 jsou označené jako `initiator` a `executor`.



Obrázek 2.1: Sekvenční diagram zobecněného životního cyklu projektu na FIT ČVUT

2.1 Požadavky na systém

Pro definování rozsahu práce slouží funkční a obecné požadavky. Funkční požadavky zachycují jednotlivé funkcionality vytvářené aplikace, jejich logickou stránku a chování. Obecné požadavky, neboli nefunkční, jsou požadavky, jež na systém kladou jistá omezení a tím upřesňují rozsah práce, avšak nedefinují žádnou aktivní složku (například uživatelská funkcionalita nebo serverová činnost).

2.1.1 Funkční požadavky

FR00 Identita uživatelů Součástí aplikace je uživatelský systém vyjma autorizačního serveru. Aplikace využívá pouze autorizační servery třetích stran, s jejich pomocí probíhá registrace i přihlášení. Identitu uživatele musí být možné přesně určit na všech webových stránkách, kromě skupiny stránek sloužící pro autentizaci či informování o základních náležitostech (například podmínky užití).

FR01 Globální role Vsechny účastníci IS mají přidělenou jednu globální roli, která určuje jejich oprávnění v rámci aplikace. Ihned po registraci dostávají výchozí roli standardního uživatele. Existuje speciální role administrátor, jenž má neomezený přístup. Další role představují jiné množiny oprávnění, které jsou definovány v případech užití. U jednotlivých globálních rolí lze nastavit kritérium důvěryhodnosti. Uživatelé s danou rolí jsou v IS uvedeny se speciálním označením vedle uživatelského jména. Aplikace umožňuje měnit roli uživatele.

FR02 Osobní informace uživatelů Aplikace umožňuje účastníkům IS prohlížet si svoje osobní informace, které jsou uchovávány na serveru. Existuje funkcionalita pro odstranění bez možnosti obnovení (v případě nutnosti zachování určitých činností budou dané činnosti zanonymizovány).

FR03 Upozornění Aplikace posílá účastníkům IS upozornění po každé významné činnosti (například ohodnocení práce, archivace projektu…), jež se jich přímo týká. Upozornění existují pouze v rámci IS a představují jednoduchou textovou zprávu. Účastníci mají možnost zobrazit svoje upozornění a označit je jako přečtené/nepřečtené nebo úplně odstranit.

FR04 Projekt – Životní cyklus Aplikace umožňuje účastníkům IS zakládat vlastní projekty, ve kterých se automaticky stávají vedoucími. Po založení je možné vytvořit tým, definovat cíle a začít tvořit obsah. V případě neaktuálnosti projektu se dá odstranit, tím zcela zanikne, nebo archivovat, pak existuje možnost ho kdykoliv obnovit. Archivací se zakáže veškerá činnost uživatelů

na daném projektu kromě rušení archivace a odchodu uživatelů z projektového týmu.

FR05 Projekt – Správa dat Každý projekt je definován množinou dat, jež ho popisují a prezentují. Data se dělí na vždy veřejné, skryté a volitelně skryté. Mezi veřejné patří:

- kategorie,
- název,
- veřejný popis,
- seznam členů projektového týmu,
- tagy,
- status archivace.

Skryté (dostupné pouze pro členy) jsou:

- seznam iterací a úkolů,
- interní popis projektu,
- snapshoty iterací a jejich hodnocení.

Za volitelně skryté se považují vakantní pozice a samotný obsah.

FR06 Projekt – Kategorie a tagy Aplikace umožňuje třídění projektů do jednotlivých skupin s pomocí kategorií a tagů. Každý založený projekt povinně patří právě do jedné kategorie, která určuje jeho primární zaměření. Kategorie lze zakládat, upravovat a odstraňovat, pokud jsou prázdné. V případě nutnosti přesnější specifikace projektu je možné použít tagy, které budou primárně sloužit jako prostředek pro vyhledávání.

FR07 Projekt – Tým a role Na tvorbě obsahu projektu se podílí uživatelé s příslušným oprávněním. Pro přiřazování oprávnění slouží projektové role. Dle logického hlediska se dají rozdělit pouze na tři typy – vedoucí, spolupracovník a návštěvník.

Vedoucí mají neomezený přístup z hlediska práv. Prvním vedoucím projektu se vždy stává uživatel, jenž daný projekt založil. Každý vedoucí může prohlásit jiného člena týmu za vedoucího. Odejít z této role však lze pouze v případě, že v projektu zůstane alespoň jeden vedoucí.

Pravým opakem vedoucích jsou návštěvníci, kteří mají právo pouze prohlížet obsah, nikoliv do něj zasahovat. Volné přihlašování na roli návštěvníka lze zakázat, či povolit.

Speciálním typem role je spolupracovník, který slouží pouze jako nadskupina pro množinu kapacitně omezených pracovních míst. Pracovní místa jsou definovány vedoucím pro inzerci náboru členů týmu. V případě, že je otevřené přihlašování, běžní uživatelé se mohou dobrovolně hlásit na vakantní pozice. Pokud vedoucí má status důvěryhodného uživatele, tak může přidat člena týmu i bez jeho souhlasu.

FR08 Projekt – Vyhledávání Aplikace umožňuje uživatelům vyhledávat existující projekty podle zadaných kritérií, pokud vedoucí projektu povolil jeho vyhledávání.

FR09 Projekt – Iterace a úkoly Aplikace umožňuje definovat plánovaný průběh projektu v podobě iterací a úkolů. Iterace je dlouhodobá, má název a plánované datum dokončení, slouží jako nadskupina pro jednotlivé úkoly. Úkoly se skládají z názvu, popisu, maximálního a minimálního počtu bodů. V průběhu tvoření obsahu projektu jednotlivé části obsahu označují úkoly jako splněné.

FR10 Projekt – Správa obsahu Aplikace umožňuje vedoucím a spolupracovníkům projektu tvořit jeho obsah. Ten může být soukromý, neboli dostupný pouze pro členy projektu, nebo veřejný, v tomto případě je dostupný pro všechny účastníky IS.

Obsah je členěn do částí. Každá část tvoří samostatný celek, jenž se zakládá na určité šabloně (například šablona pro text, obrázek, graf), a může označovat jeden či několik úkolů za splněné. Seřazené části dávají výsledný obsah projektu, jenž se dá prohlížet online.

FR11 Projekt – Snímky iterací Aplikace poskytuje funkcionalitu vytvoření snímku určité iterace. Je to záznam aktuálního stavu všech částí obsahu, které splňují alespoň jeden z úkolů vybrané iterace. Tento záznam je dostupný pouze pro čtení a existuje nezávisle na obsahu projektu. Snímek lze po vytvoření odevzdat pro ohodnocení, které spočívá v přiřazování bodů s volitelným komentářem k jednotlivým úkolům snímku. Dané hodnocení snímku se promítá do celkového přehledu iterací projektu. V případě, že existuje víc

ohodnocených snímků jedné iterace, tak se bere nejnovější (nejpozději upravené). Existuje možnost změnit ohodnocení snímku, ale danou činnost může provádět pouze uživatel, jenž je původním hodnotitelem snímku.

FR12 Projekt – Důvěryhodnost Aplikace označuje projekt za důvěryhodný, pokud alespoň jeden z vedoucích je důvěryhodný. To se projevuje v podobě speciálního značení vedle názvu projektu.

FR13 Omezení obsahu IS pouze na projekty Informační systém je pouze služba pro správu prací, nebude poskytovat žádnou jinou funkcionalitu pro uchovávaní dat s rychle stárnoucí hodnotou. Tím se zamezí nutnost neustálého udržování aplikace z hlediska obsahu.

2.1.2 Obecné požadavky

NR00 Veřejné API Aplikace nabízí otevřené API (application programming interface) pro vývojáře klientských aplikací.

NR01 Dokumentace Součástí aplikace je vývojářská a uživatelská dokumentace.

NR02 Rozšiřitelnost Aplikace bere v ohled budoucí rozvoj aplikace a zvyšující se nároky na server.

NR03 Optimalizace uživatelského rozhraní Webové rozhraní aplikace je adaptované pro základní prohlížeče (Google Chrome, Mozilla Firefox, Apple Safari) a je responzivního nebo adaptivního typu optimalizované pro dvě velikosti obrazovek – mobilních telefonů (pod 1 000 px) a desktopových počítačů (nad 1 000 px).

NR04 GDPR Aplikace splňuje požadavky GDPR (General Data Protection Regulation).

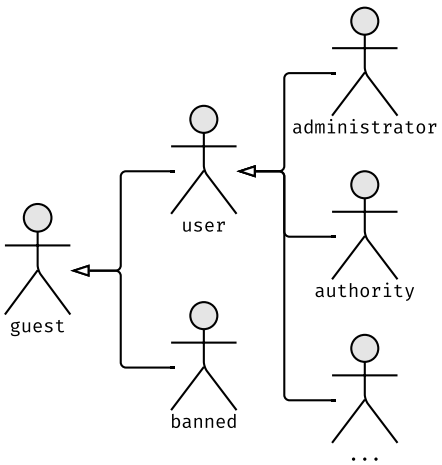
NR05 Jazykové verze Rozhraní aplikace bude dostupné v anglickém jazyce.

2.2 Uživatelské činnosti

2.2.1 Uživatelské role

Z funkčních požadavků plyne, že každý uživatel v systému má vždy přiřazenou právě jednu z globálních rolí. Tím se stanovují jejich práva na provádění určitých činností v rámci IS.

Základní množinu uživatelských rolí tvoří `administrator`, `banned`, `user`, `authority` a imaginární role `guest`, jež se přímo nevyskytuje v IS a pouze označuje neautorizované osoby. Diagram na obrázku 2.2 znázorňuje logickou dědičnost těchto rolí na základě přístupových práv. Administrátor (`administrator`) systému má povolené všechny typy činností, zablokovanému (`banned`) uživateli je povoleno pouze přihlášení a přehled soukromých dat. Přístupová práva ostatních rolí jsou smíšená a jsou podrobně definována v seznamu případů užití.

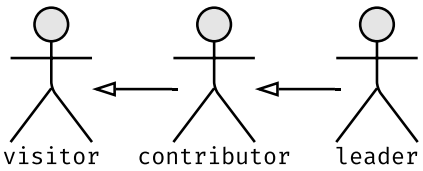


Obrázek 2.2: Diagram logické dědičnosti globálních uživatelských rolí v systému

V každém projektu uživatelé budou nabývat sekundárních rolí, jež budou mít vliv pouze na daný projekt. Mezi ně patří vedoucí projektu (`leader`), skupina uživatelsky definovaných rolí spolupracovníků (`contributor`) a návštěvníci (`visitor`) s právy pouze pro čtení. Jejich logická dědičnost je znázorněna ve formě diagramu na obrázku 2.3.

2.2.2 Případy užití

Případ užití neboli UC (use case) je činnost, k vykonávání které dochází ze strany uživatele IS (buď člověka nebo jiného systému). Jednotlivé činnosti jsou řazeny do dříve



Obrázek 2.3: Diagram logické dědičnosti uživatelských rolí v projektu

definovaných funkčních požadavků a během implementování jsou přímo promítány na jednotlivé typy oprávnění, jež budou přiřazovány globálním a projektovým rolím.

FR00 Identita uživatelů

UC00 `guest` se může zaregistrovat nebo přihlásit do IS.

FR01 Globální role

UC01 `administrator` může měnit roli uživatele.

FR02 Osobní informace uživatelů

UC02 Jakýkoliv účastník IS může prohlédnout svoje osobní informace.

UC03 Jakýkoliv účastník IS může odstranit svůj účet.

FR03 Upozornění

UC04 `user` může zobrazit, upravit stav nebo odstranit svoje upozornění.

FR04 Projekt – Životní cyklus

UC05 `user` může založit projekt.

UC06 `user` s projektovou rolí `leader` může odstranit projekt.

UC07 `user` s projektovou rolí `leader` může nastavit stav archivace.

FR05 Projekt – Správa dat

UC08 `user` s projektovou rolí `leader` může upravit veřejné informace a interní popis projektu.

FR06 Projekt – Kategorie a tagy

UC09 `user` s projektovou rolí `leader` může přidávat, upravovat a odstranovat tagy projektu.

UC10 `authority` může zakládat, upravovat a odstranovat kategorie, pokud neobsahuje ani jeden projekt.

FR07 Projekt – Tým a role

UC11 `user` s projektovou rolí `leader` může přidávat, upravovat a odstraňovat projektové role typu `contributor`.

UC12 `authority` s projektovou rolí `leader` může přidávat uživatelé do týmu.

UC13 `user` s projektovou rolí `leader` může měnit projektovou roli uživatele.

UC14 `user` s projektovou rolí `leader` může odstraňovat uživatelé z týmu.

UC15 `user` s projektovou rolí `leader` může povolovat nebo zakazovat volné přihlášení do projektové role `visitor` a rolí typu `contributor`.

UC16 `user` se může hlásit na vakantní místo.

FR08 Projekt – Vyhledávání

UC17 `user` může vyhledat projekt, pokud je projekt zařazen do seznamu pro vyhledávání.

UC18 `user` s projektovou rolí `leader` může přidávat nebo odebírat projekt ze seznamu pro vyhledávání.

FR09 Projekt – Iterace a úkoly

UC19 `user` s projektovou rolí `leader` může přidávat, upravovat a odstraňovat iterace a úkoly.

FR10 Projekt – Správa obsahu

UC20 `user` s projektovou rolí `contributor` může přiřazovat části obsahů úkolům.

UC21 `user` s projektovou rolí `contributor` může zakládat, upravovat a odstraňovat části v obsahu.

UC22 `user` s projektovou rolí `visitor` může zobrazit obsah.

FR11 Projekt – Snímky iterací

UC23 `user` s projektovou rolí `contributor` projektu může vytvořit snímek.

UC24 `user` s projektovou rolí `visitor` projektu může zobrazit snímek.

UC25 `user` s projektovou rolí `contributor` projektu může odevzdat snímek.

UC26 `authority` s projektovou rolí `leader` může ohodnotit snímek, pokud je odevzdaný.

UC27 `authority` s projektovou rolí `leader` může změnit hodnocení snímku, pokud je jeho hodnotitelem.

UC28 `user` s projektovou rolí `leader` může odstranit snímek, pokud nebyl ohodnocen.

2.3 Primární aktivity

Z celé množiny funkcionalit IS lze vyčlenit několik primárních aktivit, na které budou uživatelé klást největší důraz při práci. Dané činnosti, vzhledem k jejich důležitosti, jsou znázorněny do UML diagramů aktivit nebo podrobně zanalyzovány.

2.3.1 Autentizace uživatelů

Základní a prvotní činností IS je registrace či následné přihlášení neautentizovaných uživatelů. Vzhledem k absenci vlastního autorizačního serveru jsou využívány autorizační servery třetích stran.

Cyklus autentizace přes vybranou službu (viz obrázek 2.4) začíná uživatelským požadavkem na server. Informační systém přesměruje uživatele na stránky třetí strany, kdy v případě první návštěvy bude uživatel dotázán na povolení poskytovaní jistých osobních informací. V případě odmítnutí je ukončen celý proces autentizace. Pokud uživatel povolil poskytování informací, nebo ho již měl schválený, tak autorizační server posílá potřebná data na server IS.

Dle poskytnutých informací server ověřuje, zda se jedná o nového uživatele, pak zakládá nový účet se všemi náležitostmi (přiřazení výchozí role, oznámení apod.), nebo o již existujícího, potom kontroluje, zda není potřeba aktualizovat některé informace vzhledem k údajům (údaje, které nemají přímý vliv na identifikaci – jméno, email apod.).

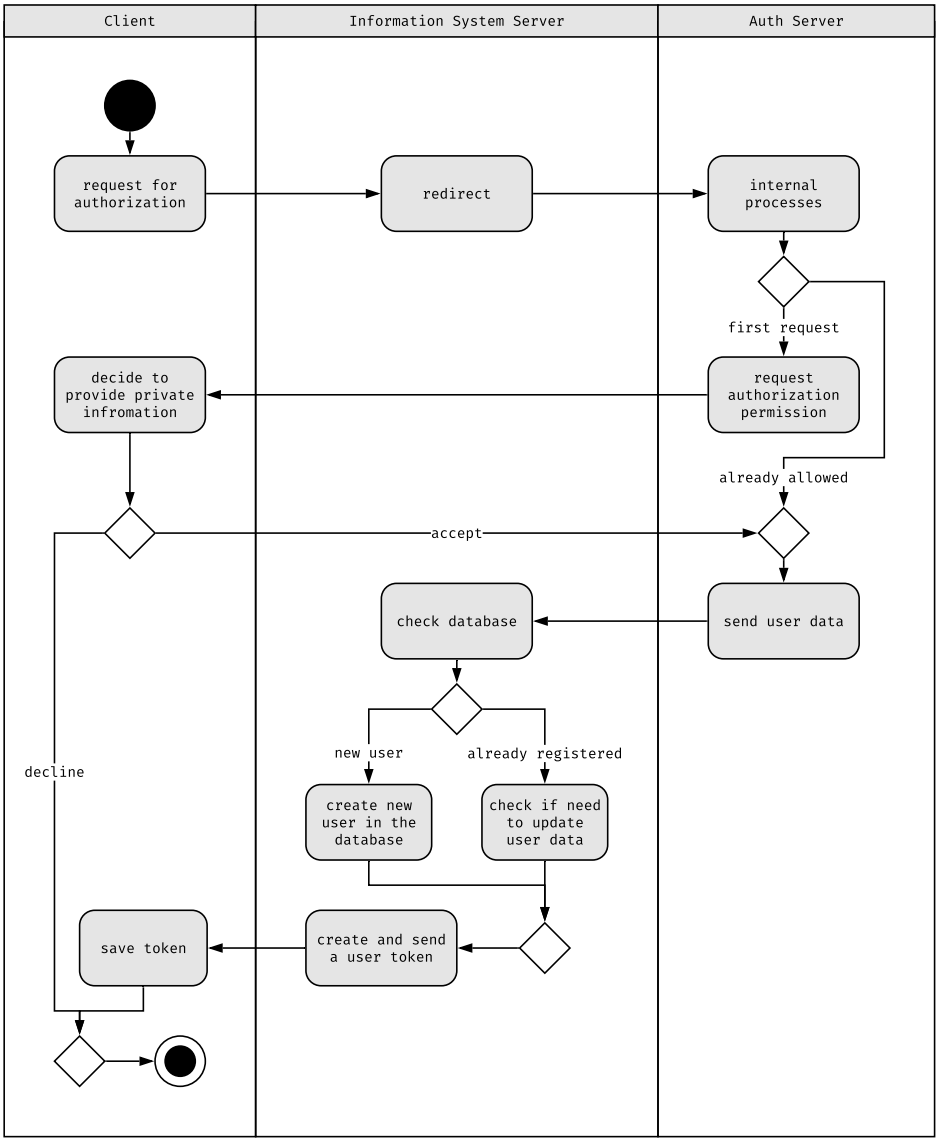
Po vytvoření, či aktualizaci databáze se vytváří unikátní autorizační klíč, jenž je následně předán klientské aplikaci. Tento klíč je uložen je uložen na klientovi a používán v případě autorizovaných požadavků na server.

2.3.2 Správa obsahu projektu

Hlavním cílem projektového týmu bude tvorba obsahu pro plnění úkolů v iteracích. Obsah se skládá z jednotlivých částí, které uchovávají svoje data – samotný text či jiné informace, název šablony, jenž bude v předem definované podobě zobrazovat informace části uživateli, a seznam úkolů, které tato část splňuje.

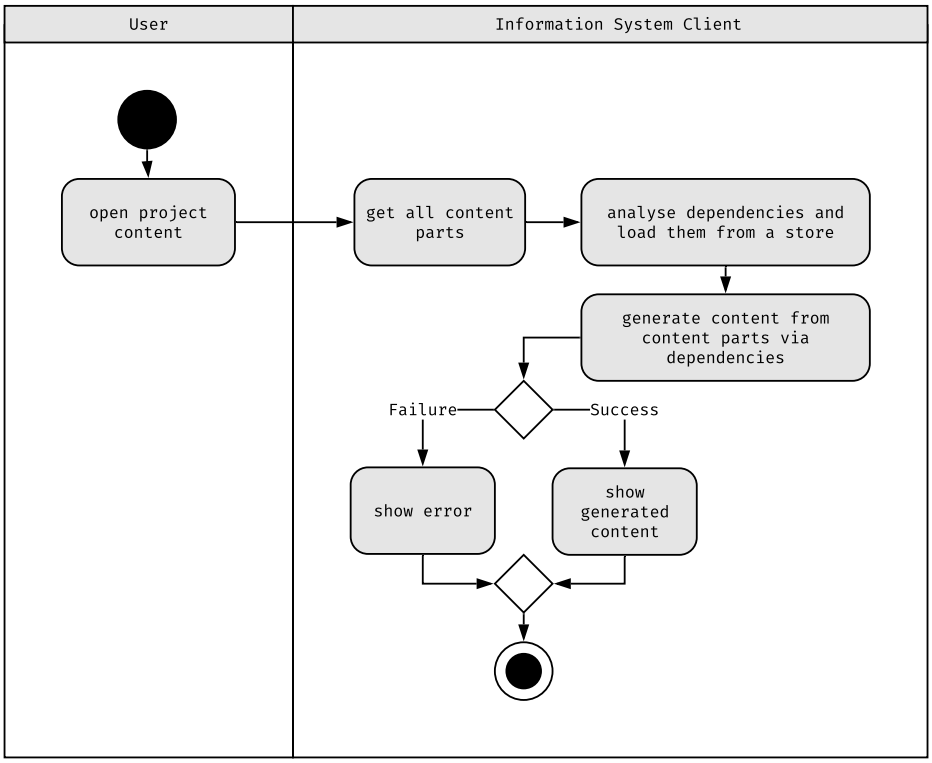
Vznik a správa částí je plně řízena vedoucími a spolupracovníky projektu. Návštěvníci a jiní uživatelé mohou pouze zobrazit obsah (pokud na to mají právo dle globální role a nastavení projektu).

V dané specifikaci všechny části mají být nezávislé, ale pro budoucí rozvoj je třeba počítat se snadno rozšiřitelnou strukturou, aby se dala nastavit výměna dat ve sdíleném úložišti. Celkový průběh generování obsahu popisuje diagram na obrázku 2.5 –



Obrázek 2.4: Diagram procesu autentizace

po požadavku uživatele zobrazit obsah jsou staženy části projektu do prohlížeče a zanalyzovány. Na základě analýzy vzniká seznam potřebných šablon, které jsou rovněž staženy ze serveru a následně použity pro generování obsahu z dat každé části. V případě chybějící nebo jinak poškozené šablony či dat části je zobrazena chyba.



Obrázek 2.5: Diagram generování obsahu projektu

2.3.3 Vyhledávání projektů

Pro většinu uživatelů bude veškerá činnost v IS začínat vyhledáváním potřebného projektu. Důvodem může být pokračování v tvorbě obsahu, snaha připojit se k existujícímu týmu, zobrazení existujících dat apod. Z uživatelského hlediska je vhodné mít pro všechny případy vyhledávání dva základní seznamy:

- seznam se všemi veřejně přístupnými projekty
- seznam projektů, v nichž je uživatel uveden jako účastník

Seznam přístupných projektů bude určen především pro vyhledávání cizích projektů a dle předpokladu bude méně využívaný, než seznam s uživatelskými projekty. Filtrování obou seznamů bude založeno na volbě obecných vlastností:

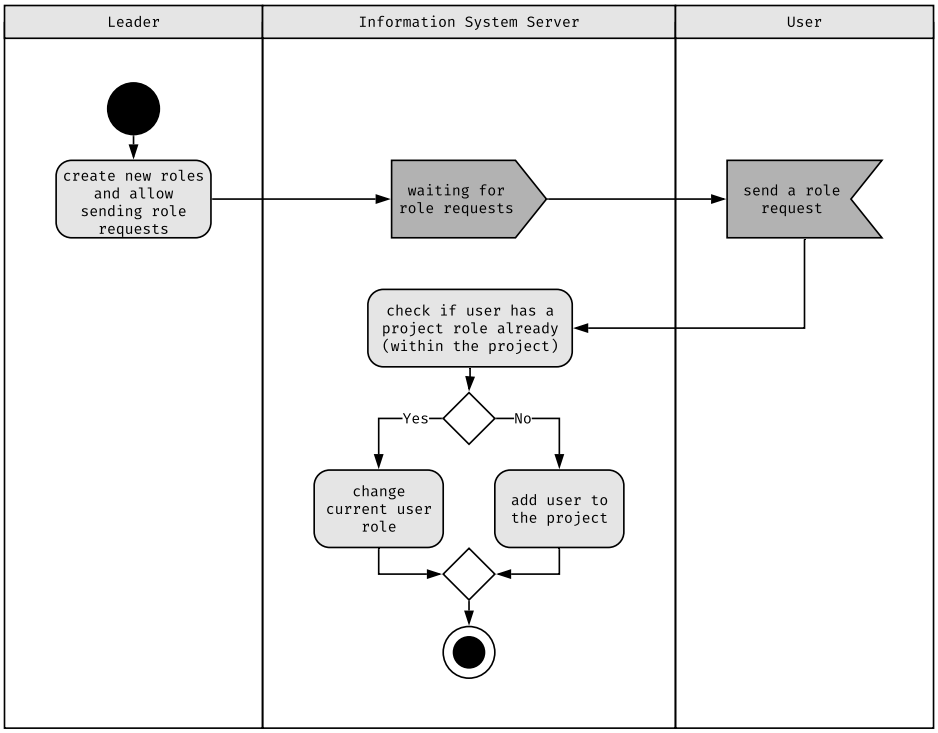
- kategorie,
- jméno projektu,
- stav archivace,
- stav volného přihlašování do projektu,
- projekty, do kterých má uživatel přístup.

Následně však může být upřesněno filtrováním dle volitelných tagů. Zavádění podkategorií nebo jiných přesně stanovených seznamů je nevhodné, protože by omezovalo uživatelskou činnost. Tagy představují volnou formu klíčových slov. Jsou ukládány při prvním použití a následně nabízeny všem uživatelům během upravování tagů.

2.3.4 Nabídka pozic

Poslední aktivitou, na kterou bude kladen důraz v IS, je týmová práce a nabídka vakantních míst. Diagram aktivit na obrázku 2.6 zachycuje průběh náboru vývojářského týmu. Nejdřív vedoucí definuje vakantní pozice, které jsou potřeba do projektového týmu a jejich kapacitu (například 5 volných míst pro vývojáře). Po otevření volného přihlašování do rolí se čeká, až uživatelé najdou projekt, projeví zájem o jednu z daných pozic a odešlou požadavek, že se o danou roli chtějí ucházet. IS se pokusí začlenit uživatele do týmu. V případě, že už je součástí projektu, tak pouze změní roli.

Daný proces přiřazování projektové role uživateli mohou ignorovat vedoucí projektů se statusem důvěryhodného účastníka systému. Mají právo přidávat uživatelé do vlastních projektů bez jejich souhlasu. V tomto případě rovněž neplatí omezení kapacity rolí.



Obrázek 2.6: Diagram průběhu náboru vývojářského týmu

Kapitola 3

Obecná analýza a návrh

Na základě požadavků na informační systém lze aplikaci rozdělit na dvě samostatné části. První je serverová aplikace s logickou složkou, jež bude poskytovat API pro klientské aplikace. Druhou nezbytnou část bude tvořit klientská webová aplikace využívající API a obsahující vlastní logiku pro rendrování[5] obsahu projektů.

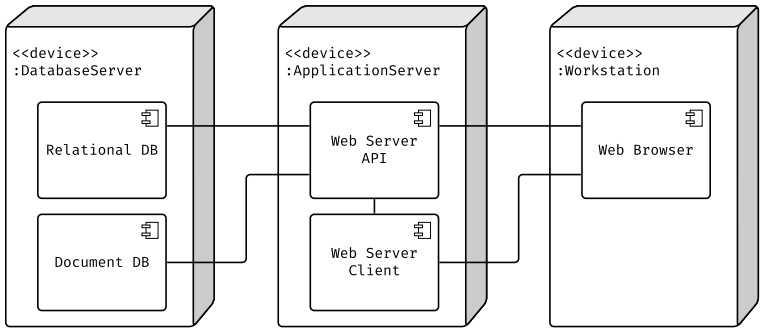
Celkový předběžný návrh fungování aplikace (viz obrázek 3.1) zahrnuje tři uzly: databázový server, aplikační server a pracovní stanici uživatele. Databázový server je využíván pro udržování instancí databázových systémů, do kterých patří relační SQL (Structured Query Language) databáze a dokumentová NoSQL (Not Only SQL) databáze. Aplikační server bude provozovat dvě navzájem komunikující aplikace – webový server s veřejným API a webový server poskytující uživatelské rozhraní. Server s veřejným API může zabezpečeně komunikovat s databázovými systémy. Dané rozdělení zajistí případný vývoj klientských aplikací pro jiné platformy. Poslední uzel představuje pracovní stanici uživatele, na kterém bude v prohlížeči spuštěno uživatelské webové rozhraní, které bude komunikovat s vlastním webových serverem a API.

Pro verzování obou aplikací bude využit VCS Git – jeden z nejpopulárnějších OSS VCS v posledních letech [12]. Výhodou Git je jeho decentralizovanost, která je velice důležitá pro vývoj OSS aplikací, protože poskytuje nezávislost jednotlivým vývojářům.

3.1 Nástroje a technologie

Kromě výše uvedeného nástroje pro správu verzí Git budou obě aplikace informačního systému využívat i další společné nástroje a technologie. Poslední část této kapitoly

[5] Zde proces, při kterém z předem získaných dat vzniká grafické uživatelské rozhraní.



Obrázek 3.1: Návrh struktury uzlů a komponent, které jsou na nich spuštěny

popisuje některé z nich s uvedením subjektivních důvodů, proč byly vybrány. Specifické nástroje, jež byly zvoleny pro implementaci pouze serverové nebo klientské aplikace, budou podrobně popsány v kapitolách 4 a 5.

3.1.1 JavaScript

Jako základní jazyk pro implementaci je vybrán JavaScript standardu ECMAScript 2016. Z hlediska klientské webové aplikace je to jediná efektivní možnost pro realizaci potřebné funkcionality. JavaScript na severu je spouštěn v NodeJS prostředí a byl zvolen pro přehlednost a jednotnost z hlediska využívaných knihoven. Některé části serveru jsou psány v JS-kompatibilním jazyku TypeScript.

3.1.2 Yarn

Správce balíčků je důležitou součástí větších projektů, dodává rozhraní pro správu a využití knihoven třetích stran. Pro jazyk JavaScript existuje několik správců, mezi které patří npm a Yarn. Dle osobních zkušeností byl vybrán Yarn.

3.1.3 Prettier

Prettier je nástroj zajišťující kvalitu kódu z hlediska vizuální stránky. Kontroluje zdrojové kódy více než 12 jazyků a knihoven, mezi které patří například JavaScript, JSX, TypeScript, SCSS, HTML a JSON [13], jež jsou využívány v implementaci IS.

Kapitola 4

Analýza a realizace serverové aplikace

Serverová aplikace je první ze dvou částí informačního systému. Tvoří centrum logické složky určené ve specifikaci a poskytuje veřejné API pro všechny klientské aplikace. Realizuje logickou část uživatelského systému, systému oznámení, správy a hodnocení projektů, tvorby týmů a dalších požadavků.

Daná kapitola popisuje tvorbu serverové RESTful aplikace. Postupně jsou analyzovány, navrhovány a popisovány implementace vybraných části služeb. Pro tyto účely jsou v některých aspektech parciálně testovány webové frameworky, databáze a možná implementační řešení s následným výběrem nejvhodnějšího řešení.

4.1 Architektura aplikace

Základní komponentou informačního systému je serverová aplikace poskytující API pro klientské programy. Obsahuje větší část logického celku. Z hlediska funkcionality lze aplikaci rozdělit do následujících vrstev:

Datová vrstva Datovou vrstvu definují datová úložiště a knihovny zprostředkovávající komunikaci mezi aplikací a úložišti.

Aplikační vrstva Aplikační vrstva představuje logickou část aplikace. Je realizována zejména s pomocí middlewarů a pomocných utilit.

Prezentační vrstva Prezentační vrstva v rámci dané aplikace není grafického typu. Představuje ji poměrně krátký zdrojový kód, jenž na základě obdržených dat

generuje odpověď ve formátu JSON (JavaScript Object Notation), která je následně odeslána webovým frameworkem.

Detailnějšímu návrhu architektury jsou věnovány následující podkapitoly, ve nichž je popsaný výběr nástrojů a způsob realizace některých funkcionalit.

4.1.1 Webový aplikační framework

Framework pro tvorbu API je centrálním systémem serverové aplikace. Jeho volba do jisté míry řídí celou architekturu, proto bylo rozhodnuto částečně zanalyzovat některé z nich pro výběr optimální varianty. Důležitá kritéria pro výběr byly:

- popularita na GitHub (existence komunity),
- minimalistické rozhraní (ne full-stack) frameworky,
- udržovaná a přehledná dokumentace,
- možnosti rozšíření frameworku dodatečnými utilitami.

Na základě těchto faktorů byly vybrány pouze 4 frameworky, jež se techniky nejvíc hodí k implementaci serveru IS:

- express,
- koa,
- restify,
- fastify.

Následný výběr byl řízen povrchovým testem rychlosti. S využitím každého frameworku byla napsána základní logická struktura (viz zdrojový kód 4.1), která se omezovala na jeden požadavek typu `GET` s jedním segmentem cesty a jedním parametrem. Výsledkem požadavku je vypsání odeslaného parametru s předpřipraveným textem.

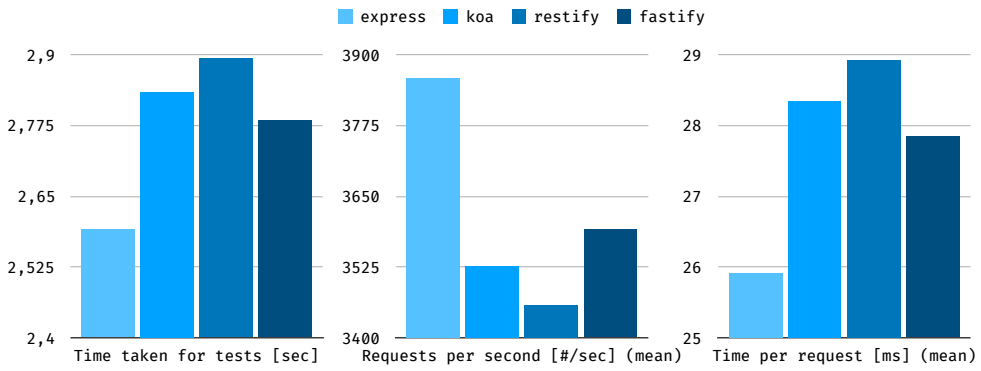
```
const app = require('express')();

app.get('/path/:param', (req, res, next) => {
   res.send('Param: ' + req.params.param);
});

app.listen(3000);
```

Zdrojový kód 4.1: Příklad testované aplikace s využitím frameworku express

Aplikace se zdrojovým kódem 4.1 a obdobné realizace s pomocí jiných frameworků byly postupně spouštěny pro 10 000 požadavků s konkurencí 100 požadavků. Jako testovací zařízení byl použit Mac mini 2014:

OS: macOS Mojave 10.14.3
Procesor: 2.5 GHz Intel Core i5
Operační paměť: 8 GB 1600 MHz DDR3

Z výsledků testování (viz obrázek 4.1) vyplynulo, že framework express je v daných podmínkách nejlepší volbou, tudíž je vybrán pro implementaci IS. Daný nástroj bude použit i v realizaci klientské aplikace, kde je do jisté míry doporučován vybraným SSR (server side rendering) frameworkem, jenž s ním uvádí řadu příkladů v dokumentaci [14].



Obrázek 4.1: Výsledky testování vybraných webových frameworků

4.1.2 Datová úložiště

Na základě specifikace lze definovat dva různé typy dat z hlediska využití. První skupina je tvořena například daty uživatelů, informacemi o projektu, snímkách obsahu apod. V této skupině je priorita kladena zachování závislostí neboli relací. Druhou skupinu tvoří text jednotlivých částí obsahu projektů. V tomto případě závislost je pouze jedna – na projektu, mnohem důležitější je možnost ukládání velkých objemů dat s měnící se strukturou.

Pro první skupinu dat je víc vyhovující jedna z relačních databází, protože na rozdíl od NoSQL databází samostatně spravuje cizí klíče a nabývá principů normalizace dat. Druhá skupina naopak vyžaduje jednu z NoSQL databází pro správu dokumentů. Kromě volnější struktury modelů, které se v daném případě ideálně hodí pro ukládání částí obsahu, poskytuje možnost horizontálního škálování databáze [15]. Daná

vlastnost bude vyžadována s rostoucími objemy uchovávaných dat, protože v porovnání s horizontálním škálováním, jež je typické pro relační databáze, je méně náročná z hlediska potřebných zdrojů [15].

Zástupci vhodných OSS relačních databází se staly MySQL a PostgreSQL. Vzhledem k mnoha existujícím článkům srovnání databází samostatná analýza nebyla potřeba. Výsledkem výzkumu jednoho z takových článků byl závěr, že oba zástupci DBMS (Database Management System) jsou stejně stabilní, ale PostgreSQL poskytuje navíc podporu některých NoSQL funkcionalit [16]. Volbou NoSQL databáze dle subjektivních zkušeností se stala MongoDB.

V průběhu implementace nastala potřeba definování dočasného úložiště pro výměnu autorizačních klíčů mezi serverem a klientem. Již zvolené databáze pro tento účel nebyly vhodnou volbou, protože by vznikla potřeba implementovat i démon pro automatické odstraňování předaných nebo nevalidních klíčů. Pro tento účel byla přidána další NoSQL databáze Redis, která se zaměřuje na uchovávání dvojic klíč-hodnota.

Všechny databáze, až na Redis nejsou napřímo využívány IS. Prostředníkem komunikace aplikace s PostgreSQL je knihovna TypeORM. Jedná se o ORM (object-relational mapping) knihovnu, jejíž úkolem je mapování databázových entit a jejich relací na objekty. Zjednodušuje obecnou komunikaci s databází a poskytuje rozhraní bez nutnosti psaní SQL příkazů. Mimo jiné zabezpečuje systém vůči útoku typu „SQL injection“. Obdobná knihovna mongoose, která vzhledem k databázi je typu ODM (object data modeling), je využívaná pro komunikaci s MongoDB.

Stejně jako v případě samotné aplikace je nutné verzovat i strukturu vybraných databází, zejména PostgreSQL, protože obsahuje přesné struktury entit, na rozdíl od MongoDB, která je uchovává převážně ve volném JSON objektu. Řešením jsou tzv. migrace – soubory, jež uchovávají rozdíl stavů před a po jejich aplikování. Migrace v IS jsou plně řízeny knihovnou TypeORM. Všechny příkazy pro vytváření, spouštění a reset migrací jsou přidány do souboru `package.json`, jenž definuje základní CLI (command-line interface) pro komunikaci s projektem.

4.1.3 Adresářová struktura

Adresářová struktura serverové aplikace je uvedená ve výpisu (viz zdrojový kód 4.2). Představuje jednoduché rozdělení na funkční zdrojové kódy v adresáři `src` a testovací soubory v adresáři `test`, jež kopírují strukturu `src` za účelem přehlednosti testovacích souborů. Adresáře `logs` a `tmp` uchovávají dočasné soubory, které nemají vliv na funkčnost aplikace.

```
.
├── logs            - logy aplikace
├── src
│   ├── api         - API routery
│   ├── config      - konfigurační soubory aplikace
│   ├── database    - modely, migrace, seed
│   ├── middlewares - middleware funkce
│   ├── routes      - statické routery
│   └── utils       - utility používané v aplikaci
├── test            - testovací soubory
├── tmp             - dočasné soubory
└── *.*             - inicializační a konfigurační soubory projektu
```

Zdrojový kód 4.2: Zkrácený výpis struktury složek serverové aplikace

4.2 API

Komunikace mezi serverem a klientem informačního systému probíhá prostřednictvím veřejně dostupného RESTful API. Důležitým faktorem je proto návrh intuitivně pochopitelné struktury URI (Uniform Resource Identifier). Zásadním pravidlem se stalo třídění přistupovaných entit do skupin dle jejich nejběžnějšího výskytu. V případě různých operací nad stejnou entitou je využíváno různých HTTP (Hypertext Transfer Protocol) metod:

GET Získání dat.
POST Vytvoření dat na základě určitých parametrů.
PATCH Aktualizace dat.
DELETE Trvalé nebo dočasné odstranění dat.

Pro generování odpovědi ze strany serveru je využívána šablona 4.3. Položka `typ` označuje stav odpovědi – úspěšná nebo neúspěšná –, `msg` obsahuje krátkou zprávu o výsledku prováděné činnosti, objekt `dat` nese v sobě užitečné informace, kvůli kterým byla volaná API metoda. Tento typ odpovědi není optimální z hlediska přenášení zbytečnych dat, byl ale zvolen kvůli zvýšení informativnosti odpovědi. Pro dokumentování API je vybrána aplikace Postman, exportovaná dokumentace je uložena na přiloženém médiu.

Vzhledem k veřejnému API je zřejmé, že k serveru budou přistupovat různé domény, případně stejná doména, jako má server, ale s odlišným portem. V běžném prostředí dochází k blokování sdílení informací mezi různými doménami kvůli bezpečnosti. Dané omezené je kladeno CORS (Cross-Origin Resource Sharing). [17]

```
{
   typ: "",
   msg: "",
   dat: {},
}
```

Zdrojový kód 4.3: JSON šablona pro odpověd serveru na uživatelský požadavek

Pro povolení využití sdílení zdrojů je třeba na straně serveru nastavit speciální hlavičky, jež budou odesílány se všemi odpovědi na požadavky. Na serveru CORS řeší speciální middleware spouštěný před každým zpracováním požadavku od klienta. Příklad nastavení HTTP hlaviček je uveden v ukázce 4.4.

```
res.header(
   'Access-Control-Allow-Origin',
   '*');
res.header(
   'Access-Control-Allow-Methods',
   'GET, POST, PATCH, PUT, DELETE, OPTIONS');
res.header(
   'Access-Control-Allow-Headers',
   'Origin, Accept, Content-Type, Authorization');
```

Zdrojový kód 4.4: Nastavení HTTP Access-Control-Allow hlaviček

Speciálním případem požadavku je `preflight`. Jedna se o HTTP požadavek typu `OPTIONS`, jenž se posílá samostatně na každé stránce před potřebným požadavkem [17]. Zjišťuje povolené domény, metody a jiné hlavičky. `Preflight` požadavek musí být vždy úspěšný, proto spolu s CORS hlavičkami je v middlewaru odchytáván před jakýmkoliv dalším zpracováním, aby bylo zajištěno, ze daný typ požadavku vždy proběhne a vrátí parametry serveru. Implementaci detekování a návrat hodnoty lze vidět na ukázce zdrojového kódu 4.5.

```
if (req.method === 'OPTIONS')
   return res.status(200).end();
```

Zdrojový kód 4.5: Zpracování `preflight` požadavku

4.3 Uživatelský systém

Dle specifikace aplikace implementuje samostatný uživatelský systém s funkcionalitou přístupových práv, nemá však vlastní autorizační server. Místo toho využívá vnější

servery třetích stran. Pro ukázkovou implementaci byl vybrán veřejný autorizační server GitHub OAuth 2.0, implementace ale počítá s připojením neomezeného množství obdobných služeb. Komunikace je zajištěna frameworkem PassportJS, jenž pro úplnou konfiguraci potřebuje pouze definování způsobu mapování dat uživatele z autorizačního serveru na entitu IS.

Po úspěšném přihlášení či registraci uživatele přes vybraný autorizační server vzniká potřeba autorizované komunikace mezi klientskou aplikací a serverem, proto je nutné zvolit způsob identifikace uživatele v systému. Jedním ze způsobů identifikace je využívání unikátního klíče během každého požadavku na server. Knihovna JSON Web Token slouží ke generování a ověřování takových klíčů. JWT (json web token) funguje na principu digitálních podpisů, tím jsou řešeny situace s udržování a správou uživatelských klíčů na straně serveru. Pro úspěšnou autorizaci je nutné pouze ověření digitálního podpisu během každého uživatelského požadavku. Veškerá následující činnost uživatele je řízena jeho právy v rámci IS.

Kapitola 5

Analýza a realizace klientské aplikace

Klientská aplikace, ať jde o jednoduché terminálové zpracování, nebo komplexní desktopový systém, má na jedno z prvních míst dávat realizaci pohodlného uživatelského rozhraní. V opačném případě celá aplikace ztrácí smysl, u uživatelů budou převládat negativní pocity, objeví se snaha najít něco jiného, co by víc odpovídalo jejich očekávání. S tímto je úzce provázána psychologická stránka lidí, jejich zvyklosti, zkušenosti s jinými programy, vnímání barev a jiné kognitivní aspekty. Každému z výše uvedených témat bude v dané kapitole věnována určitá část v souvislosti s návrhem IS.

5.1 Analýza uživatelů

Všechny existující klientské aplikace mají vždy svoji cílovou skupinu lidí, které ji využívají. Cílem vývojářů a návrhářů takových aplikací je poskytnout uživatelům prostředí, které bude v ideálním případě optimální ve všech směrech. Pokud jde o úplně novou aplikaci, tak prvotní návrh (prototyp) nebude nikdy vyhovující, protože není známo jak se budou uživatelé chovat a co všechno budou využívat. V případě úspěchu prototypu se postupem času bude vytvářet komunita, která bude do jisté míry řídit směr vývoje aplikace. Pomyslným cílem každého projektu by tak měl být ideální stav – uživatelé nic nechtějí a všechno perfektně funguje.

Aplikace implementovaná v dané práci spadá do první kategorie (prototyp). Proto se tato podkapitola věnuje základní analýze uživatelů, aby se stanovily přibližné požadavky z hlediska uživatelského rozhraní k již existující logické složce (API).

5.1.1 Psychologie percepce

Vnímání lidí se v různých částech světa liší, protože mají odlišné písmo, zvyky, životní úroveň apod. Proto je důležité zkoumat navrhovanou aplikaci i z psychologického hlediska. Implementace klienta dané práce se soustředí na prostředí střední Evropy. Nebude počítat s RTL (right-to-left) rozhraním a speciálním vybavením a funkcionalitou pro zdravotně postižené osoby (poruchy percepce, pohybu apod.).

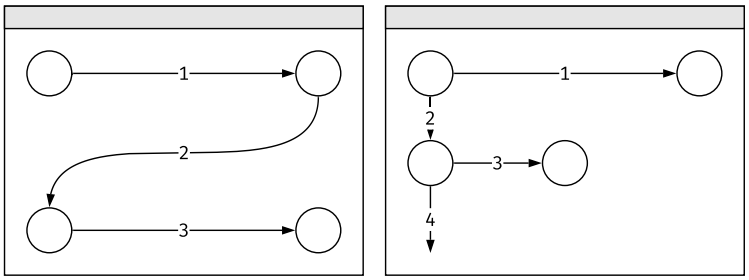
Stejně jako v případě prvního setkání s člověkem je v aplikacích důležitý první dojem, který je v tomto případě ovlivněn například dobou inicializace programu, převládajícími barvami, formou jednotlivých elementů a dalšími faktory. Přitom negativní emoce vyvolané aplikací jsou vidět mnohem lépe, než pozitivní [18]. Uživatelé nevnímají dobré rozhraní, protože je pro ně obvyklé z hlediska zvyklostí.

Oblasti web-designu se týkají především dva pojmy – UX (user experience) a UI (user interface). UX se věnuje tomu, aby dojem uživatele z použití aplikace byl co nejvíc příjemný, jinými slovy se stará o zlepšení uživatelských zkušeností. UI převádí tyto zkušenosti do vhodné formy a definuje základní barevné schéma. Barevná paleta se postupem času stává sekundárním identifikačním faktorem [19] aplikace. To se projevuje například ve vyhledávání programu v seznamu (v případě, že názvy jsou doprovázeny unikátním logem). Z tohoto důvodu odstíny a jejich kombinace, které jsou voleny během prvního veřejného prototypu, není vhodné kardinálně měnit, aby uživatelé nemuseli se opětovně přizpůsobovat novým podmínkám.

Dobrým příkladem z hlediska barevné palety je vizuální styl ČVUT [20]. Základní barvou je zvolena modrá, jež v kombinací s formou lva se stává unikátní pro danou instituci. ČVUT má i další, sekundární barvy, jež se mohou vyskytovat ve vizuálním zpracování materiálů univerzity, nebudou však s ní bezpodmínečně asociovány, pokud se použijí samostatně se stejnou formou (například bílý lev na zeleném pozadí).

Pro vizuální stránku IS byla primární barvou vybrána studená neintenzivní modrá barva přecházející do tmavší šedé, ukázku dané barvy lze vidět na obrázku 5.3. Spolu s bílou tvoří neutrální kombinaci vhodnou pro dlouhodobou práci. Rozhraní se proto snaží dodržovat především monochromatické prostředí, které v případě nutnosti vyvolání pozornosti používá komplementární barvy – žlutou nebo oranžovou. Sekundárními barvami pro vedlejší prvky byly zvoleny červená, zelená, stříbrná a modrá. Využívají se především na zvýraznění existence možné činnosti – například odeslat, vytvořit –, a informování o výsledku – úspěch, chyba, varování či zdržení.

Spolu s barvou je důležitá i forma od obecného rozvržení stránky až po formu jednotlivých elementů. V designu stránek pro cílové publikum zvyklé pro psaní LTR (left-to-right) jsou typické vzory F a Z [21] uvedené na obrázku 5.1. Jedná se o obdobu zlatého řezu v umění. Informační systém se drží vzoru F, který je v tomto případě vhodnější z hlediska rychlosti poskytovaní užitečných informací.



Obrázek 5.1: Z-vzor a F-vzor uživatelského rozhraní

5.1.2 Cílová zařízení

Klientské webové aplikace jsou zpravidla zobrazovány běžnými uživateli v GUI (graphical user interface) prohlížečích na zařízeních s odlišnou velikostí monitoru. Dle statistik portálu StatCounter z hlediska zařízení používanými uživateli v posledních letech převládají mobilní [22]. Tudíž větší zastoupení mají menší velikosti obrazovek. Z tohoto důvodu je adaptace uživatelského rozhraní IS pro mobily nutná, avšak vzhledem k povaze projektu důraz bude kladen především na desktopové velikosti obrazovek. Dle předpokladu pouze malé procento lidí bude schopno psát projekty na mobilu. Menší zařízení budou využívány pouze pro informační účely. Pokud by v pozdější době vznikla potřeba vytvořit přesnější rozhraní, tak by bylo vhodné zvážit spíš plnohodnotné mobilní aplikace, než pouze webové rozhraní. Z hlediska prohlížečů je dle obecných požadavků nutné adaptovat IS pro prohlížeče:

- Google Chrome,
- Mozilla Firefox,
- Apple Safari.

Většina nekompatibilních funkcionalit bude řešena automaticky s pomocí podpůrných nástrojů. Mezi problémová místa bude patřit zejména přizpůsobení CSS (Cascading Style Sheets) a interpretace standardů ECMAScript 6 a ECMAScript 7.

5.2 Architektura aplikace

Uživatelské rozhraní IS představuje převážně tenký klient typu webové aplikace dostupný přes webový prohlížeč. Část klientu týkající se zobrazování obsahu projektu se však blíží chování tlustého klientu, protože definuje logiku generování a vykreslování částí obsahu.

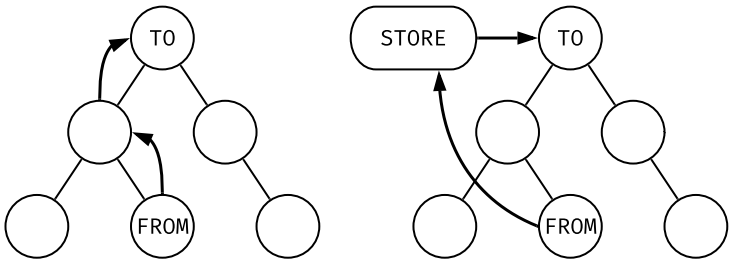
Vzhledem ke komplexním požadavkům aplikace a potřebě pokročilých funkcí základní jednoduchá implementace bez využití frameworků a knihoven není vhodná. Jednou z přijatelných variant architektury aplikace je SPA (single-page application) model, který je výhodný z hlediska rychlého načítání stránek a minimalizace stahovaných informací ze serveru. Negativním prvkem SPA je však velice pomalé načítání klientské aplikace. Pro první zobrazení jakékoliv stránky je nutné stáhnout celý zkompilovaný soubor aplikace, jenž v případě složitých systémů může dosahovat velikosti několika MB. Kromě načítání daného souboru je většinou ještě potřeba asynchronně dodatečně stáhnout potřebná data ze serveru přes API. To se stává problémem na zařízeních s omezenou rychlostí připojení. Pro zobrazení čehokoliv je nutné vyčkat úplného stažení všech závislostí.

Jedním z řešení prvotního načítání jsou SSR frameworky. Základní koncept jednotné aplikace zůstává zachovaný, avšak princip načítání první webové stránky se mění. Základní stránka s potřebnými daty je generována již na serveru a předávána jako statický HTML (Hypertext Markup Language) dokument. To napodobuje chování původních webů – uživatel dostává soubor typu `text/html`, jehož je schopen ihned zobrazit. Veškeré logické chování SPA aplikace se načítá až po zobrazení užitečného obsahu. Kromě rychlého uživatelského přístupu má toto chování i vliv na SEO (search engine optimization) indexaci stránek webovými vyhledávači. V případě daného IS však nemá význam, protože se jedná o aplikaci s chráněným obsahem.

Na základě této analýzy byla vyzkoušena implementace několika menších SPA a SSR aplikací s využitím React a Vue. Výsledkem výběru se stal React s pomocným frameworkem pro SSR NextJS, protože dle subjektivního hlediska byly nejvhodnější pro realizaci daného webového klienta.

K vybraným nástrojům NextJS a React je třeba mít prostředek pro sdílení dat mezi komponentami. V základním React je tato potřeba řešena přes tzv. „lifting“, který je sice jednoduchý, avšak nepřehledný. Pro předání informace z kořenové komponenty a zpět je nutné posílat potřebná data nebo funkci přes všechny mezikomponenty ve virtuálním DOM React [23]. To vede k psaní zbytečného kódu. Místo daného

způsobu lze využít jednu z knihoven pro vytváření společného úložiště. Typickým nástrojem je Redux (viz obrázek 5.2).



Obrázek 5.2: Ukázka rozdílu komunikace mezi komponenty bez (vlevo) nebo s (vpravo) využitím Redux

Z daného výběru nástrojů vyplývá, že základním jazykem pro realizaci klientské části je JavaScript. Architektura je postavena na CommonJS a ECMAScript 6 modulech, které jsou importovány do různých částí aplikace podle potřeby. To zajišťuje přehlednost a jednoduchou podporu. Dle funkcionality lze aplikaci rozdělit do následujících vrstev:

Inicializační vrstva Inicializační vrstva je řízena NextJS, z logického hlediska se jedná o spojení datové a prezentační vrstvy. Její chování se mění na základě prostředí, ve kterém je spouštěna (server nebo klient). Vždy stahuje a připravuje data z API serveru, ale pokud se jedná o serverové prostředí, tak generuje i obsah typu `text/html`.

Prezentační vrstva Prezentační vrstva je součástí React komponent, definuje generování prvků na webové stránce.

Aplikační vrstva Aplikační vrstva je rovněž součástí React komponent, definuje logické chování DOM (Document Object Model) elementů.

5.2.1 Použité technologie

Kromě primárních frameworků zvolených během návrhu architektury pro dosažení potřebné funkcionality byly zvoleny následující nástroje a technologie:

Jazyk popisu kaskádových stylů Základní CSS je příliš přímočarý pro popis složitějších struktur. Jeho náhradou se mohly stát jazyky SCSS (Sassy CSS), LESS (Leaner Style Sheets), Stylus a další, jež přidávají do CSS syntaxe logické struktury – podmínky, cykly apod. Dle subjektivních zkušeností byl zvolen SCSS.

Vykreslování tabulek s pokročilým vyhledáváním Pro danou funkcionalitu byly vhodné dva frameworky – DataTables a React Table. Nejdřív byl integrován DataTables, funkcionalita vyhovovala požadavkům, avšak implementace byla náročná a nepřehledná, protože docházelo k prolínání jQuery a React syntaxe. Z tohoto důvodu byl framework DataTables nahrazen React Table.

Knihovna popisu rozvržení stránky O rozvržení stránek se stará částečně importovaná knihovna Bootstrap.

Kompilace projektu Kompilaci celého projektu řídí BM (bundle manager) Webpack, jenž je vyžadován i frameworkem NextJS. Kromě výchozích nastavení je použit pro definování `alias` proměnných kvůli přehlednějšímu importu JavaScript modulů.

5.2.2 Adresářová struktura

Adresářová struktura klientské aplikace uvedená ve výpisu (viz zdrojový kód 5.1) se řídí požadavky frameworku NextJS. Složky `pages` a `static` se zdrojovými kódy nelze přesouvat do podadresářů, všechny ostatní závislosti se nachází v `src` a jsou většinou roztříděny dle typu (redux actions, redux reducers, konfigurační soubory aplikace...). Výjimku smíchaných typů zdrojových kódů tvoří složky s React komponenty – šablony stránek a sekundární komponenty. V těchto výjimkách jsou pro přehlednost sloučeny do jedné složky JavaScript soubory komponenty a její SCSS styly. Stejným způsobem jsou strukturovány i složky s jednotlivými stránkami. Bez BM by hlídání všech závislostí v takové struktuře bylo problematické, používaný Webpack s pluginami je však nastaven pro automatické vyhledávání specifických souborů. Vývojáři zbývá pouze dodržovat určitou strukturu aplikace.

```
.
├── _templates     - šablony zdrojových kódů (stránky, redux apod.)
├── pages          - nutná složka NextJS pro stránky
├── src
│   ├── actions    - redux actions
│   ├── components - sekundární React komponenty a SCSS styly
│   ├── config     - konfigurační soubory aplikace
│   ├── layouts    - základní šablony stránek
│   ├── reducers   - redux reducers
│   ├── store      - redux store
│   ├── styles     - globální SCSS styly
│   └── utils      - utility používané v aplikaci
├── static         - nutná složka NextJS pro statické soubory
└── *.*            - inicializační a konfigurační soubory projektu
```

Zdrojový kód 5.1: Zkrácený výpis struktury složek klientské aplikace

5.3 Realizace funkcionalit

Převážná část webového klientu je jednoduché zobrazování dat předávaných z serverem, případně naopak volání API metod. Na základě uživatelských informací uložených v JWT, mezi které patří i seznam práv, se zobrazují, či naopak skrývají jednotlivé React komponenty. Jedinou implementačně zajímavou funkcionalitu tvoří zpracování částí obsahu interprety s následným generováním uživatelsky připraveného rozhraní.

5.3.1 Generování obsahu projektu

Každý založený projekt má svůj vnitřní obsah, který v průběhu existence projektu vytváří jednotliví uživatelé. Při načítání obsahu projektu uživatelem se ze serveru stahují jednotlivé části. Jedná se o struktury ve formátu JSON, které obsahují data pro generování uživatelsky přijatelného obsahu (s grafickým rozhraním) a název jednoho z interpretů obsahu (v specifikaci nazývaný jako šablona). Příklad nejjednodušší JSON struktury je uveden ve zdrojovém kódu 5.2. Proměnné `interpreter` a `store` jsou povinné pro každou část, vnitřní struktura `store` už se však řídí vybraným interpretem.

```
{
   "interpreter": "text-plain",
   "store": {
      "title": "Nunc in elit",
      "text": "Pellentesque eu sapien erat."
   }
}
```

Zdrojový kód 5.2: Příklad JSON struktury části obsahu

Vzhledem k tomu, že se části obsahu uchovávají v NoSQL databázi, tak nejsou omezovány relacemi a přesně zadaným obsahem. Mongoose ODM, která je zprostředkovatelem spojení serverové API aplikace a MongoDB, definuje pouze nutně potřebnou strukturu v `schema` objektech, zbytek se automaticky přizpůsobuje všem novým typem interpretů a struktury proměnné `store`. Tím je zajištěna požadovaná ohebnost obsahu.

Po načtení všech částí se generuje pole unikátních názvů interpretů, jež jsou použity v částech. Na jeho základě se s použitím nakonfigurované cesty k úložišti vkládají `<script>` tagy do `<head>` stránky. Je to potenciální místo pro útok na uživatelé systému, protože dovoluje přidat nebezpečný kód do jednoho z interpretů, který se automaticky spustí na všech zařízeních, jež ho načtou s obsahem IS. Z tohoto důvodu je nutné používat pouze úložiště s omezeným přístupem. V prototypu implementace

je využíván veřejný GitHub repozitář s přístupem k souborům přes službu jsDelivr. Tato sekundární služba je nutnou součástí přístupu k souborům v repozitáři, protože GitHub vrací všechny soubory s MIME (Multipurpose Internet Mail Extensions) typem `text/html`, což znemožňuje jejich vykonávání v roli JavaScript skriptu. jsDelivr je OSS CDN (Content Delivery Network) služba, která dané omezení řeší a poskytuje kopie souborů repozitáře.

```
(() => {
   // Unique name
   const name = 'text-plain';

   // Render function
   const render = (store) => {
      return `${store}`;
   }

   // Declare interpreter
   if(!window.iterations) window.iterations = {};
   if(!window.iterations.interpreters)
      window.iterations.interpreters = {};
   window.iterations.interpreters[name] = {
      render
   };
})();
```

Zdrojový kód 5.3: Ukázka nejjednoduššího příkladu interpretu

Interpret v základní podobě (viz zdrojový kód 5.3) představuje samospouštěcí lambda funkci (tím je zajištěna ochrana vůči konfliktu globálních proměnných), jež má unikátní jméno a povinnou metodu `render`, která přijímá jako parametr `store` části obsahu a výsledkem poskytuje HTML kód. Během načítání vytváří odkaz na svoje metody v globální proměnné prohlížeče `window.iterations.interpreters`, odkud jsou volány IS.

Podobná struktura interpretu obsahu z hlediska rozšiřitelnosti je omezena pouze možnostmi jazyka JavaScript v prostředí prohlížeče. Kromě povinné metody `render();` je možné definovat podpůrné funkce, manipulovat s již existujícím obsahem stránky, přidávat knihovny třetích stran, vytvářet interaktivní grafy, 3D vizualizace a další funkcionalitu.

Ve specifikaci správy obsahu IS je uveden požadavek pro návrh struktury, která by se dala rozšířit i pro komunikaci mezi jednotlivými částmi. To v daném případě lze realizovat úpravou cyklu generování obsahu. Po stažení potřebných věcí se rendering nebude provádět rovnou, před ním může být vytvářena například dočasná uživatelská databáze typu IndexedDB, která bude zprostředkovávat veškerou komunikaci. Samozřejmě bude nutné doplnit části i interprety o další povinné prvky.

5.4 Uživatelské rozhraní

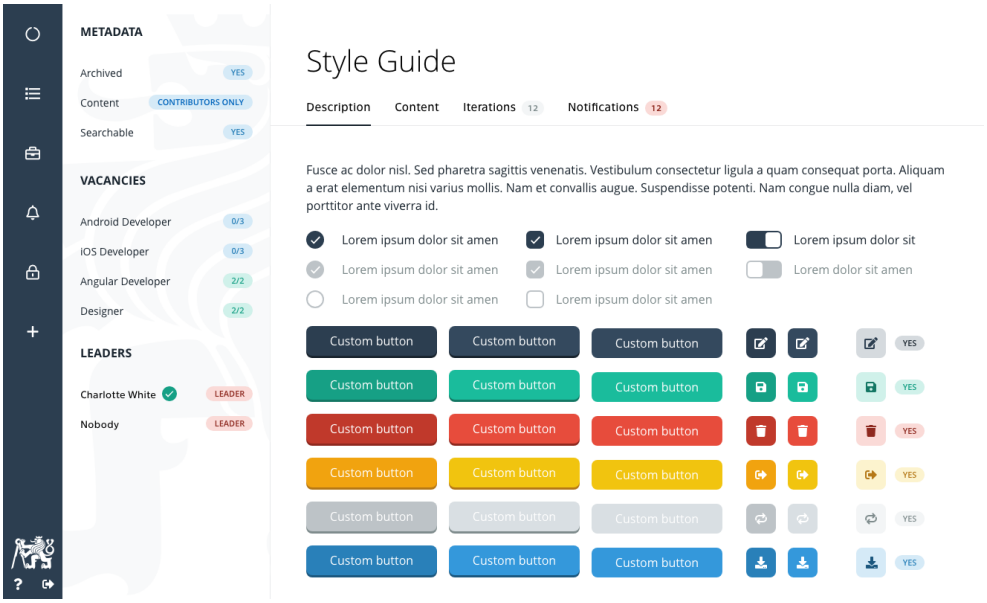
Na základě předchozích rozhodnutí o architektuře aplikace a realizace jednotlivých funkcionalit je navrženo vyhovující rozhraní. Základním východiskem je koncept plochého designu[6], jenž se projevuje ve snaze o zobrazení ovládacích prvků v přehledné, zjednodušené formě a zároveň skrytí méně potřebných funkcionalit bez většího dopadu na komfortní ovládání aplikace. Byl preferován před ostatními styly, protože obvykle nevyžaduje žádné dodatečné prvky a může být plně popsán s využitím pouze CSS, tudíž je nenáročný z datového hlediska.

Před navrhováním jednotlivých stránek byl vytvořen grafický manuál[7]. Jedná se o pomocnou stránku obsahující všechny základní prvky vyskytující se v designu – barevná paleta, styly textů, nadpisy, prvky formulářů apod. Ukázku části manuálu lze vidět na obrázku 5.3, elektronická verze[8] je dostupná v příloze této práce.

[6] Běžnější je anglický výraz „flat design“

[7] Zde je přesnější anglický výraz „style guide“

[8] Pro všechny grafické návrhy uživatelského rozhraní je využíván program Adobe XD



Obrázek 5.3: Ukázka části grafického manuálu s aplikovaným stylem pro České vysoké učení technické v Praze

Kapitola 6

Právní náležitosti

Daná kapitola zkoumá vliv legislativy na oblast webových aplikací, jaké nároky jsou kladeny v současné době a co je potřeba split v případě implementace IS pro akademické instituce. Zaměření kapitoly je pouze na Obecné nařízení na ochranu osobních údajů, ochrana autorských práv a další náležitosti nejsou předmětem analýzy.

Rozvoj oblasti webových služeb v posledních letech způsobil, že společnosti (například Facebook, Inc. a Twitter, Inc.) i jednotlivci působící na internetu sbíraly na vlastních serverech data dobrovolně poskytované jejich uživateli. Vznikla situace, kdy dané subjekty nebyly omezovány zákony a mohly do jisté míry nakládat se shromaždovanými daty dle jejich potřeb. Tuto situaci bylo potřeba řešit. Jedním z významných kroků, který změnil povinnosti subjektů (včetně majitelů webových služeb), jež shromažďují nebo zpracovávají osobní údaje občanů zemí EU (Evropská unie) se stal den 25. května 2018, kdy nabylo účinnosti Obecné nařízení na ochranu osobních údajů neboli GDPR [24].

GDPR přináší řadu povinností, dle článku [25] musí každá služba ovlivněná tímto nařízením poskytovat uživatelům práva pro přístup k osobním údajům v přehledné formě, dávat možnost opravy těchto údajů v případě nesprávnosti, případně odstranit všechny osobní údaje ze systémy. Za osobní údaje se v tomto případě považují všechny informace, na základě kterých lze identifikovat konkrétní osobu. Mohou mezi ně patřit konkrétní údaje (jméno, mobilní číslo) nebo skupina obecných (popis osoby) [26].

Informační sytém navrhovaný a implementovaný v dané bakalářské práci se snaží omezit množství uchovávaných osobních informací. Autorizace uživatelů probíhá přes autorizační servery třetích stran a uchovává pouze jejich unikátní ID (Iden-

tification Data) v číselné podobě. Danou informaci může uživatel systému sám anonymizovat (pro uživatelé daný krok znamená výmaz účtu). Úplné odstraněné však z technického hlediska není možné, protože během existence účtu vznikají data, jež nepatří pouze uživateli (obsah projektů, tvorba a hodnocení iterací). Jsou nezbytně nutné pro plynulý běh IS. Nezávislá data typu upozornění jsou však odstraněna navždy. Jednotný přehled všech uživatelských informací je poskytován na stránce FAQ (frequently asked questions) informačního systému. Případné opravy údajů nejsou v danou chvíli zautomatizovány, budou řešeny osobně v komunikaci s poskytovatelem služby.

Sepsaní přehledu zpracování osobních údajů a podmínek ochrany osobních údajů, které jsou rovněž nutné pro poskytování webových služeb, jsou mimo rozsah dané práce, protože vyžadují pokročilejší právní znalosti.

Kapitola 7

Testování, nasazení a dokumentace

7.1 Testování

Testování je nedílnou součástí každé aplikace. Je nutné pro zajištění kvality kódu a správnosti poskytovaného produktu. Obecně se dá rozdělit na dva základní typy – manuální, jež spočívá v testování každé funkcionality a změny lidmi, a automatické, které využívá předem připravených programů nebo skriptů. Manuální testování oproti automatickému je přesnější a může poskytnout mnohem větší spektrum hodnocení, není však rychlé a je ovlivněno lidským faktorem. Automatické testování je proto preferováno vždy, když není potřeba mít zpětnou vazbu přímo od uživatele.

7.1.1 Automatické testování

Obě části IS jsou připravené pro automatické testování pro ověřování základních funkcionalit systému. Z hlediska výběru testovacího frameworku bylo vybíráno mezi Jest a Mocha. Po vyzkoušené integraci obou bylo rozhodnuto ponechat Jest, protože samostatně poskytoval veškerou potřebnou funkcionalitu. Mocha z tohoto hlediska potřebovala dodatečné knihovny.

7.1.2 Manuální testování

Vzhledem ke komplexnosti a časové náročnosti projektu manuální testování poskytuje jedinou možnost precizního testování aplikace prostřednictvím akceptačních testů uvedených na přiloženém médiu. Je časově náročnější, ale poskytuje širší spektrum hodnocení. V této souvislosti skupina lidí byla požádána provést předem připravené scénáře a oznámit nalezené chyby či jiné problémy. V závislosti na výsledcích byla

provedena korekce rozhraní. V budoucnu drtivá většina akceptačních testů má být nahrazena automatickými testy.

7.2 Nasazení

Způsob nasazení daného informačního systému je popsán ve vývojářské dokumentaci. Pro nasazení IS je přepokládána dispozice virtuálního serveru s operačním systémem linux. Spuštění instancí databází je řízeno technologií kontejnerizace Docker. Příklad konfiguračního souboru se nachází v kořenovém adresáři serverové aplikace. Pro zajištění stabilního prostředí s detailnějším monitorováním procesů jsou aplikace spouštěny v pm2 – démonu pro správu procesů.

Daný případ konfigurace serveru není optimální. Pro dlouhodobý běh aplikace je nutné preciznější nastavení databází, omezení přístupu k serveru přes určité porty, zavedení automatického zálohování, systém oznámení o stavu aplikací, povolení HTTPS (Hypertext Transfer Protocol Secure) protokolu a další náležitosti.

7.3 Dokumentace

Informační systém je dokumentován dvěma způsoby – pomocí komentářů v samotném zdrojovém kódu a nezávislých příruček, jež obsahují všeobecné informace. Formát komentářů se řídí dle norem generátorů dokumentací. Pro serverovou aplikaci psanou v jazyce TypeScript se jedná o generátor TypeDoc, v případě webového klienta psaného v JavaScript jde o podobný generátor `JSDoc`. V případě potřeby existuje možnost vytvoření manuálu na základě okomentovaných částí.

Dodatečné příručky tvoří samostatné celky spravované frameworkem `docsify`. Uživatelská příručka popisuje vybrané případy užití na základě webového klienta. Vývojářská dokumentace zachycuje detaily, jež by bylo nevhodné dávat do zdrojového kódu[9].

[9] Kopie obou příruček jsou dostupné na přiloženém médiu.

Kapitola 8

Rozvoj informačního systému

Vzniklá implementace aplikace je prvním prototypem – beta verze, jež potřebuje zčásti provést optimalizaci a následně otestovat v reálném nasazení se skutečnými uživateli, aby se mohly detekovat chyby, na které se nepřišlo během umělého testování (automatického i uživatelského).

Informační systém dle způsobů rozvoje lze rozdělit na dva hlavní směry – obecné univerzální zdokonalování a rozvoj se zaměřením na FIT ČVUT, na základě analýzy služeb kterého byl navržen daný IS.

DO00 Podpora internacionalizace a lokalizace Aplikace v současném stavu podporuje pouze jeden jazyk – angličtinu. Je potřeba vytvořit úložiště pro překládané výrazy, napsat implementaci middleware pro serverovou část klienta realizující detekování aktuálního jazyka uživatele (například s pomocí URI) a přidat dosazování přeložených výrazů během SSR i CSR (client side rendering). V případě nutnosti realizovat i jiné potřebné formátování.

DO01 Hromadné zakládání projektů Zakládání projektů se ne vždy omezuje založením jednoho dobrovolného projektu s jedním počátečním uživatelem v roli vedoucího. V případech zakládání projektů pro skupinu studentů nebo vyhlašováním otevřeného přihlašování na množinu semestrálních prací s otevřeným zadáním se nabízí možnosti hromadného zakládání projektů.

Založit dobrovolné projekty hromadně – definují se společné metainformace a počet kopií projektů. Výsledkem je skupina identických projektů s otevřeným přihlašováním.

Založit povinné projekty hromadně – definují se společné metainformace projektů a množina uživatelů. Výsledkem je množina založených identických projektů, pro každého uživatele jedna kopie. Pouze pro globální role s pokročilými možnostmi správy uživatelů.

DO02 Serverová implementace snímků Je třeba dokončit serverovou implementaci případů užití týkajících se plnění úkolů, vytváření snímků iterací a odevzdávání pro hodnocení. Realizace daných funkcionalit vykazovala v některých případech neočekávané chování, čím narušovala plynulou uživatelskou interakci. Uživatelské rozhraní je třeba upravit do finální podoby.

DO03 Nepříznivé scénáře API dotazů Aplikace není propracovaná z hlediska ošetřování nepříznivých situací (například rozpoznání typu chyby požadavku). Je nutná realizace detailnějších návratových hodnot API a ošetřování daných situací s vhodným doplněním uživatelského rozhraní (včetně odblokování již implementovaných případů užití na straně serveru).

DO04 Nové interpretátory částí obsahů a integrace se službami třetích stran Interpretátory částí se vždy vážou na jeden určitý typ obsahu a jsou omezovány ukládáním informací pouze ve svém nebo serverovém úložišti. Pro rozšíření možností generování obsahu projektu je potřeba napsat víc typů interpretátorů (například zobrazování tabulek). V případně vhodných úprav životního cyklu rendrování obsahu je možné realizovat nahrávání či ukládání dat na vnější úložiště typu Google Drive, YouTube, Dropbox, Lucidchart a další.

DO05 Analýza využití systému a aktualizace UX V případě rozvoje sytému a zvyšování počtu uživatelů je třeba postupně zlepšovat i UX klienta. To bude možné po implementaci skriptů pro sbírání statistik. Na základě jejich výsledků lze provést potřebné změny.

DO06 Optimalizace stávajícího systému Vzhledem k nedokonalosti stávající implementace je potřeba provést optimalizaci některých funkcionalit. Mezi již známá slabá místa patří odesílání všech výsledků vyhledávání jedním souborem a absence automatického obnovování JWT po vypršení doby platnosti.

FIT00 Integrace CVUT OAuth 2.0 serveru Aplikace již počítá s integrováním jiných autorizačních serverů. V serverové části stačí přidat novou PassportJS strategii s vhodnou konfigurací pro přesměrování na autorizační server a realizo-

vat mapování dat z odpovědi na databázi. Z hlediska uživatelského rozhraní je třeba přidat pouze jedno autorizační tlačítko. V případě potřeby využití `access token` autorizačního serveru bude potřeba rozšířit databázi a vyřešit ukládání a obnovování `access token` a `refresh token`.

FIT01 Integrace s FIT Klasifikace Výsledné hodnocení projektů může být vhodné exportovat do služby FIT Klasifikace. V nejjednodušší verzi se jedná o zápis celkového počtu bodů projektu všem jeho členům registrovaným přes autorizační server ČVUT.

Závěr

Cílem této práce byla analýza stávajících způsobů vedení studentských projektů zaměřená na FIT ČVUT s následným návrhem a implementací OSS IS.

V souvislosti se stanoveným cílem první část této práce byla věnována rešerši některých dostupných systémů na FIT ČVUT pro správu projektů. S pomocí slovního popisu a konceptuálních modelů byly zanalyzovány jejich hlavní rysy, poskytovaná funkcionalita a silné a slabé stránky. Následně na základě získaných znalostí a v souladu se zadáním bakalářské práce byla sepsána specifikace nového informačního systému pro podporu aktivit kolem studijních projektů. Vzniklá specifikace postupně popisuje hlavní životní cyklus projektů a jaké funkční a obecné požadavky budou kladeny na nový systém. V samostatných podkapitolách je popsána uživatelská činnost, primární aktivity jsou navíc zpřesněny konceptuálními modely v jazyku UML.

V souladu s definovanou specifikací byl vytvořen obecný návrh OSS aplikace z hlediska vhodných technologií a architektury. Vzniklý systém se skládá ze dvou samostatných částí – serverové aplikace poskytující veřejně dostupné API a klientské webové aplikace přizpůsobené pro desktopové i mobilní rozhraní. Obě části jsou psány ve stejném skriptovacím jazyku JavaScript (případě TypeScript, který udržuje zpětnou kompatibilitu), to přináší přehlednost a jednotnost z hlediska využívaných knihoven.

Serverová část disponuje vlastním uživatelským systémem založeném na vnějších autorizačních OAuth 2.0 serverech. Spolu s implementovaným systémem práv tvoří jednoduché rozhraní pro kontrolu přístupu uživatelů k jednotlivým funkcionalitám. Jako úložiště jsou využívány databáze PostgreSQL (udržování systémových entit a jejich relací), MongoDB (uchovávání obsahu projektů) a Redis (předávání autorizačních klíčů). Komunikace s dostupnými klientskými aplikacemi se provádí přes RESTful API.

Implementace klienstkého webového rozhraní je založena na technologii SPA a SSR. Z větší části představuje pouze vizualizaci dat ze serveru, ale speciální částí se stal způsob generování obsahu projektu. Logika generování vizuální reprezentace dat byla přenesena na GitHub úložiště, odkud je v případě potřeby automaticky stahována jednotlivými uživateli přes službu jsDelivr.

Výběr všech technologií byl zdůvodněn v jednotlivých kapitolách práce. V případě nutnosti byly provedeny i některé srovnávací testy frameworků či implementačních řešení. K oběma aplikacím jsou dodávány testovací soubory a krátké dokumentace. V jedné z posledních kapitol je uveden seznam možných vylepšení informačního systému.

Nový informační systém je publikován na serveru GitHub, z důvodů nedostatku dat a zkušeností však není dokonalý. Hlavní cíl práce – vytvoření nového OSS nástroje pro správu studentských projektů – je splněn, sekundární cíl – integrace se službami FIT ČVUT – nebyl plně realizován. Projekt z důvodu nedostatku potřebné funkcionality není hotový pro nabídnutí využití na FIT ČVUT. Budoucí rozvoj a uplatnění služby budou dodatečně konzultovány s fakultou. V případě neúspěchu bude projekt nadále vyvíjen nezávisle.

Literatura

1. TÝM SWINPRO. *Swinpro - Swinpro - Vítejte* [online]. 2019 [cit. 2019-02-15]. Dostupné z: https://project.fit.cvut.cz/swinpro/.
2. TÝM SWINPRO. *Developers – SWINPRO - mlejnjir* [online]. 2019 [cit. 2019-02-15]. Dostupné z: https://project.fit.cvut.cz/trac/SWINPRO/wiki/Developers.
3. TÝM SWINPRO. *Changeset 1645 – SWINPRO - mlejnjir* [online]. 2019 [cit. 2019-02-15]. Dostupné z: https://project.fit.cvut.cz/trac/SWINPRO/changeset/1645.
4. TÝM SWINPRO. *Changeset 1907 – SWINPRO - mlejnjir* [online]. 2019 [cit. 2019-02-15]. Dostupné z: https://project.fit.cvut.cz/trac/SWINPRO/changeset/1907.
5. CENTRUM ZNALOSTNÍHO MANAGEMENTU. *Moodle FIT* [online]. 2019 [cit. 2019-02-15]. Dostupné z: http://moodle.fit.cvut.cz.
6. MOODLE. *Moodle - Open-source learning platform | Moodle.org* [online]. 2019 [cit. 2019-02-15]. Dostupné z: https://moodle.org.
7. CENTRUM ZNALOSTNÍHO MANAGEMENTU. *Centrum znalostního managementu FEL ČVUT* [online]. 2018 [cit. 2019-02-15]. Dostupné z: http://czm.fel.cvut.cz/cs/.
8. ČVUT FIT. *Autoři | BI-DBS* [online]. 2019 [cit. 2019-02-15]. Dostupné z: https://dbs.fit.cvut.cz/authors/.
9. ALMSAEED STUDIO. *AdminLTE Control Panel Template* [online]. 2019 [cit. 2019-04-06]. Dostupné z: https://adminlte.io/.
10. ČVUT FIT. *FIT ČVUT Course Pages* [online]. 2019 [cit. 2019-02-15]. Dostupné z: https://courses.fit.cvut.cz.

11. GITLAB, Inc. *GitLab.org / GitLab Community Edition · GitLab* [online]. 2019 [cit. 2019-02-15]. Dostupné z: https://gitlab.com/gitlab-org/gitlab-ce.
12. G2. *Best Version Control Systems* [online]. 2019 [cit. 2019-04-11]. Dostupné z: https://www.g2.com/categories/version-control-systems.
13. PRETTIER. *What is Prettier?* [online]. 2019 [cit. 2019-04-11]. Dostupné z: https://prettier.io/docs/en/index.html.
14. ZEIT. *Server Side Support for Clean URLs* [online]. 2019 [cit. 2019-04-19]. Dostupné z: https://nextjs.org/learn/basics/server-side-support-for-clean-urls/create-a-custom-server.
15. SARAVANAN, Nandhini. *Database Scaling: Horizontal and Vertical Scaling* [online]. 2019 [cit. 2019-04-20]. Dostupné z: https://hackernoon.com/database-scaling-horizontal-and-vertical-scaling-85edd2fd9944.
16. BUI, An. *PostgreSQL Vs. MySQL* [online]. 2018 [cit. 2019-04-20]. Dostupné z: https://blog.panoply.io/postgresql-vs.-mysql.
17. HOSSAIN, Monsur. *CORS in Action: Creating and consuming cross-origin APIs*. First edition. Manning Publications Co., 2015. ISBN 9781617291821.
18. SHARIAT, Jonathan; SAUCIER, Cynthia Savard. *Tragic Design*. First edition. O'Reilly Media, Inc., 2017. ISBN 9781491923610.
19. JONES, Bruce. *Color Theory: The Importance of Color in Web Design* [online]. 2014 [cit. 2019-04-19]. Dostupné z: https://www.designandpromote.com/color-theory-the-importance-of-color-in-web-design/.
20. ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE. *Grafický manuál identity ČVUT* [online]. 2019 [cit. 2019-04-19]. Dostupné z: https://www.cvut.cz/sites/default/files/content/e254fb38-e72d-463b-8c9f-cb0435416f29/cs/20161215-graficky-manual-identity-cvut-v-praze.pdf.
21. GUEST AUTHOR. *How to Use F & Z Patterns in Your Landing Page Design* [online]. 2018 [cit. 2019-04-19]. Dostupné z: https://www.wordstream.com/blog/ws/2018/07/31/f-z-patterns-landing-page-design.
22. STATCOUNTER. *Desktop vs Mobile vs Tablet vs Console Market Share Worldwide* [online]. 2019 [cit. 2019-04-07]. Dostupné z: http://gs.statcounter.com/platform-market-share#monthly-201801-201903.
23. FACEBOOK. *Lifting State Up* [online]. 2019 [cit. 2019-04-20]. Dostupné z: https://reactjs.org/docs/lifting-state-up.html.
24. ŠKORNIČKOVÁ, Eva. *Co je GDPR a jak bude aplikováno v Česku* [online]. 2019 [cit. 2019-04-19]. Dostupné z: https://www.gdpr.cz/gdpr/co-je-gdpr/.

25. SEGEŤA, Luboš. *Připravte svůj web na GDPR* [online]. 2018 [cit. 2019-04-19]. Dostupné z: https://proficio.cz/pripravte-svuj-web-na-gdpr.
26. KOUBA, Tomáš. *Jak na GDPR na webu – praktický návod* [online]. 2018 [cit. 2019-04-19]. Dostupné z: https://www.netmagnet.cz/blog/gdpr/.

Příloha **A**

Seznam použitých zkratek

API	application programming interface
BM	bundle manager
CDN	Content Delivery Network
CD	continuous delivery
CI	continuous integration
CLI	command-line interface
CORS	Cross-Origin Resource Sharing
CSR	client side rendering
CSS	Cascading Style Sheets
DBMS	Database Management System
DBS	Databázové systémy
DOM	Document Object Model
ERD	entity-relationship diagram
EU	Evropská unie
FAQ	frequently asked questions
FIT	Fakulta informačních technologií
GDPR	General Data Protection Regulation
GUI	graphical user interface
HTML	Hypertext Markup Language
HTTPS	Hypertext Transfer Protocol Secure
HTTP	Hypertext Transfer Protocol
ID	Identification Data
IS	informační systém
JSON	JavaScript Object Notation

JWT	json web token
LESS	Leaner Style Sheets
LTR	left-to-right
MIME	Multipurpose Internet Mail Extensions
NoSQL	Not Only SQL
ODM	object data modeling
ORM	object-relational mapping
OSS	open-source software
REST	Representational State Transfer
RTL	right-to-left
SCSS	Sassy CSS
SEO	search engine optimization
SPA	single-page application
SQL	Structured Query Language
SSR	server side rendering
UC	use case
UI	user interface
UML	Unified Modeling Language
URI	Uniform Resource Identifier
UX	user experience
VCS	version control system
ČVUT	České vysoké učení technické

Příloha **B**

Obsah přiloženého média

- `/`
  - `README.txt` ..... stručný popis obsahu média
  - `src`
    - `server` ..... zdrojový kód implementace serverové části
    - `interpreters` ..... zdrojový kód implementace interpretů
    - `client` ..... zdrojový kód implementace klientské části
  - `docs` ..... textové dokumenty
    - `thesis` ..... zdrojový kód a PDF soubor bakalářské práce
    - `docs-dev` ..... vývojářská dokumentace
    - `docs-user` ..... uživatelská dokumentace