VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

NÁSTROJ PRO SPRÁVU DOKUMENTŮ V MANAGEMENTU PROJEKTŮ

ELECTRONIC DOCUMENT MANAGEMENT IN PROJECT MANAGEMENT TOOL

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. OLGA GAVRYLIUK**
AUTHOR

VEDOUCÍ PRÁCE — **Doc. RNDr. JITKA KRESLÍKOVÁ, CSc.**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2018/2019

Zadání diplomové práce

Studentka: **Gavryliuk Olga, Bc.**
Program: Informační technologie Obor: Management a informační technologie
Název: **Nástroj pro správu dokumentů v managementu projektů**
Electronic Document Management in Project Management Tool
Kategorie: Softwarové inženýrství
Zadání:

1. Seznamte se se znalostními oblastmi managementu projektů dle aktuálního standardu PMI. Zaměřte se zejména na vstupy a výstupy (dokumenty) procesů znalostních oblastí Řízení kvality, Řízení lidských zdrojů a Řízení komunikace v rámci projektu.
2. Seznamte se s principy a modely elektronické správy dokumentů (EDM).
3. Specifikujte požadavky na systém pro správu dokumentů výše uvedených znalostních oblastí s použitím zvoleného modelu. Systém navrhněte s ohledem na rozšiřitelnost pro správu dokumentů ostatních znalostních oblastí.
4. Zvolte vhodné vývojové prostředí a implementujte prototyp funkcí navrženého systému vybraných po dohodě s vedoucí.
5. Na vzorku dat vybraném po dohodě s vedoucí ověřte funkčnost vytvořeného prototypu nástroje.
6. Zhodnoťte dosažené výsledky a diskutujte možnosti jeho dalšího rozšíření.

Literatura:

- *A Guide To The Project Management Body Of Knowledge:* Sixth Edition, Project Management Institute, 2017. ISBN 978-1-62825-184-5.
- *Agile Practice Guide*: global standard Project Management Institute, 2017. ISBN 978-1-62825-199-9.

Při obhajobě semestrální části projektu je požadováno:

- Splnění bodů 1 až 3 zadání.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Kreslíková Jitka, doc. RNDr., CSc.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 22. května 2019
Datum schválení: 30. října 2018

Abstrakt

Tato práce se zabývá systémy na elektronickou správu dokumentů (EDMS) z perspektivy vybraných znalostních oblastí procesů v projektovém řízení. Cílem této práce bylo na základě vhodně vybraného EDM modelu vytvořit vlastní EDM systém, který by napomáhal v managování dokumentů, které vznikají během procesů řízení ve vybraných oblastech projektového řízení (kvalita, lidské zdroje a komunikace v rámci projektu) s možností rozšíření na další znalostní oblasti.

Abstract

This work deals with electronic document management systems (EDMS) from the perspective of selected knowledge areas of project management processes. The aim of this thesis was to create an EDM system based on an appropriately selected EDM model, which would assist in the management of documents that arise during management processes in selected areas of project management (quality, human resources and communication within the project) with the possibility of extending to other knowledge areas.

Klíčová slova

Elektronická správa dokumentů, EDM, EDMS, DMS, databáze, relační model databáze, objektový model databáze, dokumentový model databáze, management projektů, dokument, PMI, PMBOK průvodce 6. vydání, řízení kvality, řízení lidských zdrojů, řízení komunikace v projektu, NoSQL, MongoDB, PostgreSQL, Quill, Yjs, TypeScript, Node.js, React, CRDT.

Keywords

Electronic Document Management, EDM, Electronic Document Management System, EDMS, Document Management System, DMS, database, relational database model, object database model, documental database model, project management, document, PMI, PMBOK guide 6. edition, quality management, human resources management, project management, communication management, NoSQL, MongoDB, PostgreSQL, Quill, Yjs, TypeScript, Node.js, React, CRDT.

Citace

GAVRYLIUK, Olga. *Nástroj pro správu dokumentů v managementu projektů.* Brno, 2019. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Doc. RNDr. Jitka Kreslíková, CSc.

Nástroj pro správu dokumentů v managementu projektů

Prohlášení

Prohlašuji, že jsem danou diplomovou práci vypracovala samostatně pod vedením paní Doc. RNDr. Jitky Kreslíkové, CSc. Uvedla jsem všechny literární prameny a publikace, ze kterých jsem čerpala.

.....
Olga Gavryliuk
21. května 2019

Poděkování

Touto cestou bych ráda vřele poděkovala vedoucí své diplomové práce doc. RNDr. Jitce Kreslíkové, CSc. především za trpělivost, pochopení, odborné rady a vedení této práce. Také děkuji své rodině a všem mým blízkým, kteří mi poskytovali podporu a motivaci během celého projektu.

Obsah

Kapitola 1

Úvod

V dávných časech měly informace cenu zlata v mnoha životních záležitostech, a úspěch přímo závisel na jejich (ne-)přítomnosti. Ještě tehdy informace hrály jednu z nejdůležitějších a klíčových rolí ve vedení mezinárodních vztahů, obecně v politice a řízení států, válečnictví, obchodě, atd.

Dané tvrzení bylo potvrzeno i tváří zakladatele obrovské bankovní dynastie Rothschilts, Nathana Rothschilde. Je autorem fráze, která se okamžitě stala populární a aktuální: „Kdo vlastní informace, vlastní celý svět.“ [15]. Tímto se chtěl autor dotknout problému významu informací v obchodě, respektive podnikání. Nathan Rothschild byl toho názoru, že ten, kdo se stal prvním vlastníkem informace, měl obrovskou výhodu oproti ostatním lidem, respektive konkurentům, a může ji použít k vlastním účelům, a tím dokázat vyřešit i ty největší úkoly či problémy [15].

Osobně s touto pravdivou frází souhlasím, jelikož si také myslím, že správná informace ve správný čas (obvykle to znamená čím dříve, tím lépe), je hlavním motorem pokroku a zárukou úspěchu v podnikání. Taková výhoda s velkou pravděpodobností může významně pomoci firmě v jejím rozvoji, a následně může vést k ekonomickému růstu a větší prosperitě. Proto, aby informace přinesla podniku co nejvíce výhod, je důležité ji umět správně použít: zabezpečit včasnou dostupnost, možnost rychlého a efektivního nalezení a sdílení informace v rámci firmy tak, aby se každý dostal k informacím, jež ke své práci potřebuje. V minulosti bylo udržovat tyto informační procesy v podniku poměrně náročné a vyžadovalo to neustálé papírování, a cena, která byla přímo závislá na množství zpracovávaných dokumentů, se stále navyšovala spolu s růstem podniku. Příchod moderních technologií (počítačů a online prostředí) otevřel nové možnosti pro zjednodušení správy informací nejen v rámci samotné firmy, ale také ve vztahu k veřejnosti. V závislosti na velikosti podniku, rozsahů jeho informačních toků, začaly vznikat jednoduché systémy pro správu a sdílení dokumentů, následně i celopodnikové informační systémy ošetřující veškeré informační procesy celé společnosti. První takové systémy se datují do 80. let 20. století, kdy velké společnosti začaly vyvíjet systémy na evidenci papírových dokumentů [10].

Tato práce bude zaměřena právě na systémy pro správu dokumentů Document Management System (DMS), zejména na jeden konkrétní, a to Electronic Document Management (systém) - EDM(s). V minulostí se tradiční DMS zajímaly o správu pouze papírových dokumentů, ale neustálý technologicky pokrok tuto situaci změnil. V současné době, stejně i v této práci, jsou tyto dva pojmy (DMS a EDM(s)) často zaměňované jako ekvivalentní, a není mezi nimi žádný významný rozdíl. EDM systém bude popsán v práci nejen obecně, ale i z pohledů zpracování dokumentů ve vybraných znalostních oblastech v rámci určitého vnitropodnikového procesu s použitím zvoleného EDM modelu.

Během práce nad libovolným projektem je správa dokumentů, neboli dokument management, úzce propojena s projektovým managementem, proto budou v první fázi čtenářům vysvětleny důležité pojmy z oblasti managementu projektů dle aktuálního standardu PMI. Následně se čtenář seznámí s pojmy z oblasti DMS, zjistí, k čemu jsou takové systémy dobré, a jaké jsou jejich hlavní funkce a výhody. V další částí práce budou specifikovány požadavky na systém pro správu dokumentů obecně a také z pohledu znalostních oblastí jako: řízení kvality, řízení lidských zdrojů a řízení komunikace. Také tato část bude obsahovat prvotní návrh budoucího systému pro správu dokumentů s ohledem na vybrané oblasti znalostí a možnosti jeho rozšíření na oblasti znalostí nezohledněné v této práci. Následně bude čtenáři představena kapitola „Implementace“, kde budou popsány použité technologie a zajímavosti z procesu realizace projektu. V poslední části práce bude popsán způsob testování vytvořeného produktu podle testovacích scénářů. S dosaženými výsledky bude možné se seznámit ve shrnutí uvedeném v kapitole „Závěr“.

Kapitola 2

Znalostní oblasti managementu projektů dle aktuálního standardu PMI

Základní standard projektového řízení, včetně podrobného popisu oblasti znalostí management projektů, procesů, nástrojů a technik podle PMI (Project Management Institute), je popsán v knize PMBOK Guide (Project management body of knowledge [34]), ze které bude vycházet celá tato kapitola. Na jejím základe budou níže vysvětleny základní pojmy projektového řízení a části standardu, na které se v tomto projektu zaměříme. Procesní pojetí problematiky projektového řízení je definováno pěti hlavními rodinami procesů, deseti oblastmi projektového řízení, jednotlivými procesy a jejich vzájemnými vazbami, kde veškeré procesy a procesní kroky mají definované své vstupy, výstupy a nástroje transformace (úkony, metody, techniky).

2.1 Základní pojmy z oblasti managementu projektů

Pro lepší orientaci a pochopení problematiky projektového řízení je nutné popsat její základní pojmy. Začneme pojmem projekt.

Projekt je dočasné úsilí o vytvoření jedinečného produktu, služby nebo výsledku. Projekty jsou realizovány tak, aby splnily cíle dosažením výsledků. **Cíl** je definován jako výsledek, na který mají být práce směrovány, strategická pozice, účel a výsledek dosažen, produkt/služba vyroben/provedena. **Produkt** je definován jako jakákoli jedinečná a ověřitelná produkce, výsledek nebo schopnost provádět službu, která je nutná pro vytvoření procesu, fáze nebo projektu. Výstupy mohou být hmatatelné nebo nehmotné.

Splnění projektových cílů může vygenerovat jeden nebo více následujících výstupů:

- Jedinečný produkt, který může být součástí jiné položky, vylepšení nebo oprava k položce, nebo nová koncová položka sama o sobě (např. oprava detekce v koncové položce);
- Jedinečná služba nebo schopnost provádět službu (např. obchodní funkce podporující výrobu nebo distribuci);
- Jedinečný výsledek, jako je výsledek nebo dokument (např. výzkumný projekt, který rozvíjí znalosti, jež mohou být použity k určení, zda existuje trend nebo nový proces, bude pro společnost přínosem);

- Jedinečná kombinace jednoho nebo více produktů, služeb nebo výsledků (např. softwarové aplikace, související dokumentace a služby helpdesku).

V definice pojmu projekt se střetáváme se slovem „dočasné úsilí“. Dočasná povaha projektů nemusí nutně znamenat, že projekt má krátké trvání, ale naopak indikuje, že projekt má určitý počátek a konec. Konec projektu se signalizuje splněním jedné nebo více z následujících skutečností:

- Cíle projektu byly úspěšně dosaženy;
- Cíle projektu nebudou nebo nemohou být splněny;
- Financování je vyčerpáno nebo není více dostupné pro přidělení k projektu;
- Již není potřeba daného projektu (např. zákazník nemíní projekt dokončovat, změny ve strategii a prioritách firmy ukončí projekt, organizační management směřuje projekt k ukončení, uzavření);
- Lidé nebo fyzické zdroje již nejsou k dispozici;
- Projekt je ukončen z důvodu legální příčiny nebo vhodnosti.

Pro úplnost je nutné se také seznámit s pojmem **proces**. Proces je série akcí nebo kroků podniknutých k dosažení určitého cíle. Každý proces je řízený děj, který má přesně stanoven začátek, konec, zadavatele a zákazníka. Procesů je mnoho, ale nás budou zajímat pouze ty, které ve své každodenní praxi používají tisíce firem, tedy procesy, které přeměňováním vstupů na výstupy generují hodnoty. Proces má také vlastnosti:

- Má definovaný vstup/dodavatele a výstup/zákazníka;
- Probíhá opakovaně a ve fázích;
- Lze jej dekomponovat – rozložit na podprocesy a aktivity;
- Vstupy a výstupy procesu jsou předvídatelné;
- Má lineární a logickou posloupnost;
- Je funkčně závislý na vnitřních procedurách a zdrojích.

2.1.1 Skupiny procesního řízení projektů

Skupina procesů řízení projektů je logické seskupení procesů řízení projektů pro dosažení konkrétních cílů projektu. Skupiny procesů jsou nezávislé na fázích projektu. Procesy řízení projektů jsou seskupeny do následujících pěti skupin procesů řízení projektů:

- **Zahajovací procesní skupina** (z angl. *Initiating Process Group*). Tyto procesy byly prováděny za účelem definování nového projektu nebo nové fáze stávajícího projektu získáním oprávnění k zahájení projektu nebo fáze.
- **Plánovací procesní skupina** (z angl. *Planning Process Group*). Tyto procesy se vyžadují pro stanovení rozsahu projektu, upřesnění cílů a definování postupu potřebného k dosažení cílů, pro které byl projekt realizován.

- **Provádění skupiny procesů** (z angl. *Executing Process Group*). Tyto procesy byly prováděny za účelem dokončení práce definované v plánu řízení projektu, aby byly splněny požadavky projektu.
- **Monitorovací a kontrolní procesní skupina** (z angl. *Monitoring and Controlling Process Group*). Procesy vyžadované pro sledování, přezkoumání a úpravu pokroku a výkonu projektu. Identifikace oblasti, ve kterých jsou požadovány změny plánu, a inicializace příslušné změny.
- **Ukončení skupiny procesů** (z angl. *Closing Process Group*). Tyto procesy byly prováděny k formálnímu dokončení nebo uzavření projektu, fáze nebo smlouvy.

2.2 Přehled znalostních oblastí managementu projektů

Projektové řízení není žádnou novinkou. Lidstvo začlenilo jeho použití do svého každodenního života již několik století nazpět: je zřejmé, že při plnění takových světoznámých „projektů“ jako: pyramidy v Gíze, velká čínská zeď, panamský průplav, přistání člověka na Měsíci, umístění mezinárodní vesmírné stanice na orbitu potřebovalo maximálně efektivní a rozumné řízení v každé fázi životního cyklu těchto projektů. Životní cyklus projektů se skládá ze série fází, které projekt protínají od začátku až po dokončení. Důležité je to, aby byl životní cyklus projektu dostatečně flexibilní a aby se dokázal vypořádat s různorodými faktory zahrnutými v projektu. Jeho flexibilita může být ovlivněna procesy fáze, kterou je nutné provést tak, aby se vývoj projektu dokončil nebo se posunul do další fáze. Z toho pozorujeme, že i velký projekt se „dekomponuje“ do malých částí, kterým říkáme proces, a že jejich efektivní a správné provedení je zárukou úspěšného a včasného dokončení jakéhokoliv díla.

V managementu projektu se procesy řízení projektu kategorizují podle oblasti znalostí (z angl. *Knowledge Areas*). Oblast znalostí je identifikovaná oblast řízení projektů, která je definovaná jejími znalostními požadavky a popsaná v rámci jejich dílčích procesů, postupů, vstupů, výstupů, nástrojů a technik.

Aktuální *PMBook Guide* uvádí 10 oblastí znalostí, které jsou vzájemně propojeny a jsou definovány odděleně od perspektivy řízení projektu. Dané oblasti znalostí se často používají ve většině projektů:

1. **Řízení integrace projektu** (z angl. *Project Integration Management*). Zahrnuje procesy a aktivity pro identifikaci, definování, kombinaci, sjednocení a koordinaci různých procesů a aktivit řízení projektů.
2. **Řízení rozsahu projektu** (z angl. *Project Scope Management*). Zahrnuje procesy potřebné k zajištění toho, aby projekt zahrnoval veškerou požadovanou práci a práci pro úspěšné dokončení projektu.
3. **Řízení plánu projektu** (z angl. *Project Schedule Management*). Zahrnuje procesy potřebné k řízení včasného dokončení ochrany.
4. **Řízení nákladů projektu** (z angl. *Project Cost Management*). Zahrnuje procesy spojené s plánováním, odhadováním, financováním a řízením nákladů tak, aby projekt mohl být dokončen v rámci schváleného rozpočtu.

5. **Řízení kvality projektu** (z angl. *Project Quality Management*). Zahrnuje procesy pro začlenění politiky kvality organizace týkající se plánování, správy a kontroly požadavků na kvalitu projektů a produktů s cílem splnit očekávání zúčastněných stran.

6. **Řízení zdrojů projektu** (z angl. *Project Resource Management*). Zahrnuje procesy pro identifikaci, získání a správu zdrojů potřebných pro úspěšné dokončení projektu.

7. **Řízení komunikace (v) projektu** (z angl. *Project Communications Management*). Zahrnuje procesy potřebné k zajištění včasného a vhodného plánování, sběru, tvorby, distribuce, ukládání, vyhledávání, řízení, kontroly, monitorování a konečné dispozice informací o projektu.

8. **Řízení rizik projektu** (z angl. *Project Risk Management*). Zahrnuje procesy provádění plánu řízení rizik, identifikace, analýzy, plánování reakcí, implementace reakcí a sledování rizik projektu.

9. **Řízení nákupní činnosti v projektu** (z angl. *Project Procurement Management*). Zahrnuje procesy potřebné k nákupu nebo nabytí produktů, služeb či výsledků potřebných mimo projektový tým.

10. **Řízení zúčastněných stran projektu** (z angl. *Project Stakeholder Management*). Zahrnuje procesy potřebné k identifikaci lidí, skupin nebo organizací, které by mohly ovlivnit projekt nebo být projektem ovlivněny, analyzovat očekávání zúčastněných stran a jejich dopad na projekt a rozvíjet vhodné strategie řízení pro efektivní zapojení zúčastněných stran do projektových rozhodnutí a provádění úkonů.

Potřeba konkrétního projektu může vyžadovat jednu nebo více dodatečných oblastí znalostí, např. určitý stavební projekt může vyžadovat řízení financí, řízení bezpečnosti a ochrany zdraví. Náš projekt však zajímají pouze tři oblasti znalostí: oblast řízení kvality, oblast řízení lidských zdrojů a oblast řízení komunikace v rámci projektu.

2.2.1 Oblast řízení kvality

Řízení kvality projektů zahrnuje procesy pro začlenění politiky týkající se plánování, správy a kontroly požadavků na kvalitu projektů a produktů s cílem splnit cíle zúčastněných stran. Řízení kvality projektů také podporuje nepřetržité aktivity zlepšování procesů, které jsou prováděny jménem vykonávajících organizací. Mezi procesy řízení kvality projektu patří:

Řízení kvality plánu (z angl. *Plan Quality Management*)

Proces určení požadavků na kvalitu nebo norem pro projekt a jeho výstupy a dokumentace toho, jak projekt prokáže soulad s požadavky. Klíčovým přínosem tohoto procesu je to, že poskytuje pokyny a směr, jak bude řízen tok kvality v průběhu celého projektu. Tento proces se provádí jednou nebo na předem definovaných bodech projektu. V tomto projektu nás zajímají především dokumenty, které slouží jako vstupy/výstupy pro daný proces. Níže si popíšeme několik příkladů projektových dokumentů, které mohou sloužit jako vstup do procesu řízení kvality plánu:

- Záznam předpokladů (z angl. *Assumption log*): obsahuje všechny předpoklady a omezení týkající se požadavků na kvalitu a dodržovaní standardů.;

- Dokumentace požadavků (z angl. *Requirements documentation*): zohledňuje požadavky, které by měl projekt a produkt dosáhnout pro splnění očekávání zúčastněných stran. Součásti dokumentace požadavků zahrnují požadavky na kvalitu projektu a produktu. Požadavky projektový tým používá pro naplánovaní způsobu, jakým bude v projektu implementována kontrola kvality;
- Matice požadavků na vysledovatelnost (z angl. *Requirements traceability matrix*): spojuje požadavky produktu s výsledky a zajišťuje provedení testování každého z požadavků z dokumentace požadavků. Tato matice poskytuje přehled testů potřebných k ověření požadavků;
- Registr rizik (z angl. *Risk register*): obsahuje informace o hrozbách a příležitostech, které mohou mít vliv na požadavky na kvalitu;
- Registr zúčastněných stran (z angl. *Stakeholder register*): pomáhá identifikovat zainteresované subjekty, které mají přímý zájem o kvalitu nebo její dopad, s důrazem na potřeby a očekávání zákazníků a sponzorů projektu.

Výsledkem či výstupem tohoto procesu mohou být následující zaktualizované a nové projektové dokumenty:

- Registr získaných znalostí (z angl. *Lessons learned register*): je aktualizován s informacemi o výzvách, které se vyskytly v procesu plánování kvality;
- Matice požadavků na vysledovatelnost (z angl. *Requirements traceability matrix*): tam, kde jsou požadavky na kvalitu specifikované tímto procesem, budou požadavky zaznamenány v matici sledovatelnosti;
- Registr rizik (z angl. *Risk register*): nová rizika, které byly identifikovány během provedení tohoto procesu, budou zaznamenávána do registru rizik a řízena pomocí procesů řízení rizik;
- Registr zúčastněných stran (z angl. *Stakeholder register*): pokud se v důsledku tohoto procesu shromáždí další informace o stávajících nebo nových zúčastněných stranách, zaznamená se to do registru zúčastněných stran;
- Plán řízení kvality (z angl. *Quality management plan*): je součástí plánu řízení projektu, který popisuje, jak budou uplatňovány platné politiky, postupy a pokyny pro dosažení cílů kvality. Popisuje činnosti a zdroje potřebné pro tým projektového managementu k dosažení cílů kvality stanovených pro projekt. Může zahrnovat cíle kvality projektu, nástroje kvality, které budou použity pro projekt, hlavní postupy relevantní pro projekt, jako je řešení neshod, postupů nápravných opatření a postupů průběžného zlepšování atd.;
- Metriky kvality (z angl. *Quality metrics*): specificky popisují projekt nebo atribut produktu a způsob ověření procesem kontroly kvality jeho dodržování. Některé příklady kvalitativních ukazatelů zahrnují procento úkolů dokončených včas, výkonnost nákladů měřenou CPI, míru selhání, počet zjištěných vad za den, celkový výpadek za měsíc, chyby zjištěné na řádku kódu, skóre spokojenosti zákazníků a procento pokrytých požadavků plánu zkoušek jako měřítko pokrytí testem;

- Aktualizace plánu projektového řízení (z angl. *Project management plan updates*): jakákoliv změna plánu řízení projektu prochází procesem řízení změn organizace prostřednictvím žádosti o změnu. Součásti, které mohou vyžadovat změnu plánu řízení projektu, zahrnují plán řízení rizik, základní linie rozsahu (scope baseline) a další.

Správa kvality (z angl. *Manage Quality*)

Proces překladu plánu řízení kvality do aktivit výkonné kvality, které zahrnují politiku kvality organizace do projektu. Klíčovým přínosem tohoto procesu je, že zvyšuje pravděpodobnost splnění cílů jakosti, stejně jako zjišťování neúčinných procesů a příčin špatné kvality. Správa kvality využívá data a výsledky procesu řízení kvality, aby odrážely celkový stav kvality projektu zúčastněným stranám. Tento proces běží po dobu celého projektu. Níže popíšeme několik příkladů projektových dokumentů, které mohou sloužit jako vstup do procesu správy kvality:

- Registr získaných znalostí (z angl. *Lessons learned register*): zkušenosti získané dříve v rámci projektu, pokud jde o řízení kvality, lze uplatnit na pozdější fáze projektu s cílem zlepšit kvalitu efektivnosti a účinnosti řízení kvality;
- Měření kontroly kvality (z angl. *Quality control measurements*): používá se k analýze a hodnocení kvality procesů a výstupů projektu v souladu s normami prováděcí organizace nebo předepsanými požadavky. Měření kvality může také porovnat procesy použité pro vytváření měření a ověření skutečných měření za účelem určení jejich úrovně správnosti;
- Metriky kvality (z angl. *Quality metrics*): jsou ověřovány jako součást procesu řízení kvality. Proces řízení kvality používá tyto metriky kvality jako základ pro vývoj testovacích scénářů pro projekt a jeho výstupy a jako základ pro iniciativy ke zlepšení;
- Report rizik: používá se v procesu řízení kvality pro identifikaci zdrojů celkových rizik projektu a nejdůležitějších faktorů celkového stavu rizik okolí, které mohou ovlivnit cíle kvality projektu;
- Plán řízení projektu (z angl. *Project management plan*): Komponenty plánu řízení projektu zahrnují i plán řízení kvality, který definuje přijatelnou úroveň kvality projektu a výrobku, a popisuje, jak zajistit tuto úroveň kvality ve svých výstupech a procesech, co dělat s nevyhovujícími produkty a jaká nápravná opatření je k tomu třeba provést.

Výsledkem či výstupem tohoto procesu mohou být následující zaktualizované a nové projektové dokumenty:

- Záznam problémů (z angl. *Issue log*): nové problémy vznikající v důsledku tohoto procesu jsou zaznamenávány do protokolu problémů;
- Registr získaných znalostí (z angl. *Lessons learned register*): je zaktualizován s informacemi o problémech, se kterými se podařilo setkat, jak by se jim dalo předejít, a také o přístupech, které dobře fungovaly pro řízení kvality;
- Registr rizik (z angl. *Risk register*): nová rizika zjištěná během tohoto procesu jsou zaznamenávána do registru rizik a řízena pomocí procesů řízení rizik;

- Zprávy o kvalitě (z angl. *Quality reports*): jsou grafické, číselné nebo kvalitativní. Poskytnuté informace mohou procesy a útvary využít k přijetí nápravných opatření za účelem dosažení očekávání kvality projektu. Informace uvedené ve zprávách o kvalitě mohou zahrnovat všechny otázky řízení kvality, které tým zvýšil; doporučení pro zlepšování procesů, projektů a produktů; doporučení k nápravným opatřením a shrnutí poznatků z procesu kontroly kvality;

- Testovací a hodnotící dokumenty (z angl. *Test and evaluation documents*): mohou být vytvořeny na základě potřeb odvětví a šablon organizace. Jedná se o vstupy do procesu kontroly kvality a používají se k hodnocení dosažení cílů kvality;

- Žádosti o změnu (z angl. *Change requests*): pokud během procesu řízení kvality dojde ke změnám, které ovlivní některou ze součástí plánu řízení projektu, projektových dokumentů nebo procesů řízení projektů nebo produktů, projektový manažer by měl předložit žádost o změnu a řídit se standardním procesem řízení integrovaných změn;

- Aktualizace plánu projektového řízení (z angl. *Project management plan updates*): jakákoliv změna plánu řízení projektu prochází procesem řízení změn organizace prostřednictvím žádosti o změnu. Součásti, které mohou vyžadovat změnu plánu řízení projektu, zahrnují plán řízení kvality, základní linii rozsahu, základní plán harmonogramu, základní linii nákladů a další.

Kontrola kvality (z angl. *Control Quality*)

Proces sledování a zaznamenávání výsledků řízení kvality za účelem posouzení výkonnosti a zajištění výstupů projektu, zda je úplný, správný a splňuje očekávání zákazníků. Klíčovým přínosem tohoto procesu je ověřování toho, zda výsledky projektu a práce splňují požadavky stanovené zúčastněnými stranami pro konečné přijetí. Kontrola kvality by měla být prováděna v průběhu celého projektu, aby se pomocí spolehlivých údajů formálně prokázalo, že se plní kritéria pro přijetí od sponzora a/nebo zákazníka. Dokumenty projektu, které lze považovat za vstupy do tohoto procesu, zahrnují, ale nejsou omezeny na:

- Plán řízení projektu (z angl. *Project management plan*): zahrnuje i plán řízení kvality, který definuje, jak má být v rámci projektu prováděna kontrola kvality;

- Registr získaných znalostí (z angl. *Lessons learned register*): poznatky získané dříve v projektu lze aplikovat na pozdější fáze projektu s cílem zlepšit kontrolu kvality;

- Metriky kvality (z angl. *Quality metrics*): specificky popisují atribut projektu nebo produktu a způsob, jakým proces kontroly kvality ověří jeho dodržování;

- Testovací a hodnotící dokumenty (z angl. *Test and evaluation documents*): jsou používány pro hodnocení dosažení cílů kvality.;

- Schválené žádosti na změnu (z angl. *Approved change requests*): aktualizace protokolu změn v rámci procesu Integrovaného řízení změn označuje, že některé změny jsou schváleny a některé ne. Schválené požadavky na změny mohou zahrnovat opravy vad, revidované pracovní metody a revidované plány. Implementace schválených změn by měla být ověřena, potvrzena pro úplnost, přezkoušena a ověřena jako správná;

- Dodávky (z angl. *Deliverables*): jsou jakékoli jedinečné a ověřitelné produkty, výsledky nebo schopnosti vykonávat službu, která musí být vytvořena pro dokončení procesu, fáze nebo projektu. Dodávky, které jsou výstupem z procesu Direct and Manage Project Work jsou kontrolovány a porovnávány s kritérii přijatelnosti definovanými v prohlášení o rozsahu projektu;
- Údaje o výkonnosti práce (z angl. *Work perfomance data*): obsahují údaje o stavu produktu, jako jsou pozorování, měření kvality a měření technické výkonnosti, informace o kvalitě projektu, o výkonnosti plánu a nákladové výkonnosti;
- Faktory podnikového prostředí (z angl. *Enterprise environmental factors*): jsou faktory životního prostředí podniku, které mohou ovlivnit proces řízení jakosti. K nim mimo jiné patří informační systém projektového řízení, nařízení legislativy a pravidla, standardy a pokyny specifické pro oblast aplikace.

Výsledkem či výstupem tohoto procesu mohou být následující zaktualizované a nové projektové dokumenty:

- Záznam problémů (z angl. *Issue log*): jestliže doručený dokument není slučitelný s požadavky na kvalitu, dokumentuje se jako problém;
- Registr získaných znalostí (z angl. *Lessons learned register*): získaný seznam zpracovaných zkušeností se aktualizuje s informacemi o zdrojích kvalitativních vad, a také o tom, jak by se jim dalo předejít, stejně tak i o dobře fungujícím přístupu;
- Registr rizik (z angl. *Risk register*): Nová rizika zjištěná během tohoto procesu jsou zaznamenána do registru rizik a řízena pomocí procesů řízení rizik;
- Testovací a hodnotící dokumenty (z angl. *Test and evaluation documents*): Testovací a vyhodnocovací dokumenty mohou být v důsledku tohoto procesu upraveny pro větší efektivnost budoucích testů;
- Měření kontroly kvality (z angl. *Quality control measurements*): jsou zdokumentované výsledky kontrolních činností. Měly by být zachyceny ve formátu specifikovaném v plánu řízení kvality;
- Ověřené dodávky (z angl. *Verified deliverables*): výstupy procesu kontroly kvality jsou ověřenými dodávkami informací, které se stávají vstupem do procesu validace rozsahu pro formalizované přijetí;
- Informace o pracovním výkonu (z angl. *Work performance information*): zahrnují informace o plnění požadavků projektu, příčinách odmítnutí, požadavcích na přepracování, doporučeních na nápravná opatření, seznamech ověřených výstupů, stavu metrik kvality a potřebě úprav procesů;
- Žádosti o změnu (z angl. *Change requests*): pokud během procesu kontroly kvality dojde ke změnám, které ovlivní některou ze součástí plánu řízení projektu, projektových dokumentů atd., projektový manažer by měl předložit žádost o změnu a řídit se standardním procesem řízení integrovaných změn;
- Aktualizace plánu projektového řízení (z angl. *Project management plan updates*): jakákoliv změna plánu řízení projektu prochází procesem řízení změn organizace prostřednictvím žádosti o změnu. Součásti, které mohou vyžadovat změnu plánu řízení projektu, zahrnují plán řízení kvality a další.

2.2.2 Oblast řízení lidských zdrojů

Řízení projektových prostředků zahrnuje procesy pro identifikaci, získání a správu zdrojů potřebných pro úspěšné dokončení projektu. Tyto procesy pomáhají zajistit správné zdroje pro projektového manažera a projektový tým ve správném čase a místě.

Existuje šest podoblastí procesů řízení projektových zdrojů, nás však zajímají pouze ty, které se týkají řízení lidských zdrojů, proto si zde popíšeme všechny oblasti a podrobněji rozebereme oblasti rozvoje a správy týmu:

Management plánovaní zdrojů (z angl. *Plan Resource Management*)

Proces definování odhadu, získání, řízení a využívání fyzických a týmových zdrojů.

Zdroje pro odhad aktivity (z angl. *Estimate Activity Resources*)

Proces odhadu týmových zdrojů, druhu a množství materiálu, zařízení a zdrojů potřebných pro práci s projektem.

Získávání zdrojů (z angl. *Acquire Resources*)

Proces získávání členů týmu, zařízení, vybavení, materiálů, zásob a dalších zdrojů potřebných k dokončení projektové práce.

Rozvoj týmu (z angl. *Develop Team*)

Proces zlepšování kompetencí, interakcí členů týmu a celkového prostředí týmu ke zlepšení výkonnosti projektu. Klíčovým přínosem tohoto procesu je zlepšení týmové práce, zlepšení interpersonálních dovedností a kompetencí, motivování zaměstnanců, snížení opotřebení a zlepšení celkového výkonu. Tento proces se provádí během celého projektu. Týmová práce je kritickým faktorem úspěchu projektu a rozvoj efektivních projektových týmů je jednou z hlavních povinností projektového manažera. Vysokého výkonu týmu lze dosáhnout využitím otevřené a efektivní komunikace, vytvářením příležitostí pro vybudovaní a upevnění týmu, rozvíjením důvěry mezi jeho členy, konstruktivním řízením konfliktu, podporou spolupráce při řešení problémů atd. Níže se nachází několik příkladů projektových dokumentů, které mohou sloužit jako vstup do procesu vývoje týmu:

- Registr získaných znalostí (z angl. *Lessons learned register*): poznatky získané dříve v rámci projektu s ohledem na rozvoj týmu lze aplikovat na pozdější fáze projektu s cílem zlepšit výkon týmu;
- Harmonogram projektu (z angl. *Project shedule*): definuje, jak a kdy poskytovat školení projektovému týmu, rozvíjet kompetence požadované v různých fázích. Identifikuje potřebu strategií rozvoje týmu založených na změnách během plnění projektu, pokud vůbec nastanou;
- Projektové týmové úkoly (z angl. *Project team assignments*): určují role a povinnosti týmu a jeho členů;
- Kalendáře zdrojů (z angl. *Resource calendars*): identifikují časy, kdy se členové projektového týmu mohou podílet na aktivitách rozvoje týmu. Pomáhá také ilustrovat dostupnost týmu během celého projektu;

- Týmová listina (z angl. *Team charter*): je tam, kde jsou dokumentovány provozní pokyny týmu. Hodnoty týmu a provozní pokyny poskytují strukturu, která popisuje, jak bude tým spolupracovat;
- Plán řízení projektu (z angl. *Project management plan*): zahrnuje, ale není omezen na plán řízení zdrojů. Udává vodítko pro poskytování odměn členů projektového týmu, zpětné vazby, dodatečných školení a disciplinárních opatření v důsledku hodnocení výkonnosti týmu a dalších forem řízení projektového týmu. Plán řízení zdrojů může zahrnovat i kritéria pro hodnocení výkonnosti týmu;
- Faktory podnikového prostředí (z angl. *Enterprise environmental factors*): jsou faktory životního prostředí podniku, které mohou ovlivnit proces vývoje týmu. Zahrnuje mimo jiné politiku řízení lidských zdrojů (najímání a ukončení pracovního poměru), kompetence a odborné znalosti členů týmu, jejich geografické rozložení.

Dokumenty projektu, které mohou být aktualizovány v důsledku provedení tohoto procesu, zahrnují mimo jiné:

- Registr získaných znalostí (z angl. *Lessons learned register*): je aktualizován informacemi o zjištěných výzvách a způsobech, jimž se bylo možné vyhnout, jakož i přístupy, které dobře fungovaly pro rozvoj týmu;
- Harmonogram projektu (z angl. *Project shedule*): činnosti pro rozvoj projektového týmu mohou mít za následek změny harmonogramu projektu;
- Projektové týmové úkoly (z angl. *Project team assignments*): jestliže vývoj týmu povede ke změnám dohodnutých na základě zadání, jsou tyto změny zaznamenány v dokumentaci k zadání projektového týmu;
- Kalendáře zdrojů (z angl. *Resource calendars*): jsou aktualizovány tak, aby odrážely dostupnost zdrojů pro projekt;
- Týmová listina (z angl. *Team charter*): může být aktualizována tak, aby odrážela změny dohodnutých pokynů pro provoz týmu, které vyplývají z vývoje týmu;
- Posouzení výkonnosti týmu (z angl. *Team performance assessments*): tým projektového managementu provádí průběžné nebo neformální hodnocení výkonnosti projektového týmu. Napomáhá to ke zlepšení dovedností, kompetencí, míry fluktuace v týmu;
- Žádosti o změnu (z angl. *Change requests*): pokud během procesu správy týmu dojde ke změnám, které ovlivní některou ze součástí plánu řízení projektu, projektových dokumentů atd., projektový manažer by měl předložit žádost o změnu a řídit se standardním procesem řízení integrovaných změn;
- Aktualizace plánu projektového řízení (z angl. *Project management plan updates*): jakákoliv změna plánu řízení projektu prochází procesem řízení změn organizace prostřednictvím žádosti o změnu. Také je možné, že změny bude potřebné provést i v plánu řízení zdrojů projektu;
- Aktualizace faktorů podnikového prostředí (z angl. *Enterprise environmental factors*): faktory, které jsou aktualizovány v důsledku procesu vývoje týmu, zahrnují mimo jiné záznamy o plánu rozvoje zaměstnanců a posouzení dovedností.

Správa týmu (z angl. *Manage Team*)

Správa týmu je proces sledování výkonu týmu, poskytování zpětné vazby, řešení problémů a řízení změn týmu s cílem optimalizovat výkonnost projektu. Daný proces poskytuje vodítko k poskytováni zpětné vazby, odměn, dalšímu školení nebo vyvození disciplinárního konání s členem týmu, jako například výsledek výkonu nebo jiných forem týmového managementu. Níže se nachází několik příkladů projektových dokumentů, které mohou sloužit jako vstup do procesu správy týmu:

- Záznam problémů (z angl. *Issue log*): používá se k dokumentování a sledování toho, kdo je zodpovědný za řešení konkrétních úkolů do cílového data;
- Registr získaných znalostí (z angl. *Lessons learned register*): jsou poznatky získané dříve v rámci projektu týkající se rozvoje týmu a mohou být použity pro pozdější fáze projektu s cílem zlepšit výkonnost týmu;
- Projektové týmové úkoly (z angl. *Project team assignments*): přiřazují projektovému týmu totožnost a role a odpovědnosti jeho členům;
- Týmová listina (z angl. *Team charter*): je dokument kde jsou zdokumentovány pokyny pro tým. Hodnoty týmu a pokyny této listiny poskytují provozní strukturu, která popisuje, jak bude tým společně pracovat;
- Plán řízení projektu (z angl. *Project management plan*): zahrnuje i plán řízení zdrojů, který definuje, jak by měly být prostředky projektového týmu řízeny a nakonec uvolněny;
- Zprávy o výkonu práce (z angl. *Work perfomance reports*): jsou fyzickou nebo elektronickou reprezentací informací o výkonu práce, jejichž účelem je vytvářet rozhodnutí, akce nebo povědomí. Zahrnují výsledky kontroly plánu, nákladů, kvality a ověřování rozsahu. Napomáhají při prognózovaní budoucích požadavků na zdroje týmu;
- Posouzení výkonnosti týmu (z angl. *Team performance assessments*): tým projektového managementu provádí průběžné nebo neformální hodnocení výkonnosti projektového týmu. Napomáhá to k vyřešení problémů a ke zlepšení interakce týmu;
- Faktory podnikového prostředí (z angl. *Enterprise environmental factors*): jsou faktory životního prostředí podniku, které mohou ovlivnit proces řízení týmu. Zahrnují i politiku řízení lidských zdrojů.

Výsledkem či výstupem tohoto procesu mohou být následující nové a zaktualizované projektové dokumenty:

- Záznam problémů (z angl. *Issue log*): nové problémy vzniklé v důsledku tohoto procesu jsou zaznamenány do speciálního záznamu;
- Registr získaných znalostí (z angl. *Lessons learned register*): seznam získaných poznatků je aktualizován v důsledku provedení daného procesu;
- Projektové týmové úkoly (z angl. *Project team assignments*): při vývoji týmů dochází ke změnám dohodnutých úkolů, tyto změny jsou zaznamenány v tomto dokumentu;

- Žádosti o změnu (z angl. *Change requests*): pokud během procesu správy týmu dojde ke změnám, které ovlivní některou ze součástí plánu řízení projektu, projektových dokumentů atd., projektový manažer by měl předložit žádost o změnu a řídit se standardním procesem řízení integrovaných změn;

- Aktualizace plánu projektového řízení (z angl. *Project management plan updates*): jakákoliv změna plánu řízení projektu prochází procesem řízení změn organizace prostřednictvím žádosti o změnu. Také je možné, že změny bude potřebné provést i v plánech řízení zdrojů, nákladů a harmonogramu projektu;

- Aktualizace faktorů podnikového prostředí (z angl. *Enterprise environmental factors*): faktory, které jsou aktualizovány v důsledku procesu správy týmu.

Kontrolní zdroje (z angl. *Control Resources*)

Proces zajištění s cílem, aby byly fyzické zdroje přiděleny a alokovány k projektu, aby byly k dispozici podle plánu, stejně jako sledování plánovaného vs. skutečného využití zdrojů. Podle potřeby se provádí nápravná opatření.

2.2.3 Oblast řízení komunikace v rámci projektu

Řízení komunikace zahrnuje procesy potřebné k zajištění toho, aby byly splněny informační potřeby projektu a jeho zúčastněných stran prostřednictvím rozvoje artefaktů a realizací aktivit určených k efektivní výměně informací. Řízení komunikace projektu se skládá ze dvou částí. První část zajišťuje, aby byla komunikace pro zúčastněné strany efektivní. Druhá část provádí činnosti nezbytné pro realizaci komunikační strategie.

Mezi procesy řízení komunikace projektu patří:

Plánování správy komunikací (z angl. *Plan Communications Management*)

Je proces rozvoje vhodného přístupu a plánu pro komunikaci v rámci projektu, založený na informačních potřebách všech zúčastněných stran nebo skupin, dostupných organizačních prostředcích a potřeb projektu. Klíčovým přínosem tohoto procesu je zdokumentovaný přístup k efektivnímu a účinnému zapojení zúčastněných stran včasným předkládáním relevantních informací. Výsledky procesu plánování komunikací by měly být pravidelně přezkoumávány v průběhu projektu a revidovány podle potřeb, aby byla zajištěna trvalá použitelnost. Tento proces je prováděn periodicky v průběhu projektu dle potřeby. Dokumenty projektu, které lze považovat za vstupy do tohoto procesu, zahrnují, ale nejsou omezeny na:

- Dokumentace požadavků (z angl. *Requirements documentation*): může zahrnovat komunikaci zúčastněných stran projektu;

- Registr zúčastněných stran (z angl. *Stakeholder register*): se používá k plánování komunikačních činností se zúčastněnými stranami;

- Projektová listina (z angl. *Project charter*): identifikuje klíčový seznam zúčastněných stran. Může také obsahovat informace o rolích a povinnostech zúčastněných stran;

- Plán řízení projektu (z angl. *Project management plan*): zahrnuje, například, plán zapojení zúčastněných stran, plán řízení zdrojů, který poskytuje návod, jak lze kategorizovat, přidělovat, spravovat a zveřejňovat zdroje týmu. Členové týmu a skupiny

mohou mít komunikační požadavky, které by měly být uvedeny v plánu řízení komunikace;

- Faktory podnikového prostředí (z angl.*Enterprise environmental factors*): faktory, které mohou ovlivnit proces řízení komunikace plánu (organizační kultura, politické prostředí, trendy, praxe nebo zvyky, geografické rozložení zařízení a zdrojů).

Výsledkem či výstupem tohoto procesu mohou být následující zaktualizované a nové projektové dokumenty:

- Harmonogram projektu (z angl. *Project schedule*): může být aktualizován, aby odrážel komunikační aktivity;
- Registr zúčastněných stran (z angl. *Stakeholder register*): může být aktualizován tak, aby odrážel plánované komunikace;
- Plán řízení komunikace (z angl. *Communications management plan*): je součástí plánu řízení projektu, který popisuje, jak bude projekt plánován, strukturován, implementován a sledován z hlediska účinnosti. Plán obsahuje informace o požadavcích na komunikace zúčastněných stran, informacích, které mají být sděleny, včetně jazyka, formátu, obsahu a úrovně podrobností, časový rámec a četnost šíření, atd. Plán řízení komunikace může obsahovat pokyny a šablony pro schůzky o stavu projektu, schůzky projektových týmů, e-schůzky a e-mailové zprávy;
- Aktualizace plánu projektového řízení (z angl. *Project management plan updates*): jakákoliv změna plánu řízení projektu prochází procesem řízení změn organizace prostřednictvím žádosti o změnu. Součásti, které mohou vyžadovat žádost o změnu plánu řízení projektu, zahrnují, ale nejsou omezeny na plán zapojení zúčastněných stran;

Správa komunikace (z angl. *Manage Communications*)

Je proces zajišťování včasného a vhodného sběru, tvorby, distribuce, ukládání, vyhledávání, správy, sledování a konečné dispozice projektových informací. Klíčovým přínosem tohoto procesu je, že umožňuje efektivní a výkonný tok mezi projektovým týmem a zúčastněnými stranami. Správa komunikace by měla umožnit flexibilitu v komunikačních činnostech, což umožní přizpůsobení metod a technik a přizpůsobení měnícím se potřebám projektu zúčastněných stran. Dokumenty projektu, které lze považovat za vstupy do tohoto procesu, zahrnují, ale nejsou omezeny na:

- Záznam změn (z angl. *Change log*): se používá ke sdělení změn, odrazování a odmítnutí požadavků na změnu u ovlivněných zúčastněných stran;
- Záznam problémů (z angl. *Issue log*): jeho prostřednictvím jsou sděleny informace o problémech zúčastněným stranám;
- Registr získaných znalostí (z angl. *Lessons learned registe*): poznatky získané dříve v projektu s řízením komunikace lze aplikovat na pozdější fáze projektu, aby se zvýšila efektivita a komunikace a komunikační proces;
- Zpráva o kvalitě (z angl. *Quality report*): zahrnuje otázky kvality a vylepšení produktu a vylepšení procesu. Tyto informace jsou předány těm, kteří mohou aplikovat nápravná opatření za účelem dosažení očekávání kvality projektu;

- Zpráva o rizicích (z angl. *Risk report*): uvádí informace o zdrojích celkového rizika spolu se souhrnnými informacemi o identifikovaných rizicích jednotlivých projektů. Tyto informace jsou sděleny majitelům rizik a dalším zainteresovaným stranám;
- Registr zúčastněných stran (z angl. *Stakeholder register*): identifikuje jednotlivce nebo organizace, které budou potřebovat různé druhy informací;
- Plán řízení projektu (z angl. *Project management plan*): Komponenty plánu řízení projektů zahrnují mimo jiné: plán řízení zdrojů, plán řízení komunikace a plán zapojení zúčastněných stran;
- Zprávy o výkonu práce (z angl. *Work perfomance reports*): Příklady zpráv o výkonu práce zahrnují reporty a zprávy o pokroku. Mohou obsahovat grafy a informace získané v hodnotách, trendech a prognózách, záložní grafy rezerv, histogramy defektů, informace o výkonu zakázky a souhrny rizik;
- Faktory podnikového prostředí (z angl.*Enterprise environmental factors*): faktory, které mohou ovlivnit tento proces, jsou mimo jiné organizační kultura, politické prostředí, trendy, praxe nebo zvyky, geografické rozložení zařízení a zdrojů.

Výsledkem či výstupem tohoto procesu mohou být následující zaktualizované a nové projektové dokumenty:

- Záznam problémů (z angl. *Issue log*): je aktualizován tak, aby odrážel jakékoli problémy s komunikací v projektu nebo jak byla jakákoli komunikace použita k ovlivnění aktivních problémů;
- Registr získaných znalostí (z angl. *Lessons learned registe*): je aktualizován informacemi o zjištěných výzvách a způsobech, jimž bylo možné se vyhnout, stejně jako přístupy, které fungovaly dobře, a co nefungovalo dobře pro řízení komunikace;
- Harmonogram projektu (z angl.*Project schedule*): může být aktualizován, aby odrážel stav komunikačních aktivit;
- Registr rizik (z angl. *Risk register*): je aktualizován tak, aby zachytil rizika spojená s řízením komunikace;
- Registr zúčastněných stran (z angl. *Stakeholder register*): může být aktualizován tak, aby obsahoval informace o komunikačních aktivitách se zúčastněnými stranami projektu;
- Projektová komunikace (z angl. *Project communications*): položky projektových komunikací mohou zahrnovat, ale nejsou omezeny na: zprávy, stav dodávky, průběh plánu, vynaložené náklady, prezentace a další informace požadované zúčastněnými stranami;
- Aktualizace plánu projektového řízení (z angl. *Project management plan updates*): jakákoliv změna plánu řízení projektu prochází procesem řízení změn organizace prostřednictvím žádosti o změnu. Součásti, které mohou vyžadovat žádost o změnu plánu řízení projektu, zahrnují, ale nejsou omezeny na plán zapojení zúčastněných stran, plán řízení komunikace atd.

Monitorování komunikace (z angl. *Monitor Communications*)

Provádí se proces zajišťování informačních potřeb projektu a jeho zúčastněných stran. Klíčovou výhodou tohoto procesu je optimální tok informací, jak je definován v plánu řízení komunikací plánu zapojení zúčastněných stran. Tento proces se provádí během celého projektu. Může vyžadovat různé metody, jako jsou průzkumy spokojenosti zákazníků, shromažďování získaných poznatků, pozorování týmu, přezkoumávání údajů ze záznamu problémů nebo vyhodnocování změn v matici pro hodnocení zapojení zúčastněných stran. Projektové dokumenty, které lze považovat za vstupy pro tento proces, zahrnují mimo jiné:

- Záznam problémů (z angl. *Issue log*): poskytuje historii projektu, záznam o problematice zapojení zúčastněných stran a způsob jejich řešení;
- Registr získaných znalostí (z angl. *Lessons learned registe*): poznatky získané dříve v projektu lze aplikovat na pozdější fáze projektu, aby se zvýšila efektivita komunikace;
- Projektová komunikace (z angl. *Project communications*): poskytuje informace o komunikacích, které byly distribuovány;
- Plán řízení projektu (z angl. *Project management plan*): Komponenty plánu řízení projektů zahrnují mimo jiné: plán řízení zdrojů, plán řízení komunikace a plán zapojení zúčastněných stran;
- Údaje o pracovní výkonnosti (z angl. *Work perfomance data*): obsahují údaje o typech a množstvích skutečně distribuovaných komunikací;
- Faktory podnikového prostředí (z angl.*Enterprise environmental factors*): faktory, které mohou ovlivnit tento proces jsou mimo jiné organizační kultura, politické prostředí, trendy, praxe nebo zvyky, geografické rozložení zařízení a zdrojů.

Výstupní projektové dokumenty, které mohou být vytvořeny nebo zaktualizovány v důsledku provádění tohoto procesu, zahrnují mimo jiné:

- Záznam problémů (z angl. *Issue log*): může být aktualizován o nové informace nastalých problémech, jejich průběhu a rozlišení;
- Registr získaných znalostí (z angl. *Lessons learned registe*): může být aktualizován o příčiny problémů, důvody vybraných nápravných opatření a případně o další komunikační registr znalostí;
- Registr zúčastněných stran (z angl. *Stakeholder register*): může být aktualizován o revidované požadavky na komunikaci se zúčastněnými stranami;
- Informace o pracovním výkonu (z angl. *Work performance information*): zahrnují informace o tom, jak probíhá komunikace v projektu, a to tak, že se vypořádají s úkony, které byly provedeny ve srovnání s plánovanými. Zvažuje zpětnou vazbu o komunikacích, jako jsou výsledky průzkumu o účinnosti komunikace;
- Žádosti o změnu (z angl. *Change requests*): proces sledování komunikace často vede k potřebě úpravy, akce a zásahu do komunikačních činností definovaných v plánu řízení komunikace. Požadavky na změnu jsou prováděny prostřednictvím procesu integrované kontroly. Výsledkem těchto požadavků na změnu může být například revize požadavků na komunikaci se zúčastněnými stranami, obsahu nebo nových postupů pro odstranění úzkých míst;

- Aktualizace plánu projektového řízení (z angl. *Project management plan updates*): jakákoliv změna plánu řízení projektu prochází procesem řízení změn organizace prostřednictvím žádosti o změnu. Součásti, které mohou vyžadovat změnu plánu řízení projektu, jsou například plány řízení komunikace, plány zapojení zúčastněných stran.

Kapitola 3

Elektronická správa dokumentů

Jak je vidět z názvu kapitoly, v této části práce se bude pojednávat o správě elektronických dokumentů a nástrojích (systémech), které to umožňují. Cílem kapitoly je pomoci čtenáři pochopit: co se považuje za dokument, co je to Správa elektronických dokumentů, k čemu a v jakých případech se používá a jaké výhody je schopna přinést podniku i okolnímu světu.

3.1 Definice pojmu dokument

V dávných časech se lidé dorozumívali spolu a předávali jeden druhému informace pomocí řeči a jazyka. Dlouhou dobu se nové poznatky, zážitky, důležité sdělení a pocity předávaly z pokolení na pokolení formou legend, příběhů či různých písní. To ale nebylo úplně dostačující a téměř od začátku lidé cítili potřebu si danou informaci koncentrovat, zaznamenávat a uchovávat pro budoucí generace pomocí znaků/písma. Také záznamy měly různou formu v závislosti na geografické oblasti (papyrové svitky, pergamen, hliněné destičky, hedvábí atd.) až nakonec se lidé naučili vyrábět papír, který postupem času umožnil vytvářet fyzické dokumenty, jak je známe v dnešní době [24].

První, kdo definoval a následně použil pojem dokument, byl Paul Otlet. Podle jeho definice je dokumentem hmotný nosič určitých rozměrů, formátu nebo svitku, na kterém jsou zaznamenány znaky reprezentující intelektuální činnost [45]. Současná definice daného pojmu se razantně neliší od původní a vypadá následovně: dokumentem je každá písemná, obrazová, zvuková nebo jiná zaznamenaná informace v analogové či digitální podobě [48].

V praxi se často setkáváme s tím, že pojem analogový dokument se často omezuje pouze na papírový nebo listinný, což ale není přesné, hmatatelný dokument může mít různou fyzickou podobu, jak bylo uvedeno výše. V této práci se budeme zabývat listinnou a zejména elektronickou podobou dokumentů. Z různých právních předpisů, definic a procesů v praxi, kde se dokumenty používají, vyplývá následující souhrn významných atributů dokumentu [40]:

- **Informační hodnota** – dokument je nositelem hodnotné informace.
- **Stálost** – dokument je stálý a neměnný.
- **Jazyk** – dokument je obvykle vyjádřen v určitém jazyce.
- **Ucelenost** – dokument je jednotka či celek, který se dělí pouze dočasně v rámci procesů zpracování.

- **Strukturovatelnost** – dokument je vždy strukturován podle jeho určení, povahy, podmínkách vzniku atd.[40]

3.1.1 Elektronický versus analogový dokument – vlastnosti

Elektronický dokument se výrazně liší od analogového (listiny, žádosti, smlouvy nebo také obrazy, fotografie apod.) svými specifickými vlastnostmi, jednou z nich je například nepřítomnost vazby s materiálním nosičem, která je nezbytnou v případě klasického dokumentu a tedy elektronický dokument nemá jedinečný originál. Srovnání vybraných vlastností dvou podob dokumentů jsou rozepsány níže [40].

Tvorba

Analogový či listinný dokument lze přímo napsat na hmotný nosič, ale elektronický dokument se vždy vytváří pomocí technických nástrojů, dokonce i v případě, že se jedná pouze o elektronickou kopii papírového dokumentu, tak ho předem musíme naskenovat, tedy použít určitý technický nástroj k jeho převodu.

Originál a přístup k ochraně

Listinný dokument má vždy jeden jediný originál, který v případě potřeby zachování předpokládá ochranu „svého“ nosiče. V případě zničení originálu již jiný nebude. Elektronický dokument má naopak nekonečně mnoho originálů, který(-é) za potřeby zachování předpokládá ochranu dat. V případě zničení originálu není nikdy známo, zda tento konkrétní originál byl poslední nebo ještě existují další stejnopisy.

Čtení

Čtení listinných dokumentů se provádí pomocí vlastních smyslů. Elektronické dokumenty se dají číst jen zprostředkovaně pomocí technických nástrojů.

Ochranné prvky

Listinné dokumenty se obvykle chrání pomocí ochranných prvků jako vlastnoruční či úředně ověřený podpis, pečeť, razítko, speciální ochranné prvky, například jaké se používají pro ochranu bankovek. Pro ochranu elektronických dokumentů se používají prvky jako: uznávaný elektronický podpis či kvalifikovaná značka, elektronické časové razítko atd.

3.1.2 Elektronický versus analogový dokument – prováděné činnosti

Analogové dokumenty většina lidí stále považuje za uv„lepší“, a to kvůli tomu, že jsou na ně zvyklí nebo kvůli jiným důvodům. Analogové dokumenty nejsou o nic více skutečné než dokumenty elektronické. I elektronický dokument může být důvěryhodný a disponovat stejně průkaznými informacemi jako analogové dokumenty [40]. Pro zdůraznění výše napsaného je uvedeno další porovnání těchto dvou typu dokumentů podle vybraných činností [39].

- Přístup oprávněných uživatelů: listinný dokument doputuje k uživatelům se zpožděním, zatímco elektronický dokument je pro ně přístupný okamžitě.

- Obecný přístup k dokumentu: listinný dokument vidí pouze ten, kdo ho má v daném okamžiku u sebe. K elektronickému dokumentu je možný okamžitý přístup téměř odkudkoliv pro jakoukoliv oprávněnou osobu.
- Sledování pohybu/manipulace s dokumentem: je téměř nemožné uhlídat pohyb a pořízení kopií listinného dokumentu. K manipulaci s elektronickými dokumenty nemají přístup neoprávnění uživatelé.
- Zabezpečení dokumentů: papírové dokumenty není možné stoprocentně zabezpečit, také se při přenášení mohou omylem ztratit nebo zničit, někdo je může účelně nebo náhodou odcizit, například spolu s taškou. Elektronické dokumenty jsou chráněny před podobnými negativními vlivy ze strany neoprávněných osob přístupovým právem.
- Dohledatelnost dokumentu: vyhledat konkrétní papírový dokument mezi desítky nebo stovkami jiných často bývá velice časově a fyzicky náročné. V podobné situaci je s elektronickým dokumentem pro úspěšné nalezení postačující znát alespoň jeho základní charakteristiky.
- Distribuce dokumentu: rozeslání více listinných dokumentů na různé poštovní adresy může být drahé ekonomicky i časově. Elektronický dokument lze snadno a levně distribuovat, například pomocí elektronické pošty.
- Zálohovaní dokumentů: listinné dokumenty nelze zálohovat, je pouze možné vytvořit kopie, které se následně uloží na nové bezpečné místo. Elektronické dokumenty se ukládají do speciálních elektronických úložišť, které jsou pravidelně zálohovány.

3.2 Co je elektronická správa dokumentů

Dokumenty ve fyzické formě provázejí lidstvo odnepaměti. Vznikají stále nové a nové a již několik století mají pro lidstvo nepostradatelný význam. Aktuálně se dá říci, že dnešní společnost ve všem vychází z dokumentů. Se začátkem použití elektronické formy dokumentů vznikla potřeba se všemi dokumenty efektivně pracovat, ideálně na jednom místě, a tudíž i potřeba aplikace, která toto umožní. K tomu slouží systémy, jež se v průběhu své historie označovaly různě. V dnešní době se setkáváme s označeními jako Document Management System (DMS) nebo Electronic Document Management (EDM) [10]. EDM systém je jedním z typů DMS systémů [2]. V rámci této diplomové práce podrobněji rozebereme EDM systémy pro správu dokumentů.

Elektronická správa dokumentů (z angl. *Electronic Document Management, EDM*) je aplikace, která umožňuje správu, organizaci, evidenci a archivaci různých druhů dokumentů v digitální podobě pomocí počítačových programů a úložišť [58]. EDM může být potřeba v podnicích, které pracují s velkým množstvím dokumentů, jako jsou faktury, objednávky, fotografie, telefonické rozhovory nebo videoklipy. EDM lze kombinovat nebo integrovat do jiných aplikací, může být například kombinován s přístupem řízení pracovních postupů či procesů v podniku [51].

Aplikace EDM je jádrem EDM systému (nebo *Electronic Document Management System, EDMS*), který je softwarovým programem pro správu elektronických informací v pracovním procesu organizace [28, 3]. EDMS umožňuje jeho uživatelům vytvářet dokumenty nebo pořizovat kopie v elektronické podobě, ukládat, upravovat, vyhledávat, tisknout, zpracovávat a jinak spravovat dokumenty v obraze, ve videu a zvuku i v textové podobě

[47, 51, 3]. Vlastnost daného systému zpracovávat digitální skenované verze originálních papírových dokumentů dovoluje postupně zredukovat použití a množství papírových dokumentů v podniku, který se rozhodl uvést EDMS do vlastních pracovních procesů [41]. EDM systém obvykle poskytuje jediný centralizovaný pohled na více databází a může zahrnovat skenery pro snímání dokumentů, tiskárny pro vytváření tištěných kopií, paměťové zařízení, jako je redundantní řada nezávislých diskových systémů a programy počítačových serverů a serverů pro správu databází obsahujících dokumenty [51].

3.3 Principy/funkce EDMS

Na současném trhu jsou dostupné různé DMS systémy, které obecně nabízejí nejrůznější funkce pro lepší správu dokumentů. Dané funkce jsou současně i hlavními principy EDM systémů tak, jako to podrobně popisují zásady EDMS. Samozřejmě ne všechny EDMS disponují stejnými funkcemi, ale každý takový systém je povinen nabízet minimální sadu funkcí, která se nazývá základní [3, 38]. Základní sada libovolného EDMS musí zahrnovat funkce: bezpečnostní (včetně přístupových práv), správa verzí dokumentu, zachycení metadat [38]. Technologický pokrok a boj o zákazníky však přinutil tvůrce DMS systémů pohybovat se kupředu: zjišťovat a nabízet uživatelům nové užitečné funkce, které by ve výsledku výrazně snížily náklady na správu dokumentů, a staly se nenahraditelnými při činnosti podniku. V důsledku toho se funkce systémů pro správu dokumentů výrazně rozšířily o celou řadu dalších, jež se v dnešní době staly jednoznačným standardem pro podniky, které to s používáním EDMS myslí vážně [38]. Níže je uveden seznam funkcí, které nabízí většina dnešních EDM systémů [42, 38, 47, 14, 20, 39]:

Vkládání dokumentů

Všechny současné systémy pro správu dokumentů nabízí možnost vkládaní dokumentů do systému prostřednictvím hromadného ukládaní dokumentů, jednoduchého formuláře, a také převodem listinných dokumentů do elektronické podoby. Některé systémy na správu dokumentů nabízí dokonce možnost vkládaní pomocí FTP[1]. Během použití dané funkce je důležité doplňovat metapopis dokumentů pro následnou jednodušší manipulaci.

Metadata

Metadata jsou nedílnou součástí každého dokumentu. Slouží k přidání dodatečných informací k dokumentům a jejich revizím, které se následně budou využívat jinými funkci EDM systému. Do metadat se zahrnují například tyto atributy dokumentu: popis obsahu dokumentu, kontextu, struktury; autor, datum vytvoření, kategorie dokumentu, atd. Nejjednodušší EDM systémy nabízejí pouze možnost přidání metapopisu prostřednictvím sady předdefinovaných parametrů pro jeho vytvoření. Pokročilejší EDMS umožňují spravovat metapopis dokumentů, revizí pro celý systém nebo jej řídit až na úrovni složek. Metapopis plní důležité funkce v podobných systémech, a to významně pomáhá při vyhledávání, identifikaci dokumentů a umožňuje snadnější seskupování, filtrování a přenášení obsahu do jiných podnikových procesů, jako například vykazování nebo pracovního postupu.

[1] **FTP** je v informatice protokol pro přenos souborů mezi počítači pomocí počítačové sítě.

Vyhledávání dokumentů

Efektivní vyhledávání je zaručeno přítomností a použitím metapopisu v dokumentech. Na základě metadat se dokumenty dají vyhledávat například podl, autora, názvu atd. EDMS také nabízejí možnost fulltextového vyhledávání v dokumentech, pokud ty poskytují fulltextový index. Vidíme tedy, že EDMS jsou schopny vytažený fulltextový index začlenit do vyhledávání. Pokročilejší systémy pro správu dokumentů nabízejí uživatelům možnost definovat kaskádovité podmínky vyhledávaní (hledat ve vyhledaném), nebo omezit prohledávaní jen určitých polí (např. Název nebo Autor).

Automatizace procesů nebo workflow

Jedná se o funkci, která je nadstavbou, jež se však stala standardem dnešních EDMS. Daná funkce EDMS podporuje procesy oběhu dokumentace: automatizace procesů, které umožní uživatelům vidět pohromadě „dokumenty k vyřízení“ (přezkumy, schválení, archivování a uchovávání dokumentů). Také automatizované procesy umožňují předávat dokumenty, informace či úkoly mezi uživatele podle sady procedurálních pravidel.

Cílem workflow je efektivní spolupráce zaměstnanců i celých oddělení ve firmě, čemuž právě systémy na správu dokumentů s podporou workflow významně pomáhají. Implementace workflow (z ang. *tok dokumentů*) zavádí standardizace pracovních postupů, přispívá k jejich zjednodušení a zlepšuje organizaci práce. To následně vede ke zvýšení kvality, efektivity práce a snížení nákladů. Workflow je důležitý pro management podniku a poskytuje mu nástroje pro sledovaní dokumentů v rámci jednotlivých procesů. Úlohu řízení automatizovaných procesů na úrovni všech používaných aplikací zdůrazňuje v dnešní době velmi používaný BPM neboli Business Process Management, který umožňuje modelovat, monitorovat, analyzovat a vyhodnocovat pracovní procesy. Také s workflow je často spojován pojem BPR neboli Business Process Reengineering, který se používá v případě, kdy je nutné provést radikální změny ve stávajících procesech.

Kontrola a správa verzí

Tato funkce umožňuje sledovat změny v obsahu dokumentu nebo jeho metadatech bez nutnosti samostatného uchovávání pojmenované verze. Kdykoliv v budoucnosti je pak možné se vrátit k některé z předchozích verzí v případě, že dojde k nežádoucí změně, smazání časti nebo celého dokumentu. U každé takové změny se eviduje datum a uživatel, který danou změnu provedl. Daná funkce předpokládá, že EDMS je schopný automatický přidělovat identifikátory dokumentům a jejich revizím. Tyto identifikátory jsou reprezentovány jednoznačnými čísly ze správní řady v rámci systému nebo složky.

Dostupnost a sdílení

Dostupnost dokumentů je zařízena centralizovaným přístupem k databázi souborů. Na základě toho jsou dokumenty poskytovány z jediného centrálního zdroje, a to distribuovaného nebo replikovaného. Dokumenty nacházející se na tomto zdroji jsou vždy aktuální. Díky centralizaci se snižuje redundance dat a odpadává nutnost vyměňovat dokumenty mezi uživateli – stačí pouze vložit dokument do centralizované databáze a nasdílet ho ostatním uživatelům pomocí odkazu na pozici, kde se dokument nachází, namísto stahování či trvalé kopie.

Archivace a zálohování

Nutnost archivace dokumentů nastává v podniku v situaci, kdy je potřeba řešit, jak postupovat se zaplněným firemním diskem. Trendem dnešní doby je digitalizace a ukládaní dokumentů do elektronických repozitářů, které jsou další součástí EDM systémů. Do nich je možné vkládat nejen zdigitalizované dokumenty, ale také vyřízenou elektronickou agendu podniku nebo aktuálně používaná data v rámci zálohování. Vyexportování dat a jejich přenesení do archivačního úložiště je vyřešeno použitím strukturovaného XML.

Přístupová práva

Daná funkce zabezpečuje omezení přístupu neautorizovaných uživatelů k datům systému. Každý systém poskytuje alespoň základní přidělovaní práv: kdo může soubor číst, editovat, mazat, přesouvat. Přístupová práva se dají nastavit určité skupině uživatelů, konkrétním osobám, atd.

Existuje celá škála EDMS, které mají libovolně možné variace nastavení systému přístupových práv: od těch, kde se přístupová práva nepoužívají vůbec a všem je dovoleno „vidět“ vše, až po ty, které jsou zabezpečeny řízením přístupových práv na úrovni složek nebo i jednotlivých dokumentů či dokonce položek metadat.

Bezpečnost

Bezpečnost je jednou z nejdůležitějších funkcí EDM systémů. Zamezuje množství nepříjemných problémů a výdajů v případě úniků informací. Funkce je zabezpečená centralizovanou zprávou souborů: data jsou uložena v chráněném úložišti, ze kterého jsou dostupné pouze prostřednictvím funkcí EDM systémů. Také bezpečnost dat se zabezpečuje prostřednictvím víceúrovňové správy uživatelů, kdy jsou evidovány dle uživatelských účtů, rolí a jmen. Také bezpečnost výrazně souvisí s problematikou přístupových práv, které hrají důležitou roli v předcházení situací s únikem firemních informací.

Integrace

Práce s dokumenty vyžaduje použití mnoha různých kancelářských aplikací pro ukládaní, otevíraní, modifikace souborů (typicky nástroje Microsoft Office). Proto EDMS zabezpečují integrace s kancelářskými aplikacemi, například soubor otevírá aplikace, ve které byl vytvořen. Mezi aplikace a EDMS jsou integrovány unikátní identifikátory dokumentů. Je také možné propojení EDMS s aplikacemi jako kalendář nebo elektronická pošta, například pro zasílání nejrůznějších notifikací (o přečtení dokumentu, o přiděleném úkolu atp.).

3.4 Předpoklady zavedení EDMS v podniku

Může se zdát, že EDMS je „novodobou“ zbytečností, kterou se firmám snaží prodat prodejci, a vždy se najde člověk, který toto za zbytečnost bude považovat. Důvody k tomu mohou být různé: firma zatím „dokumentoobrat“ zvládá vše rychle a téměř bez problémů, bude těžké zaškolit kolektiv a především ho přeladit na použití nového složitého softwaru, atd. [22]. Ale když se na to podíváme z pohledu zkušeností větších společností, a také z dlouhodobého hlediska, zjistíme, že EDMS je nepostradatelným pomocníkem projektových manažerů a zavádějí se v situacích, jako jsou následující [41]:

- Oběh dokumentů ve firmě probíhá neřízeně, dokumenty jsou uloženy v desítkách šanonů, ale nikdo z pracovníku přesně nemůže říct, kde se ten nebo jiný hledaný potřebný dokument nachází.
- Během zpracovaní elektronických dokumentů (zvlášť, jestliže s nimi jako obvykle pracuje více pracovníků) si pracovníci posílají dokument mezi sebou elektronickou poštou. Následně je třeba například dokument dohledat, nastává problém, že se musí vyřešit kdo a kdy ho e-mailem posílal a s jakým předmětem.
- Je fakt nekontrolovaného skladování elektronických dokumentů v paměti uživatelských počítačů nebo ve sdílených složkách na serverech. S takovým přístupem se k dokumentům bez velkých obtíží může dostat kdokoliv s cílem přečtení, modifikace nebo i smazaní důležitých dokumentů a informací v nich uložených.
- Další případ bude popsán na základě příkladu: za faktury a účetnictví je zodpovědný účetní, který jako jediný má k účetnictví firmy přístup. Jak ale postupovat v situaci, když se občas určitou fakturu nepodaří najít nebo se doklad i ztratí.
- Opakovaně nastává situace, kdy se hledá konkrétní dokument po delší době, a na základě předešlých zkušeností si dovolím tvrdit, že může nastat skutečnost, že jej nemusíme najít vůbec.
- Opakovaně se vynakládá energie na hledání nebo opětovné vytvoření toho, co se nepodařilo najít, namísto účelné práce, která pracovníky motivuje více, než klasický hledací proces.
- Činnost podniku je opakovaně narušena kvůli nedohledání dokumentů a počet podobných případů roste s počtem nových zaměstnanců a poboček.

Tyto situace jsou jen zlomek toho, čemu se dá předejít zavedením EDM systému ve firmě. Avšak výše popsané případy musí posloužit vedoucím rozvíjející se společnosti jako první signály pro seriózní zamyšlení se nad zavedením vyhovujícího EDM systému.

3.5 Modely EDMS

Systémy EDM mají z principu databázové jádro. Databázi (neboli bank dat, datová základna) si lze představit jako místo, kam se ukládají data [1]. Jednoduchým příkladem z reálného života je například knihovna nebo kartotéka, kde jsou data zorganizovány a uloženy podle určitého systému. V případě, že víme, jak daný systém použít, můžeme s nimi efektivně pracovat: uložit, vyhledat požadované data mezi ostatními [17].

Studenti

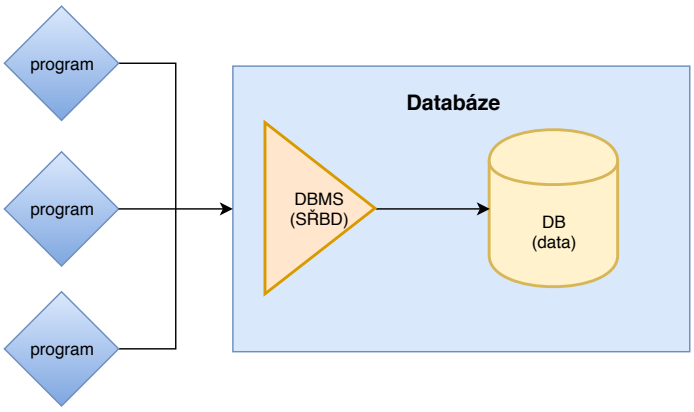
IČ	Jméno	Příjmení	Adresa	Okres	Rodné číslo	Věk	Typ školy
1	Jana	Nováková	Frýdecká 18, Nový Jičín	Nový Jičín	685112/5648	36	VOŠ
2	Kamil	Strouhavý	Přibilova 15, Frýdek Místek	Frýdek Místek	851213/4567	19	VOŠ
3	Elena	Michnová	Letní 45, Ostrava	Ostrava	895203/4532	15	SOŠ
4	Klára	Janíková	Mošnov 45	Ostrava	885111/3462	16	SOŠ
5	Julie	Přibilová	Rybí 453	Nový Jičín	875312/4675	17	SOŠ
*	(Nové)					0	

Záznam: 1 z 5 Bez filtru Vyhledávání

Obrázek 3.1: Znázornění obecného vzhledu DB.

Databáze je tedy systém souborů (dat) s pevnou strukturou záznamů, které se vztahují k jednomu tématu a jsou mezi sebou propojeny. Jeden z možných vzhledů databáze je

znázorněn na obrázku 3.1. Obvykle se rozlišují pojmy báze dat (DB), což jsou samotná data, a také pojem systém řízení báze dat – SŘBD (z angl. *Data Base Management System, DBMS*), který se stará o správné fyzické uložení dat do DB, zabezpečuje požadované funkce (přístup, vyhledávaní, atd.) nad daty a jejich správu. V závislosti na kontextu v současné době se běžně označením databáze myslí uložená data i software (SŘBD) [9]. Obrázek 3.2 slouží pro lepší pochopení vztahu DB a SŘBD.



Obrázek 3.2: Znázornění vztahu DB a SŘBD.

Podle toho, na jakém databázovém jádru je EDM systém založen, existují tři druhy EDMS modelů podle způsobu ukládaní dat: relační, objektový nebo dokumentový. Výhody a omezení EMDS definuje jeho jádro.

3.5.1 Relační model

Nejrozšířenějším a nejpoužívanějším modelem v současnosti je relační model. Tento model databáze byl vynalezen v roce 1970 E.F. Coddem, mladým programátorem v IBM. Ve svém článku „A Relational Model of Data for Large Shared Data Banks“ Codd navrhl posun od uložení dat v hierarchických nebo navigačních strukturách do organizování dat v tabulkách obsahujících řádky a sloupce [64].

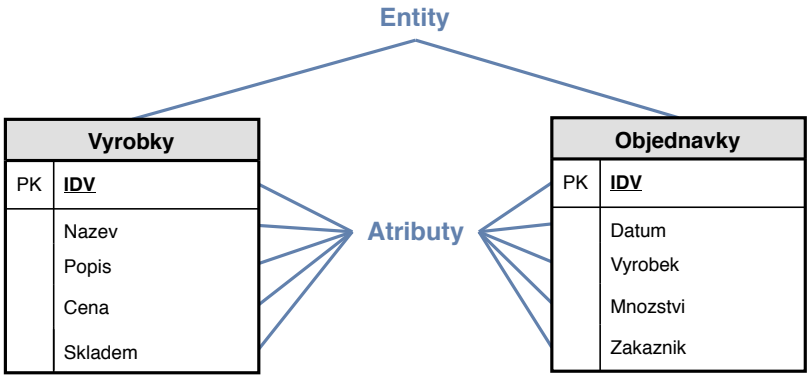
Relační databáze je sada formálně popsaných tabulek, ze kterých lze získat nebo znovu sestavit data různými způsoby, aniž by bylo potřeba reorganizovat databázové tabulky. Standardní uživatelské a aplikační programovací rozhraní (API) relační databáze je jazyk Structured Query Language (SQL). Příkazy SQL se používají pro interaktivní dotazy pro informace z relační databáze i pro shromažďování dat pro sestavy [44].

Tabulky

V oblasti relačních databází se nejčastěji setkáme s konstatováním, že základní struktura takových databází je tvořená tabulkami, tj. že data jsou v relační databázi strukturovaná do *tabulek*. V teorii relačního modelu ale pojem tabulka nevyhovuje exaktnímu matematickému vyjadřování, proto se zde používá pro označení takové tabulky pojem *relace*. Tedy, data v takové databázi jsou na sobě závislá a logicky uspořádané do relací, které se reprezentují tabulkami [64]. Příklad vzhledu tabulky relační DB také představuje obrázek 3.1.

Jedna relace/tabulka vždy popisuje pouze jednu *entitu*. Entitou se rozumí prvek reálného světa, jako například: osoba (student, zaměstnanec, zákazník, atd.), věc (auto, před-

mět, atd.), místo (pekárna, přednášková místnost, atd.), myšlenka, o které shromažďujeme data (objednávka, výpůjčka, atd.) [64, 44].



Obrázek 3.3: Znázornění entit Výrobky, Objednávky spolu s atributy.

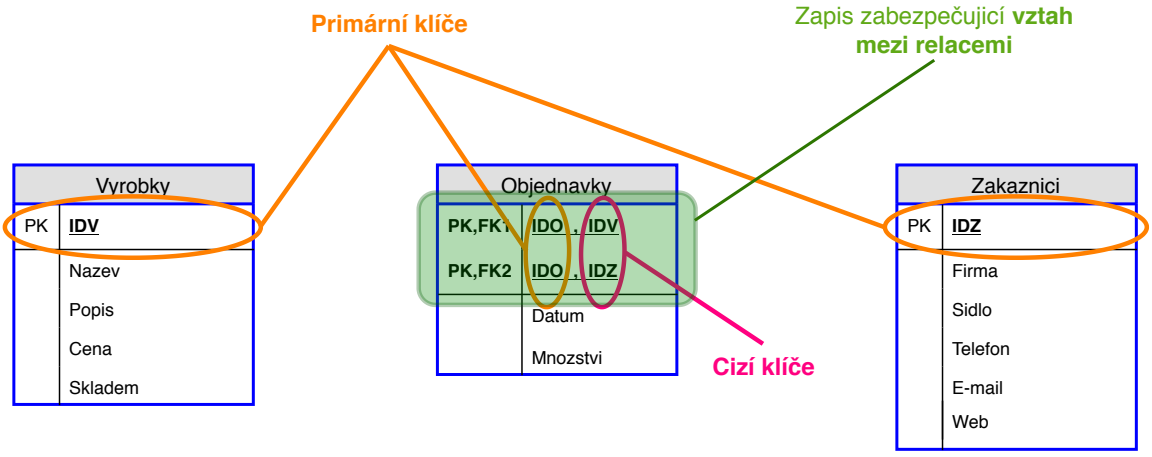
Každá tabulka se skládá ze sloupců a řádků. Vlastnosti každé entity popisují *atributy*, které jsou reprezentované sloupci. Každý sloupec musí být definován jedinečným názvem a vyhovujícím datovým typem podle dat v něm uložených (text, číslo, logická hodnota, atd.). Atribut je atomická jednotka (nejde rozložit na další jednotky) informace, která popisuje entitu. Například, v entitě„Zaměstnanec“ atributy budou následující: název, popis, cena, skladem 3.3. Jinak se atributu také říká *pole* tak, jako jsou to jednotlivá pole v záznamu (řádku tabulky).

Každý jeden řádek tabulky reprezentuje konkrétní záznam v databázové tabulce, tj. záznam je tvořen údaji v jednom řádku databázové tabulky. Jsou to informace o jednom popisovaném objektu v databázi, například informace o pracovníku, knize, výrobku. Níže na obrázku 3.4 jsou znázorněny vztahy výše definovaných pojmů ve výsledné tabulce, kterou vidí konečný uživatel naprogramované DB [64, 12].

Pracovníci

Titul	Příjmení	Jméno	Pohlavi	Datum narození	Délka praxe	Vzdělání
Ing.	Kovářová	Markéta	žena	11.1.1978	10	VYUČ
	Zeman	Petr	muž	18.5.1974	7	VYUČ
	Pokorný	Jan	muž	16.1.1972	6	VŠ
	Gut	Andrej	muž	20.11.1970	3	VYUČ
	Hlinka		muž	21.9.1955	9	VYUČ
	Novaková		ena		5	VŠ
	Plechatý		muž	13.4. 67	8	VYUČ
Mgr.	Jirásek	Alois	muž	1.10.1963	25	VŠ
Ing.	Skvor	Ferdinand	muž	28.6.1978	16	VŠ
	Stránská		ena	15.7.1993	1	VYUČ
	Palyzová		ena	22.8.1964	23	VYUČ
	Humr	Klement	muž	4.11.1967	11	VYUČ

Obrázek 3.4: Znázornění vztahů základních prvků relační DB.



Obrázek 3.5: Znázornění vztahů mezi entity relační DB pomocí cizích a primárních klíčů.

Primární a cizí klíče

Každá tabulka musí mít atribut, pomocí kterého bude možné jednoznačně identifikovat každý příslušný záznam neboli řádek tabulky. Takový atribut se v teorii relačního modelu nazývá *primární klíč (PK)*. Jedná se o kombinaci sloupců nebo jen jeden sloupec, který zaručí unikátnost klíče napříč tabulkou a DB. PK se reprezentuje určitým jedinečným identifikačním číslem n z číselné řady s tím, že každý další záznam dostává číslo $n+1$ ve srovnání s předchozím. Číslováni lze zařídit automatický použitím vhodných funkcí při návrhu tabulek. Typicky se takové číslo nazývá *ID_jmeno_objekta* a plní funkce primárního klíče. Také podle pravidla integrity entit hodnota primárního klíče nikdy nesmí chybět (ukazovat na NULL) [64, 46].

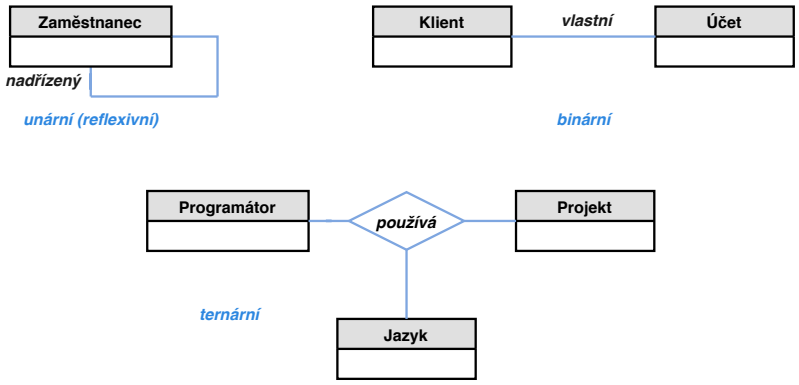
Dalším důležitým pojmem je *cizí klíč, FK* neboli nevlastní klíč. Cizí klíč je atribut v určité tabulce, který je v jiné tabulce primárním klíčem. Slouží především pro vyjádření vztahů (relací) mezi řádky databázových tabulek. Jedná se o atribut či jejich skupinu, která nám umožní identifikovat spolu související záznamy z různých tabulek, tj. cizí klíč reprezentuje vztahy mezi tabulkami databáze. Cizí klíč má následující vlastnosti [64]:

- V případě složeného cizího klíče je jeho hodnota platná, jestliže je plně zadaná nebo plně nezadaná.
- Cizí klíč je zodpovědný za vytváření vztahů mezi tabulkami a jeho hodnota se musí shodovat s některou hodnotou ve sloupci v odkazované tabulce. Odkazující se řádek tak má právo odkazovat pouze na jeden řádek v odkazované tabulce, proto musí být platně zadané hodnoty unikátní v odkazovaném sloupci (v praxi jde z pravidla o primární klíč odkazované tabulky).

Lépe pochopit vzájemný vztah cizího a primárního klíče nám pomůže obrázek 3.5.

Integritní omezení

Soulad hodnot klíčů v databázi ji drží pohromadě a představuje významné pravidlo relační databáze – *referenční integritu (omezení)*, které uvádí, že relační databáze nesmí obsahovat žádnou nesouhlasnou hodnotu cizího klíče. Jednoduše řečeno – referenční omezení zamezují uživateli zadávat do DB nesprávná data. Data v referenční databázi musí také splňovat



Obrázek 3.6: Znázornění stupňů vztahů mezi tabulkami DB.

všechna *integritní omezení*, a jen tehdy je můžeme považovat za konzistentní. Integritními omezení se nazývají všechna omezení, které se kladou na data v DB, a můžeme je rozdělit do dvou skupin [64]:

Obecná omezení jsou omezení, jež musí být splněny v každé DB daného (relačního) typu, bez ohledu na konkrétní zaměření aplikace, ve které se DB bude používat. Dané omezení definuje relační model pro sloupce, které v budoucnu budou sloužit jako primární a cizí klíče.

Specifická omezení závisí na konkrétní aplikační oblasti. Například, v případě vytvoření DB pro informační systém naší fakulty by se takové omezení týkalo hodnoty přihlašovacího jména (musí to být znakový řetězec, jehož délka stanoví 8 znaků, kde první znak je písmeno „x“, dalších 5 znaků jsou písmena a poslední dva jsou čísla).

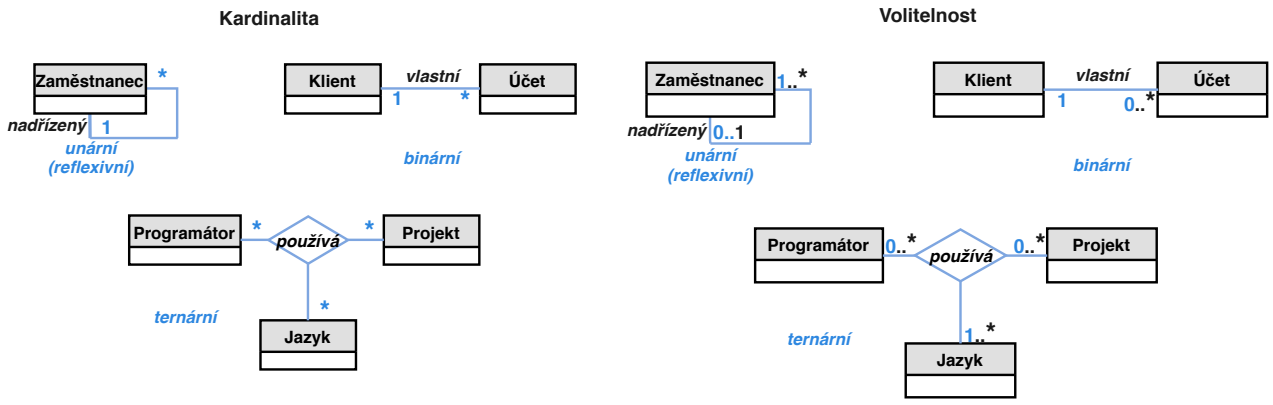
Základní datové typy

V relační databázi jsou nejčastěji používané následující datové typy: Integer (INT) – celé číslo, Varchar (VARCHAR) – krátký řetězec do 255 znaků, Text (TEXT) – delší text, Boolean (BOOLEAN) – logická hodnota, Date (DATE) – datum formátu YYYY-MM-DD [64].

Vazby mezi tabulkami

Vztahy mezi tabulkami v relační databázi jde realizovat několika způsoby na základě vztahu informace v jedné tabulce k informacím v jiné. Každý vztah je realizován třemi vlastnostmi [64, 62]:

- Stupněm: unární, binární, ternární, N-ární vztah, kdy se vztah realizuje mezi jednou, dvěma, třemi, n-relacemi zároveň. Obrázek 3.6.
- Kardinalitou vztahu – počtem vztahů daného typu, ve kterých může participovat jedna entita (žádný, 1:1, 1:N, M:N, případně přesněji). Obrázek 3.7 znázorňuje výše napsané typy kardinalit.
- Volitelností účasti v určitém vztahu vyjadřuje minimální počet vztahů daného typu, ve kterých musí participovat jedna entita (0 – volitelné/1 – povinné), resp. účast entitní množiny ve vztahu (částečná/úplná). Je znázorněná na obrázku 3.7.



Obrázek 3.7: Znázornění druhů kardinality a volitelností vztahů mezi tabulkami DB.

Transakce s důrazem na ACID

Relační databáze využívá transakce s důrazem na ACID [52]. Akronym ACID se dekóduje jako [18]:

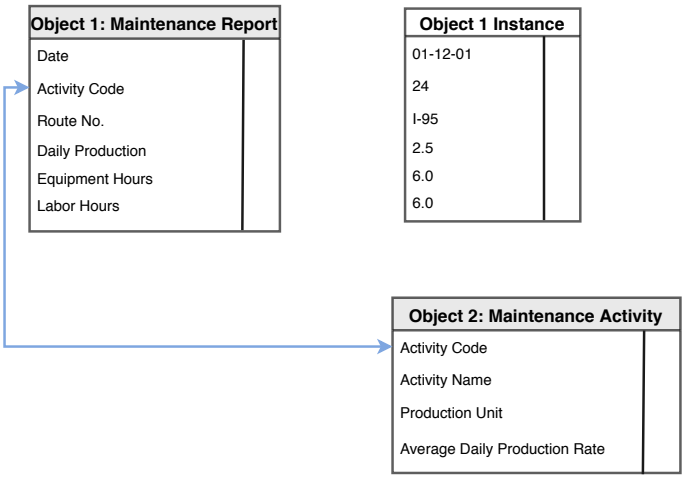
- **Atomicita** (z angl. *atomicity*) – transakce je atomická, tj. buď se provede celá nebo se neprovede vůbec.
- **Konzistence** (z angl. *consistency*) – transakce převede databázi z jednoho konzistentního stavu do jiného. Je zaručen zápis pouze platných dat bez porušení integritních omezení.
- **Izolovanost** (z angl. *isolation*) – změny v rámci jedné transakce před jejím dokončením nejsou viditelné ostatním transakcím.
- **Trvalost** (z angl. *durability*) – změny úspěšně dokončené transakce jsou v databázi uloženy natrvalo.

EDM systémy založené na relačním modelu

EDM systémy založené na relačním modelu využívají následující prostředí pro návrh a vytvoření relační databáze: For Projects, MS SQL, MySQL, PostgreSQL, Oracle, SAP [29]. Jedním z EDM systémů využívajících relační model DB je aplikace *IS ALeX* [6].

Výhody a nevýhody relační databáze

Aktuálně jsou relační databáze hlavním prostředkem pro správu dat různých komerčních aplikací. Je zřejmé, že díky své schopnosti spravovat velký objem údajů a mít při tom jednoduchou strukturu se dá s jistotou říci, že relační databáze zůstanou i nadále hlavním prostředkem pro správu dat většiny komerčních aplikací. Každopádně má tento model DB řadu výhod i nevýhod. Jednoznačnou výhodou je snadná standardizovatelnost a přenositelnost díky své jednoduchosti. Další výhodou i nevýhodou je to, že daný model omezuje vztahy a strukturu hodnot, se kterými pracuje na pouhou množinu tabulek na předem definované množině zakládaných datových typů. Velkou nevýhodou relačního modelu je velký rozdíl ve schématu reálných dat a vnitřní tabulkové struktury DB. Vztahy mezi daty se reprezentují pouze tabulkami a může nastat situace, kdy databáze ztratí přehlednost a stane



Obrázek 3.8: Příklad objektově orientovaného modelu DB.

se hůře spravovatelnou. Do takové situace může vést aplikace ze složitějším datovým modelem, který bude potřebovat velké množství tabulek vzájemně provázaných pomocnými odkazy (tzv. klíči). Špatná spravovatelnost znamená pro programátora nutnost zasáhnout do tabulkové reprezentace pokaždé, když se objeví změny [66].

3.5.2 Objektový model

V době, kdy se začal vyvíjet a aktivně používat relační databázový model, se již začínaly navrhovat aplikace, pro které nebyl klasický relační model dat zcela dostačující, a požadavky některých aplikačních oblastí překračovaly jeho možnosti. Týkalo se to především aplikací z oblasti multimédií, designu, geografických systémů apod. Byla potřeba vynalézt nový databázový model, který by umožnil lepší mapovatelnost mezi složitými reálnými daty a jejich reprezentaci v databázovém systému [66, 63].

Koncem 60. a během 70. let se začaly objevovat první objektové jazyky a poprvé vznikla myšlenka ukládat data v objektové podobě. Na konci 80. let začal rychle růst vliv paradigmatu objektové orientace (OO) a následně možnost jednoduše pracovat se složitě strukturovanými daty. Od té doby se objektová orientace začala výrazně prosazovat v oblasti programování, návrhu a postupně i analýzy [63]. V následkutoho byl návrhnout a implementovan objektový model dat, tj. objektová databáze. V roce 1991 spatřil svět první dokument (kniha), která se považuje za objektový „standard“ Object Data Management Group (ODMG 1.0.), další ODMG 3.0. v roce 2000 [32]. Slovo standard bylo napsáno v uvozovkách tak, jako pro objektové databáze neexistuje všeobecně přijatý standard obdobný jako SQL.

Objektově orientovaná databáze a také *Objektově orientované systémy databází* (z angl. *object oriented database management systems, OODBMS)* – je systém správy dat/databázi, ve kterém je informace představena formou objektů (obrázek 3.8). Daný systém vychází ze známých principů objektově orientovaného programovaní a modelovaní, je ale navíc obohacen o podporu tříd objektů, dědění vlastností, metod třídy podle podtříd a jejich objektů, techniky reprezentace vztahů, perzistence, dotazování, apod. [5]. Níže budou popsány základní principy objektového modelu [49].

Objekty a třídy

Objektově orientovaný model se maximálně podobá reálným objektům a místo řádků tabulky se zde přímo ukládají objekty, které se charakterizují třídami. Tyto objekty jsou podobné objektům, jak je známe z objektových programovacích jazyků. Z toho vidíme, že tento model dekomponuje reálné informace ze světa na *objekty*.

Objekt je libovolná entita, kterou jsme schopni jednoznačně a nezávisle identifikovat v rámci konkrétního kontextu reálného světa. Z toho je zřejmé, že i několik datově shodných objektů jsou navzájem odlišné a každý jeden z nich má jednoznačnou identitu. V objektovém modelu se taková identita zajišťuje OODBM systémem, který generuje unikátní *identifikátor (z ang. object identifier, oid)*. Daný identifikátor zůstává neměnný po celou dobu života objektu [57].

Třída je abstraktní popis objektu, který definuje složky objektu, operace (v rámci OO nazývané *metody*), které lze nad objektem provádět. Jinými slovy, třída je šablona sloužící k definici uživatelských datových typů. Každý jeden objekt je instancí určité třídy. Každá třída je schopná instanciovat nekonečný počet objektů, které jsou strukturálně shodné. Třídy mají *rozhraní* a *implementaci*, tj. popis atributů, operací a kód popisující činnost operací [50]. Příklad vzhledu třídy je představen níže ve formě kódu.

```
public class Auto {
    private static int pocet = 0; //Staticka promenna
    private int rok_vyroby; //Promenne instance
    private boolean stav;
    private float hmotnost, o_rychlost, m_rychlost;

    public Auto () //Bezparametricky konstruktor
    {
        pocet++; // Incrementace pri vytvoreni noveho objektu
        rok_vyroby = 2000;
        stav = true;
        hmotnost = 1000;
        o_rychlost = 100;
        m_rychlost = 130;
    }
}
```

Literály

Kromě objektů se v objektově orientovaném modelu také používají *literály*. Jsou to datové entity určitého datového typu s tím rozdílem, že literály nemají vlastní identitu. Literály se nejčastěji používají jako datové atributy objektů. Nejde nad nimi měnit množinu operací kvůli tomu, že množina operací nad datovými typy literálů je pevně stanovena. Také, oproti objektům, nejsou literály *proměnlivé* (z angl. *mutability*) tak, jako není možné měnit hodnoty jejich datových složek [49].

Operace

Existuje několik typů operací nad objekty [49]:

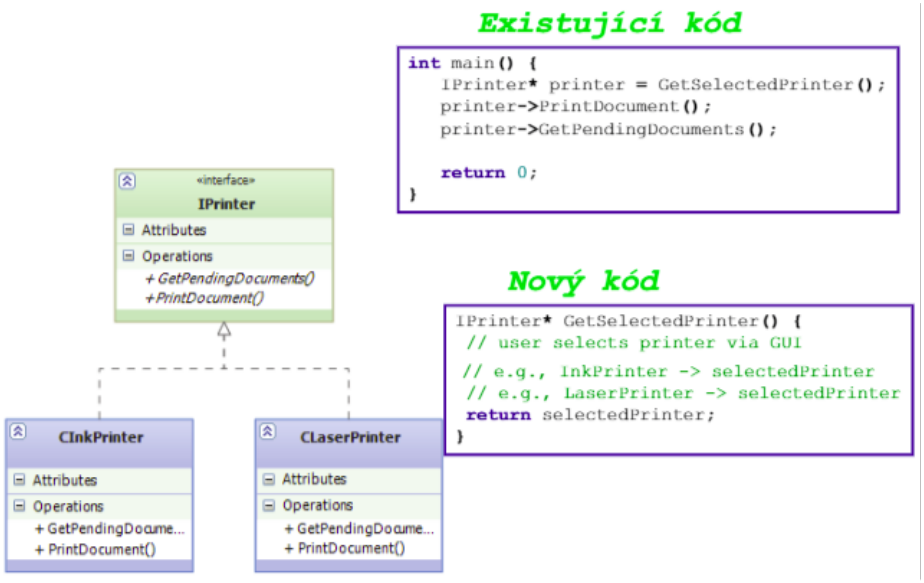
- *Konstruktor:* provádí inicializaci objektu v okamžiku jeho vytvoření. Každý objekt může mít jeden nebo více konstruktorů. Jeden z možných vzhledů konstruktoru je představen kódem uvedeným níže.
- *Destruktor:* provádí úklid objektu před jeho smazáním. Volá se v okamžiku rušení objektů.
- *Mělké (z angl. shallow) kopírování:* provádí kopírovaní atributů objektu tak, že odkazy na jiné objekty se zachovávají, tj. ukazují na stejné objekty jako v originálním objektu.
- *Hluboké (z angl. deep) kopírování:* provádí kopírovaní nejen atributů objektu, ale také i kopie objektů, na které se originální objekt odkazoval.
- *Ostatní:* metody sloužící k přiřazovaní hodnot atributů, zjišťování hodnot atributů, k manipulaci s atributy objektu, provádění výpočtů, produkce uživatelského výstupu, atd.

```
//Parametricky konstruktor
public Auto (int rok_vyroby_, boolean stav_, float hmotnost_,
           float o_rychlost_, float m_rychlost_) {
               pocet++;
               rok_vyroby = rok_vyroby_;
               stav = stav_;
               hmotnost = hmotnost_;
               o_rychlost = o_rychlost_;
               m_rychlost = m_rychlost_;
}

// Nebo

public Auto (int rok_vyroby_, boolean stav_, float hmotnost_,
float o_rychlost_, float m_rychlost_) : rok_vyroby(rok_vyroby_),
stav(stav_), hmotnost(hmotnost_), o_rychlost(o_rychlost_),
m_rychlost(m_rychlost_) {pocet++;}
```

Zapouzdření

Jak bylo zmíněno výše, okolí objektů má přístup pouze k rozhraní jeho operací a implementace operací je vždy skryta před okolním světem. Jedná se o jednu z typických vlastností objektově orientovaného přístupu, která zvyšuje míru abstrakce a nezávislosti objektů. Kromě skrývání implementace je možné s různým stupněm skrývat rozhraní operace a atributy označit jako [21]:

- veřejné (z angl. *public)* – budou přístupné z okolních objektů;
- soukromé (z angl. *private)* – budou přístupné pouze v rámci daného objektu;
- chráněné (z angl. *protected)* – budou přístupné v objektech dané třídy, a také v objektech tříd, které z této třídy byly zděděny.



Obrázek 3.9: Příklad vzhledu dědičnosti v OOP.

Dědičnost

Dědičnost (obrázek 3.9) reprezentuje opětovné použití rozhraní, a také umožňuje používat hierarchický přístup při návrhu tříd. Třídy nižší úrovně mohou dědit vlastnosti a chování jedné nebo více nadřazených tříd. Novou třídu je tedy možné vytvořit odvozením z podobné existující třídy a přidáním nových potřebných atributů a operací nebo případně předefinováním zděděných. Dědičnost může být jedno nebo vícenásobná. V prvním případě to znamená, že třída může dědit pouze z jedné třídy vyšší úrovně. V druhém případě, že může dědit z několika nadřazených tříd [21].

Polymorfismus

Polymorfismus neboli mnohotvárnost (znázorněna v kódu níže) způsobuje, že objekty na stejný podnět reagují různým způsobem v závislosti na svém typu, tj. činnost operace se může lišit podle třídy objektu, nad kterým se provádí. S polymorfizmem souvisí pojem *pozdní vazba*, který označuje způsob vyvolání polymorfních operací: aplikace při vyvolání operace dynamicky během chodu programu zvolí kód metody na základě třídy objektu, nad kterým byla metoda vyvolaná. Vazba může být statická a dynamická. Statická vazba: vyvolaná metoda dle datového typu odkazu (překladač nezajímá typ objektu, kam odkaz odkazuje). Dynamická vazba: vyvolaná metoda dle datového typu objektu, na který odkaz odkazuje [21, 49].

```
//Odkazna stejny typ objektu, jakeho je sam typu:

Auto a = new Auto();
Osobni o = new Osobni();
a.sjistiVlastnosti(); //Vrati vlastnosti auta
o.zjistiVlastnosti(); //Vrati vlastnosti osobniho auta

//Odkaz ukazuje na jiny typ objektu (v kontextu dedicke posloupnosti), jakeho je sam ty
```

```
Auto c; //Ukazuje na potomka
c = a; //Volani metody tridy auto
c = o;
c.zjistiVlastnosti(); //Volani metody tridy osobni
```

Perzistence dat

Perzistence je schopnost systému uchovávat data trvale, nejen po dobu běhu programu, ale i mezi opětovným spuštěním, takové data se nazývají perzistentní. Data dostupná jen po dobu běhu programu se nazývají tranzientní. Kvalitní databázový systém by měl být schopen podporovat oba druhy dat [66].

Perzistence je jedna z nejdůležitějších vlastností napříč všemi druhy databázových systémů. V rámci objektově orientovaných systémů se perzistence týká udržování libovolných objektů různých tříd. Kvůli možné složitější datové struktuře a velkého množství referencí (vazby mezi strukturami) je důležité aplikovat nejvhodnější typ perzistence. Pro objektově orientované databáze je takovým *ortogonální perzistence*, která splňuje většinu požadavků OODBS. Tento model perzistence zajišťuje ukládaní dat ve stejném tvaru jako v paměti, transparentní otvíraní a ukládaní, načítaní dat „on demand“, stejný systém perzistence pro data a kód [36].

Dotazovací jazyk

Dotazovacím jazykem v objektově orientovaných DB vystupuje vždy objektově orientovaný jazyk (např. C++, Java, Smalltalk). Tento jazyk se využívá pro aplikaci a také databázi. Také zabezpečuje těsný vztah mezi objektem aplikace a uloženým objektem v databázi, nabízí možnost hledat objekty pomocí deklarativního programovacího přístupu [11].

Shrnutí základních principů objektového modelu

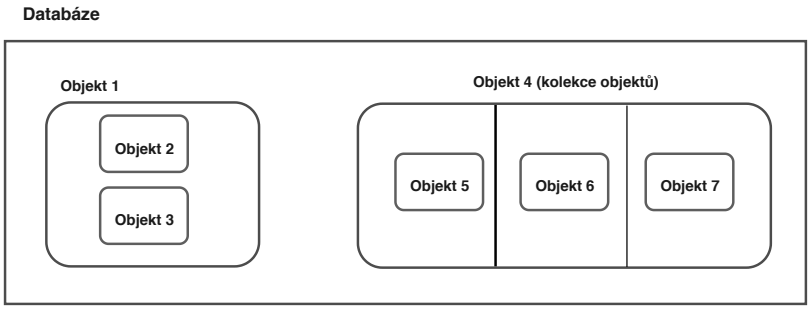
V objektově orientované databázi ukládání objektů probíhá bez nutnosti jejich transformace. Databáze obsahuje třídy s datovou strukturou a metodami. Je možné zapouzdření objektu a dědění tříd, polymorfizmus, používá se co nejpřesnější přístup k souborům (na základě OID) [29].

RDM	**OODM**
Relace (tabulka)	Kolekce (množina objektů – i z více tříd)
N-tice (řádek)	Objekt
Atribut	Datová složka a metody objektu
Primární klíč (na logické úrovni)	OID (na fyzické úrovni)

Tabulka 3.1: Přímé porovnání RDM a OODM z hlediska datového modelu.

Porovnání objektově orientovaného modelu s relačním

Jednotlivé databázové typy mají celou řadu rozdílů [29], které mají dopad na mnoho aspektů v rámci databázových aspektů. Tabulka 3.1 znázorňuje významné rozdíly z hlediska **datového modelu** těchto dvou modelů .



Obrázek 3.10: Ukázka struktury objektové databáze

Také tyto dva modely se výrazně liší **způsobem vkládání** dat do databáze. Oproti relačním databázím je v objektově orientovaných ukládaní výrazně jednodušší: objekt se ukládá bez zbytečných příprav pomocí metod a provideru, který jako parametr použije celý objekt, jež máme v plánu uložit.

Rozdíly podoby modelů dat získaných z databáze při **čtení** závisí na použitém dotazovacím jazyku. Na rozdíl od RDM je návratovou hodnotou v objektové orientovaném modelu objekt, proto není nutné provádět úpravy (např. mapování atd.), ale může být zapotřebí provést některé operace před samotným dotazem.

Další rozdíl mezi RDM a OOM je vidět při **modifikaci** dat. Pro změnu dat v objektově orientované databázi je nutné data načíst a následně po provedení modifikace uložit celý objekt. Tento postup se provádí bez závislosti na velikosti úprav (drobné, rozsáhle).

Operace **mazání** se v obou databázových modelech provádí obdobně. Výhodou OOM oproti RDM je použití OID jako přímého odkazu do paměti.

Tyto dva modely se také liší způsobem logického **provázání** ukládaných dat. U objektově orientovaného modelu je logická struktura (obrázek 3.10) tvořena vnořováním jednoho objektu do druhého. Vnoření reprezentuje OID, který plní i funkci fyzického odkazu na objekt, a také umožňuje co nejpřímější přístup k objektu v DB. Přístup k objektu je pak možný samostatně nebo na základě jeho vnoření v jiných objektech. Při nutnosti získání určitých dat není potřeba provádět vyhledávaní, stačí pouze identifikovat „kořenový“ objekt a objekty, které jsou do něj vnořeny, se načtou spolu s ním. Pro získání potřebných dat nebo přesnějšímu provedení konkrétní operace je možné nastavit „hloubku“ dotazu nebo čtení, úprav a mazání objektů.

EDM systémy založené na objektovém modelu

EDM systémy založené na objektovém modelu využívají následující prostředí pro návrh a vytvoření objektové databáze: Versant Object Database, FastObjects, GemStone, Jasmine, O2, ODE, Objectivity, ObjectStore. Jedním z EDM systémů využívajících objektový model DB je aplikace Xerox DocuShare [11, 47].

Objektově-relační databáze

Tento model je „výsledkem vztahu“ dvou předchozích modelů, krátce si připomeneme jeho obraz. Cílem stvoření tohoto databázového modelu bylo spojit výhody relačního a objektového modelu, tj. stvořit takový model DB, který by měl relační základ, ale byl by obohacen o objektovou implementaci. První objektově-orientační rysy byly zadokumentovány ve stan-

dardu SQL-1999. Tento nový model měl ze začátku problém podpory objektových rozšíření u významných dodavatelů systémů řízení databází, ale velice rychle se situace začala měnit na opak [63].

Model dat objektově-relační databáze

V daném modelu to byl úspěšný, ale ne bezproblémový pokus o obohacení tabulek o objektovou orientaci. Změny nastaly i z hlediska datové struktury: relace začaly být obecnější (*nested relation model*). Perzistentní data se stále ukládaly do tabulek, ale jejich hodnoty dostaly možnost mít bohatší struktury (definovanou jako ADT - *abstract datatype*). Nyní tabulka již není normalizovaná, a v její sloupcích může být strukturovaná hodnota. ADT vede k zapouzdření dat a operací. Zavedla se obdoba OID, která umožňuje vytvářet nový typ vazeb mezi tabulkami. Dotazovacím jazykem nového modelu se stal SQL-1999 a výpočetní model měl formu navigace po tabulkách pomocí kurzoru a referencí [63].

3.5.3 Dokumentový model

Oblast databázových systémů se stále mění v důsledku reakcí na stále náročnější požadavky nejen odborné společnosti. S nástupem doby čísel došlo k velkému nárůstu analyzovatelných dat, které mají velký význam a hodnotu. Proto je důležitý výběr vhodné analýzy, která je schopna přinést cenné poznatky, kvalitnější interpretaci, prezentaci a zároveň možnost lepšího rozhodování.

Od moderních databází se pro rychlejší, spolehlivější a komfortnější práci s daty vyžaduje splnění celé řady požadavků [52]:

- decentralizace úložišť dat, úmyslná redundance pro odolnost proti výpadkům a rychlost, možnost zpracování velkých objemů dat a velkého množství operací /big data/, atd.
- možnost pracovat s novými problematickými datovými typy (údaje klíč-hodnota, objekty, nestrukturované dokumenty, RDF grafy, atp.)
- možnost iterativního vývoje (časté změny schématu databáze nebo dokonce žádné schéma, různé/nejasné způsoby použití databáze, atp.)
- vysoké požadavky na škálovatelnost (mobilní zařízení jako klienti i úložiště/poskytovatelé dat, nerovnoměrné rozložení zátěže prostorově i časově, specifické požadavky na dostupnost, na předem neznámé dotazy nelze optimalizovat indexy, atp.)

Odpověď na výše popsané požadavky byla snaha přizpůsobit široce používané relační databáze moderním požadavkům, ale velmi brzy se zjistilo, že v následku prováděných změn se začínají ztrácet rysy relačního konceptu a snaha o jejich dodržovaní nevhodně omezuje práci s databází (úmyslné zanedbávání *ACID*). To však vedlo ke vzniku specializovaných nerelačních (post-relačních, ne-relačních (NoSQL)) databází, které byly vhodnější pro specificky strukturovaná, uložená a přistupovaná data [52].

Databáze dokumentů, nazývaná také úložiště dokumentů nebo dokumentově orientovaná databáze, slouží k ukládání, získávání a správě polostrukturovaných dat (různých druhů dokumentů). Dokumentová databáze má dokumentový model databáze a je podmnožinou typu databáze NoSQL. Níže je uveden stručný popis historie a funkcionality NoSQL.

NoSQL databázový model a jeho rysy

NoSQL (ze začátku „not SQL“ a později „not only SQL“) měldatabázový model podle jeho autora Carla Strozzi umožnit rychlé a efektivní zpracování kolekcí dat se zaměřením na výkon, spolehlivost a agilnost [43]. První NoSQL databáze byly úplným opakem relačních databází. Byly zaměřeny na dostupnost s co nejkratší odezvou, bez možnosti definovat strukturu dat, bez podpory vztahů jednotlivých záznamů, bez možnosti provádět operace JOIN. Hlavní rysy NoSQL [52]:

- podporují nerelační datový model (základní typy NoSQL databází: klíč-hodnota, dokumentové, sloupcové, grafové, atd.);
- podporují distribuovanou architekturu (jsou velmi silné v distribuovanosti);
- mají různý přístup k práci s daty a jejich dotazování (většina NoSQL jsou open-source);
- NoSQL většinou řeší CAP (Consistency, Availability, Partition Tolerance neboli Brewerův teorém [54]) omezením konzistence dat (přístup *BASE*). Teorém říká, že u sdílených systémů je možné uspokojit maximálně 2 ze 3 požadavků v jednom momentě.

Relační databáze využívá transakce s důrazem na ACID, NoSQL využívá přístup BASE (vysvětleno níže). BASE není tak striktní, a dovoluje dočasnou nekonzistenci dat pro zvýšení dostupnosti a výkonu, projevuje se rychlejší odezvou i když cenou dočasné nekonzistence v uložených datech, která se řeší při čtení (např. verzování, nevalidní cache), při zápisu (např. distribuce změn), nebo asynchronně (např. replikace dat) [52]. Akronym BASE se „dekóduje“ jako:

- **Basically Available** – aplikace funguje v podstatě po celou dobu;
- **Soft-State** – uložená data nemusí být po celou dobu konzistentní;
- **Eventual consistency** – nakonec i tak bude databáze v určitém nám známém stavu.

V dnešní době se dá narazit na prolínání těchto dvou přístupů u některých databází (některým RDB je možné nastavovat úroveň „izolovanosti“ transakce, některé NoSQL databáze umožňují transakce s vlastnostmi ACID a definice vztahů).

NoSQL dokumentové databáze

Dokumentové databáze používají dokumenty jako strukturu pro ukládání a dotazování. V tomto případě termín „dokument“ může odkazovat na dokument Microsoft Word nebo PDF, ale to není nutné, a obvykle je to blok XML nebo JSON 3.11. Namísto sloupců s názvy a typy dat, které se používají v relační databázi, obsahuje dokument popis datového typu a hodnotu tohoto popisu. Každý dokument může mít stejnou nebo odlišnou strukturu. V případě, že je potřeba do databáze dokumentů přidat další typy dat, není potřeba upravovat celé databázový schéma, jako tomu bylo u relačních databází. Data lze jednoduše přidávat přidáním objektů do databáze [55].

Dokumenty jsou seskupeny do kolekcí, které slouží podobnému účelu jako relační tabulka. Databáze dokumentů poskytuje dotazovací mechanismus pro vyhledávání kolekce dokumentů s určitými atributy.

Obrázek 3.11: Ukázka dokumentového modelu určité sociální sítě.

EDM systémy založené na dokumentovém modelu

Jedním z EDM systémů založeném na NoSQL, využívajícího dokumentový model DB, je aplikace *IBM Notes* (známá jako Lotus Notes).

Výhody a nevýhody dokumentového modelu

Databáze dokumentového modelu nabízí důležité výhody, pokud jsou požadovány, včetně vlastnosti [52, 55, 56]:

- Flexibilní modelování dat: vzhledem k tomu, že aplikace založené na webových, mobilních, společenských a internetových aplikacích mění povahu aplikačních datových modelů, dokumentové databáze eliminují potřebu vynucení relačních datových modelů a podporují nové typy aplikačních datových modelů.
- Rychlý zápis: na rozdíl od tradičních relačních databází některé dokumentové databáze upřednostňují dostupnost zápisu přes přísnou datovou konzistencí. To zajišťuje, že zápis bude vždy rychlý, i v případě, že selhání některé části hardwaru nebo sítě povede k malému zpoždění v replikaci dat a konzistenci v prostředí.
- Rychlé provedení dotazovaní: mnohé dokumentové databáze mají výkonné vyhledávací nástroje a funkce indexování, které poskytují rychlé a účinné možnosti dotazování.
- Distribuovaný systém: aplikace může být umístěna na více než jednom serveru, což umožňuje snadné vyjmutí dílčích částí archivu, snadnou zálohovatelnost, přenositelnost, a využívání této části bez nebo s omezeným síťovým připojením.
- Dobrá škálovatelnost: je možné téměř neomezené horizontální škálování a zachování lineárního růstu výkonu. Jelikož dochází k prudkému nárůstu dat, která je potřeba ukládat, analyzovat a zpracovat, je tato vlastnost velmi důležitá.

Byl zde představen NoSQL dokumentový databázový model s popisy jeho základů, funkcionality a výhod. S jistotou se dá říci, že NoSQL databáze zatím nedokáží plnohodnotně nahradit relační databáze nebo jiné, ale nabízí mnoho nových směrů pro práci s daty cestou spojení (prolínání) vlastností několika různých druhů datových modelů při návrhu aplikací.

Kapitola 4

Specifikace požadavků a prvotní návrh systému pro správu dokumentů

Specifikace požadavků je jedním z klíčových bodů při práci s libovolným projektem. Proto v této kapitole budou stručně a obecně popsány specifikace požadavků, a spolu s tím i návrh na funkcionality vlastního elektronického systému na správu souborů. Další důležitou částí při tvorbě výsledného produktu je i jeho návrh, který zde bude zobrazen pomocí mock-upů a diagramů. Také bude v této kapitole nastíněn způsob rozšíření použitelnosti výsledného systému na další oblasti projektového řízení.

4.1 Obecná specifikace požadavků na vlastní EDM systém

Na základě teorie z oblasti elektronické správy dokumentů a také analýzy vybraných existujících řešení jsme specifikovali a navrhli rozsah funkcí, které by měl splňovat moderní a konkurenceschopný elektronický systém na správu dokumentů. Vlastní EDM systém bude mít formu webové aplikace. Na základě nastudované teorie a vlastního výběru jsem se rozhodla, že jádrem navrhovaného EDM systému bude **dokumentový model** databáze, tj. jeden z představitelů NoSQL databází. Níže je prvotní návrh funkcí, který bych chtěla maximálně splnit ve vlastním EDM.

Všechny současné systémy pro správu dokumentů nabízejí celou řadu možností **vkládaní dokumentů** do EDM systému. Systém podle vlastního návrhu by měl splňovat nejrozšířenější z nich, jako jsou: vkládaní dokumentů prostřednictvím hromadného importu dokumentů, a to různých typů (PDF, Microsoft Word, ODT, naskenovaný dokument). Také by měl systém umožňovat nahrávaní příloh (obrázků, atd.) do samotných dokumentů a vytváření nových dokumentů s možností pokročilého formátovaní. Celá aplikace by měla fungovat interaktivně: všechny změny by se zobrazovaly ostatním uživatelům okamžitě bez nutnosti obnovení stránky. Na základě menšího průzkumu ve známých firmách jsem zjistila, že uživatelé podobných systémů by měli rádi na výběr možnost převodu nahraných dokumentů (v případě naskenovaného dokumentu, PDF, atd.) do textového formátu pomocí technologie **OCR**[1] nebo případně kombinace zachování původního a převedeného dokumentu.

[1] **OCR** – optické rozpoznávání znaků, což je metoda, která pomocí scanneru umožňuje digitalizaci tištěných textů, s nimiž pak lze pracovat jako s normálním počítačovým textem.

Při práci s dokumenty ve firmě je velice důležitá **komunikace a kolaborace**. Proto jsem navrhla, že můj systém by měl poskytovat možnost spolupráce na dokumentech, ačkoliv zatím není jasné, zda současnou či ne. Toto se upřesní během implementace kvůli potřebě ověření technologických možností vybraných nástrojů. Komunikace bude vyřešena prostřednictvím komentování dokumentů nebo jeho části (připomínky). Pokud to technologické možnosti vybraných nástrojů a čas dovolí, tak vlastní EDM systém bude poskytovat pro podporu komunikace integrovaný chat u dokumentu, a také **drag&drop**[2] je to funkcionalita pro pohodlné přesouvání souborů pro podporu uživatelské přívětivosti.

Pro řízení by měl systém k vytvořeným dokumentům mít možnost vytváření úkolů, jejich přiřazení členem týmu a nastavení termínu dokončení.

Metadata jsou nedílnou součástí každého dokumentu. Slouží k přidání dodatečné informace k dokumentům a jejich revizím, která následně bude využívaná jinými funkci EDM systému. Ve vlastním systému plánuji využít metadata z pohledu jejich automatického generování nebo i možnosti správy pro následné jednodušší vyhledávaní nebo filtrování dokumentů.

Vyhledávaní nebo **filtrování** dokumentů je nezbytnou součástí libovolného elektronického systému pro správu dokumentů, proto jsem pro svůj systém také navrhla vyhledávaní, a to na základě dostupných metadat, tak i na základě fulltextového vyhledávaní v klasických dokumentech typu PDF, Word a jiných typů – naskenovaných dokumentů (pomocí využití OCR).

Automatizace procesů neboli **workflow** podporuje procesy oběhu dokumentace: automatizace procesů, které umožní uživatelům vidět pohromadě „dokumenty k vyřízeni“ (přezkumy, schválení, archivování a uchovávání dokumentů). Také automatizované procesy umožňují předávat dokumenty, informace či úkoly mezi uživatele. Navrhla jsem, aby můj EDM v rámci workflow umožňoval užitečnou kategorizaci dokumentů podle aktuálního stavu (nový, schválený, potřebné schválení, atd.), a také upozorňoval na změnu personál, který má právo a měl by dokument přesunout do stavu jiného (např. do stavu schválený). Při nahrání dokumentu pracovník označí posuzovatele, který ihned dostane upozornění o nutnosti zrevidovat dokument. Dokument, který byl označen za schválený, bude mít zablokovanou editaci pro všechny členy, kteří na něm spolupracovali, až na „nejvyššího“ uživatele. Jako jednu z variant automatizace v projektovém řízení jsem navrhla, aby můj systém při vytvoření události v kalendáři s názvem Porada nebo Meeting, automaticky vytvořil a uložil do systému dokument s tímto názvem a datem. Do daného dokumentu se následně zapíšou výsledky události přidělenou osobou.

Kontrola a správa verzí představuje nutnost pro správu dokumentů zpřístupněním historie samotného dokumentu. Po každém uložení dokumentu se při změně vytvoří nová revize bez nutnosti manuálního vytváření kopií nebo přejmenovávaní. Revize se bude číslovat od počátečního čísla 1 a inkrementálně s každou revizí zvětšovat. Uživatel bude mít taktéž možnost vrátit dokument do stavu zvolené revize, nebo porovnat aktuální verzi s verzí vybranou.

Dostupné dokumenty by v navrhovaném systému měly být vždy aktuální. Při jakékoliv změně u jednoho uživatele by změny měly být reflektovány u ostatních uživatelů bez nutnosti manuálně obnovit stránku.

Dokumenty by se měly dát **sdílet** formou odkazů nebo přímo uživatelské skupině či vybranému jednotlivci. Druhá strana by měla obdržet email nebo upozornění. Uživatel by

[2] **Drag and drop** *(táhni a pusť)* je operace používaná v grafickém uživatelském rozhraní, kdy uživatel „uchopí“ virtuální objekt (např. soubor nebo ikonu) a přesune ho „přetažením“ na jiné místo (nebo na jiný objekt).

měl mít přístup ke všem upozorněním z notifikačního panelu. Nutností je taktéž hromadný export vybraných dokumentů (původní formát nebo vlastní). Uživatel, který dokument vytvořil nebo upravil, by si mohl zapnout funkci přijímání upozornění o přečtení.

Přístupová práva zabezpečuje omezení přístupu neautorizovaných uživatelů k datům systému, čímž plní i bezpečnostní funkce pro EDM. Každý takový systém poskytuje alespoň základní přidělovaní práv, a můj systém nebude výjimkou. Moje řešení tedy bude rozdělovat uživatele do čtyř kategorií nebo rolí, a dělit je na tyto úrovně oprávnění: čitatel – může pouze zobrazit a stáhnout obsah, přispěvatel – může číst, upravovat, nahrávat nové dokumenty, přejmenovat, přispívat k diskusím atd. (obvykle se omezuje na obsah vytvořený vlastními silami), pokročilý uživatel – může aktualizovat / upravovat vlastnosti přidané ostatními uživateli, správce – má některé administrativní možnosti pouze na webu. Umí spravovat celkové oprávnění, přidávat uživatele, mazat obsah, spravovat pravidla složek, zrušit úpravy uzamčené jiným uživatelem.

Integrace různých formátů je součástí práce s dokumenty a vypomáhá si použitím různých kancelářských aplikací pro ukládání, otevírání a modifikaci. Proto je vhodná integrace nejpopulárnějších formátů, jakými jsou například dokumenty z kancelářského nástroje Microsoft Word. Systém by měl dokázat soubory tohoto typu otevřít nebo zkonvertovat do editovatelného formátu. Taktéž se nabízí možnost integrace různých služeb, jako například odesílaní mailů přímo ze systému.

Obecná specifikace požadavků v oblastech projektového řízení

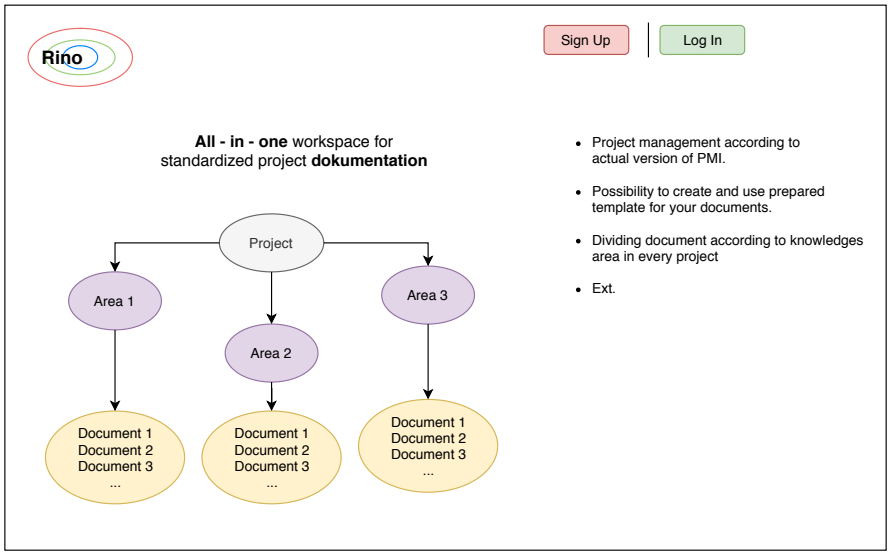
Co se týká požadavků v oblastech projektového řízení, systém by měl poskytovat možnost vytváření projektu, který bude obsahovat předpřipravené šablony jednotlivých dokumentů a jejich etap pro zvolené oblasti projektového řízení. V případě, že bychom chtěli, aby byl systém rozšiřitelný o další oblasti, bylo by umožněno importování nebo vytvoření dokumentových šablon, které by rozšířily systém o nové typy dokumentů a procesů.

4.2 Prvotní návrh vzhledu výsledného programu

Ve výrobě a designu je maketa, nebo mock-up, měřítko či full-size model designu nebo zařízení, užitý na výuku, demonstraci, hodnocení návrhu, propagace, a jiné účely. Maketa je prototyp, pokud poskytuje nebo znázorňuje alespoň část funkčnosti systému, a umožňuje testování návrhu [59]. Jak bylo uvedeno výše, mock-upy se široce používají v nejrůznějších sférách lidského života: od „náčrtu“ využití místa na pozemku či v bytě, po designování ve strojírenství či softwarového inženýrství. V této práci termín „mock-up“ bude chápán spíše jako grafický návrh pro určitou aplikaci nebo program. Taková forma návrhu napomáhá především k získání obecné představy vzhledu výsledného programu, zpětné vazby od uživatelů, možnostem jeho tvůrců zhodnotit vlastní návrh a opravit jeho nedostatky při implementaci. Z těchto důvodů bylo vyřešeno, že praktický návrh výsledného programu začne mock-upy, které pokrývají základní funkce programu ze specifikovaných požadavků a jsou představeny níže.

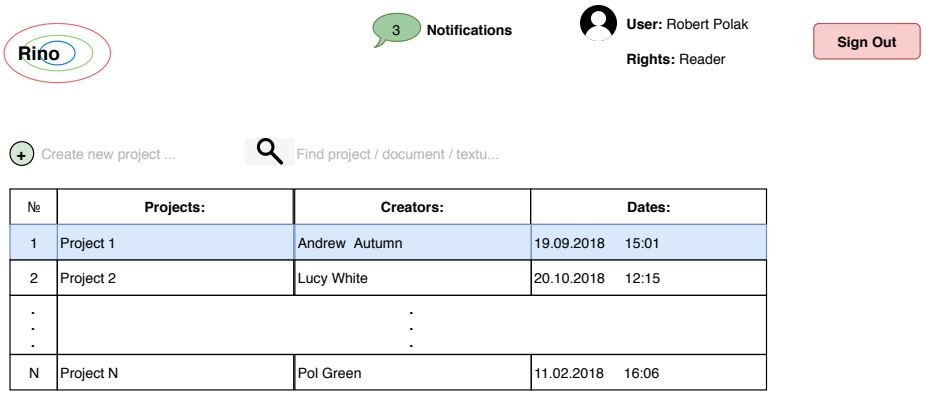
4.2.1 Úvodní stránka, přihlašovaní a registrace

Na obrázku číslo 4.1 je zobrazen mock-up úvodní stránky, kterou uvidí uživatel, pokud bude chtít použít naši webovou aplikaci. Proto představu, k čemu se aplikace používá, je dobré mít krátký popis, který stručně popisuje účel a možnosti daného nástroje. Odsud by



Obrázek 4.1: Návrh úvodní „stránky“ aplikace.

se uživatel mohl přihlásit do aplikace nebo si založit vlastní účet. Po kliknutí na jedno z tlačítek „Sign Up“ nebo „Log In“ by se měla zobrazit další stránka (obrázek 4.2), která by umožňovala vybranou akci dokončit.



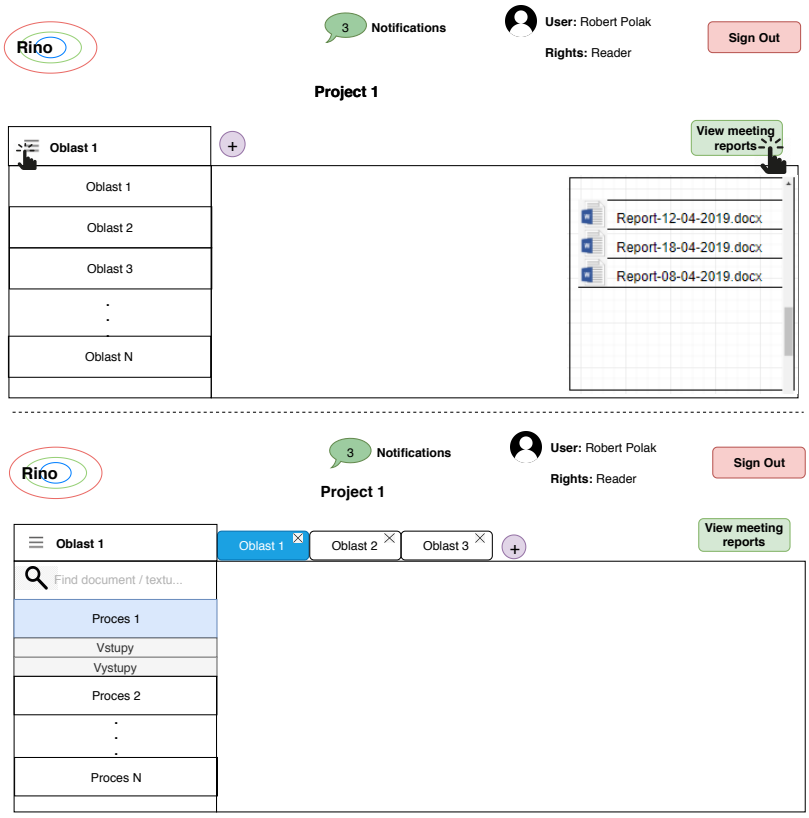
№	Projects:	Creators:	Dates:
1	Project 1	Andrew Autumn	19.09.2018 15:01
2	Project 2	Lucy White	20.10.2018 12:15
. . .	. . .		
N	Project N	Pol Green	11.02.2018 16:06

Obrázek 4.3: Vzhled stránky po přihlášení.

4.2.2 Vytvoření projektů, výběr oblasti a procesu

Na obrázku 4.3 je představen mock-up stránky, kterou uživatel uvidí ihned po přihlášení. Každý nový uživatel má po registraci právo jen na čtení některých volně dostupných dokumentů. V případě, že je uživatel pozván ke společné práci s některými z nich, vedle ikonky uživatele s jeho jménem se zobrazí číslo upozornění, které reportují o provedení změn v „sledovaných“ dokumentech.

Obrázek 4.2: Vzhled přihlašovaní a registrace uživatele v aplikaci.



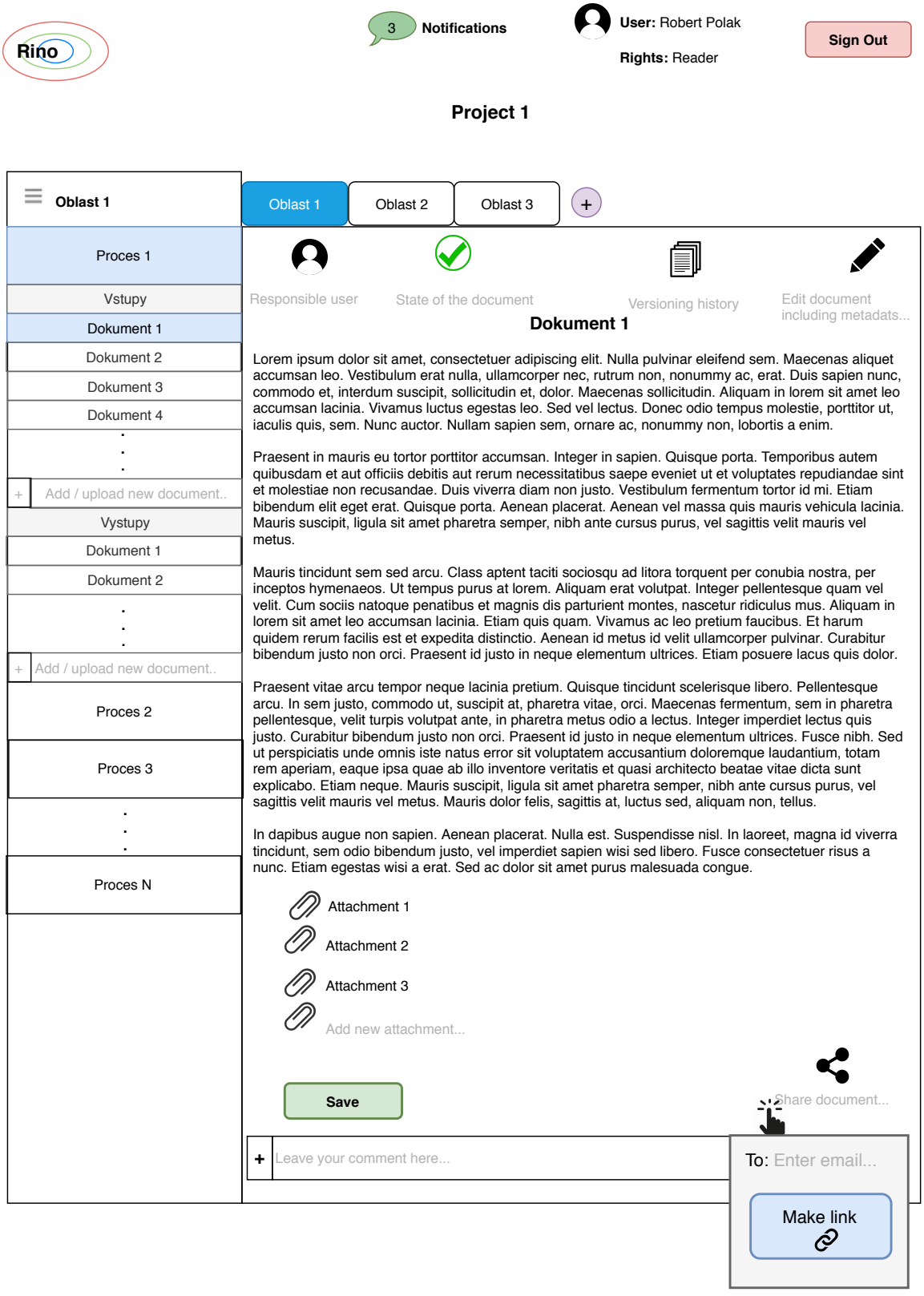
Obrázek 4.4: Znázornění možností výběru znalostní oblasti či procesu.

Také tato stránka nabízí uživateli možnost vytváření projektů nebo vyhledávaní potřebných informací podle projektu, dokumentu či textu. Odsud uživatel může pokračovat výběrem jednoho z projektů pro další práci nebo se z jakéhokoliv místa v aplikaci z ní odhlásit.

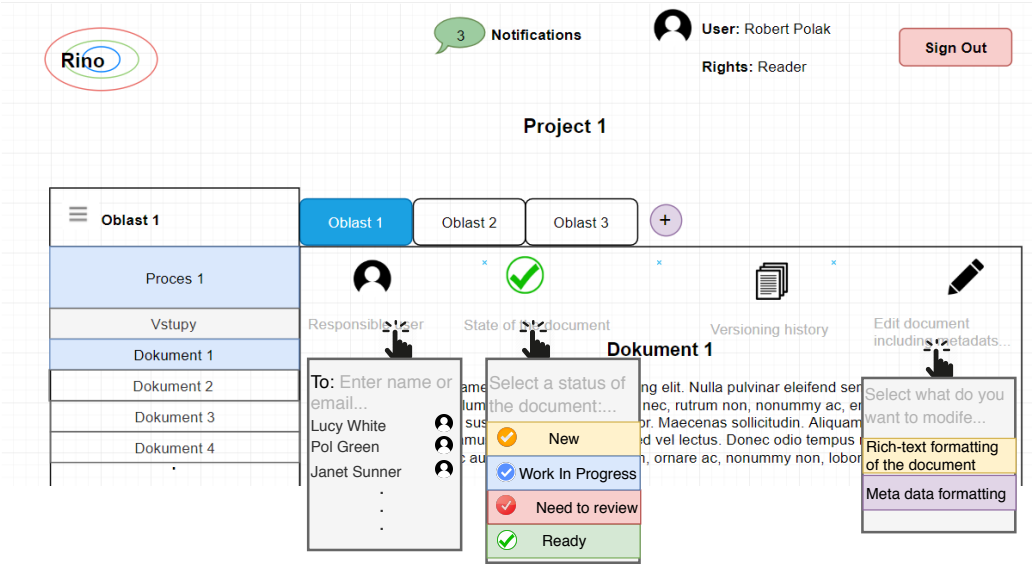
Další mock-up se skládá ze dvou částí (obrázek 4.4). První část se uživateli zobrazí po výběru projektu a umožní vybrat jednu ze znalostních oblastí projektu nebo symbolem „plus“ si vytvořit novou, respektive, vlastní znalostní oblast. Tímto způsobem by se vyřešila otázka rozšiřitelnosti systému pro správu dokumentů s ohledem na ostatní znalostní oblasti. Zelené tlačítko vpravo představuje možnost k prohlížení zpráv z porad k danému projektu. Nepochybně by pro uživatele bylo užitečné si v případě potřeby tyto zprávy v daném projektu moci prohlédnout. Druhá část mock-upu z obrázku 4.4 ukazuje na možnost výběru jakéhokoliv procesu z předem vybrané znalostní oblasti v projektu. Je tady také znázorněna možnost vyhledávání dokumentů v procesech, například podle textu, názvu, atd.

4.2.3 Obecný pohled na dokument a práci s ním

Mock-up na obrázku 4.5 představuje vzhled hlavní obrazovky systému. Tady má uživatel možnost vybrat, se kterými dokumenty v procesu chce pracovat (vstupníma/výstupníma), popřípadě s novými, které si nahraje do aplikace z počítače nebo vytvoří s/bez pomoci předem připravených šablon. Do daného dokumentu bude možnost nahrávat přílohy, sepisovat komentáře a také je ukládat, a dokument sdílet přes email nebo prostřednictvím odkazu.



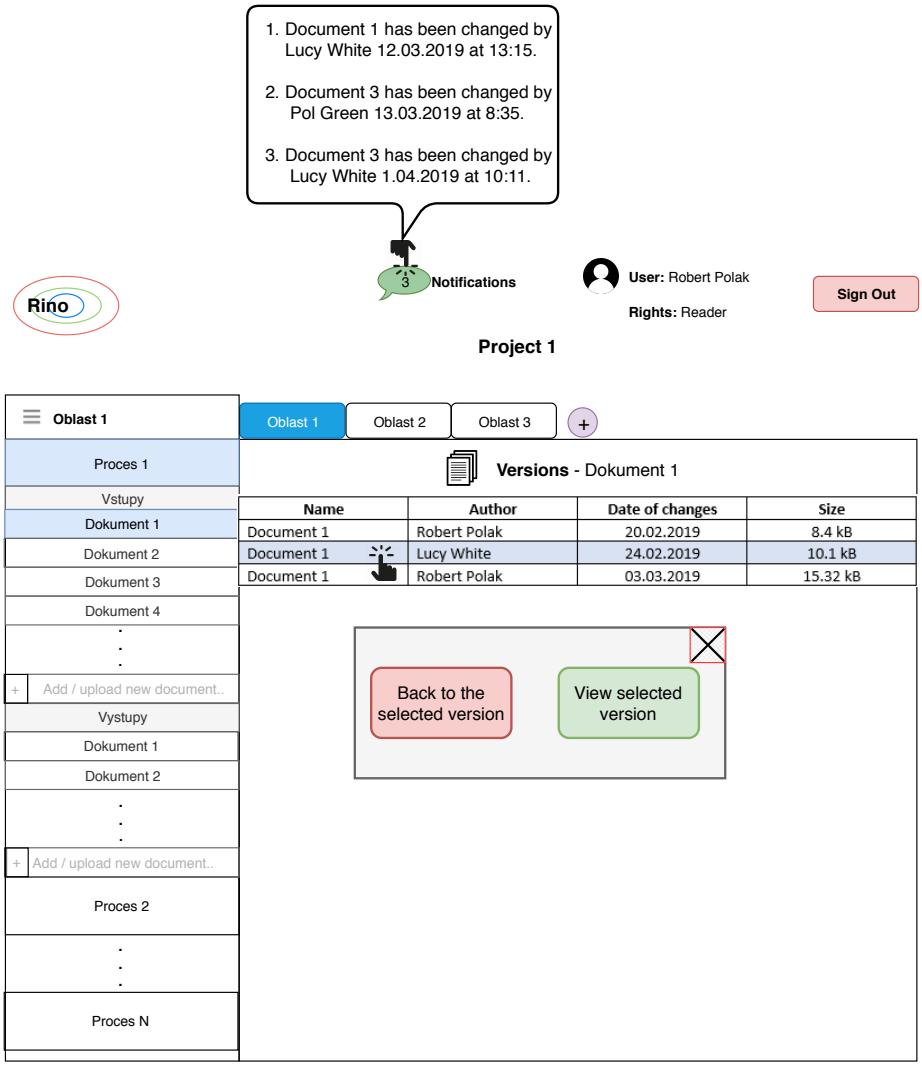
Obrázek 4.5: Znázornění obecného vzhledu na dokument v aplikaci a možnosti práce s ním.



Obrázek 4.6: Znázornění funkcionality vybraných prvků v textovém „editoru“.

Další mock-up (obrázek 4.6) detailněji zobrazuje horní část hlavní stránky aplikace. Tato část obsahuje tlačítka pro: výběr či přiřazení zodpovědného uživatele k dokumentu, výběru a přiřazení statusu dokumentu (Nový / Práce právě probíhá / Potřebuje schválení / Hotový), volbu editace dokumentu nebo jeho metadat, a také možnost si prohlédnout uložené verze daného dokumentu.

Obrázek 4.7 znázorňuje stránky s historií verzí, která se objeví po kliknutí na ikonku verze. Tato stránka zobrazuje informace o tom, kdy byl daný dokument uložen, kým pozměněn a jeho aktuální velikosti v okamžiku uložení. Také představuje možnost vybrat si jeden ze záznamů verzí a za předpokladu dostatečné úrovně práv uživatele vrátit dokument do podoby jedné z těchto verzí. Dále je z daného moc-upu vidět přibližně, jakou podobu mají upozornění, pokud se klikne na ikonku notifikace.



Obrázek 4.7: Znázornění verzí a notifikace o změně stavu dokumentu.

V této kapitole byl představen prvotní grafický návrh budoucího systému pro správu dokumentů prostřednictvím jednoduchých mock-upů. Při jejich tvorbě bylo cílem splnit co největší počet specifikovaných požadavků, což se vcelku podařilo. Výsledný vzhled a rozsah funkcí budou silně závislé a omezené možnostmi vybraných nástrojů a technologií pro jejich implementaci. Avšak i přes to jsou moc-upy důležitou částí procesu navrhování pro lepší orientaci programátora při implementování.

4.3 Návrh ERD a UCD diagramů

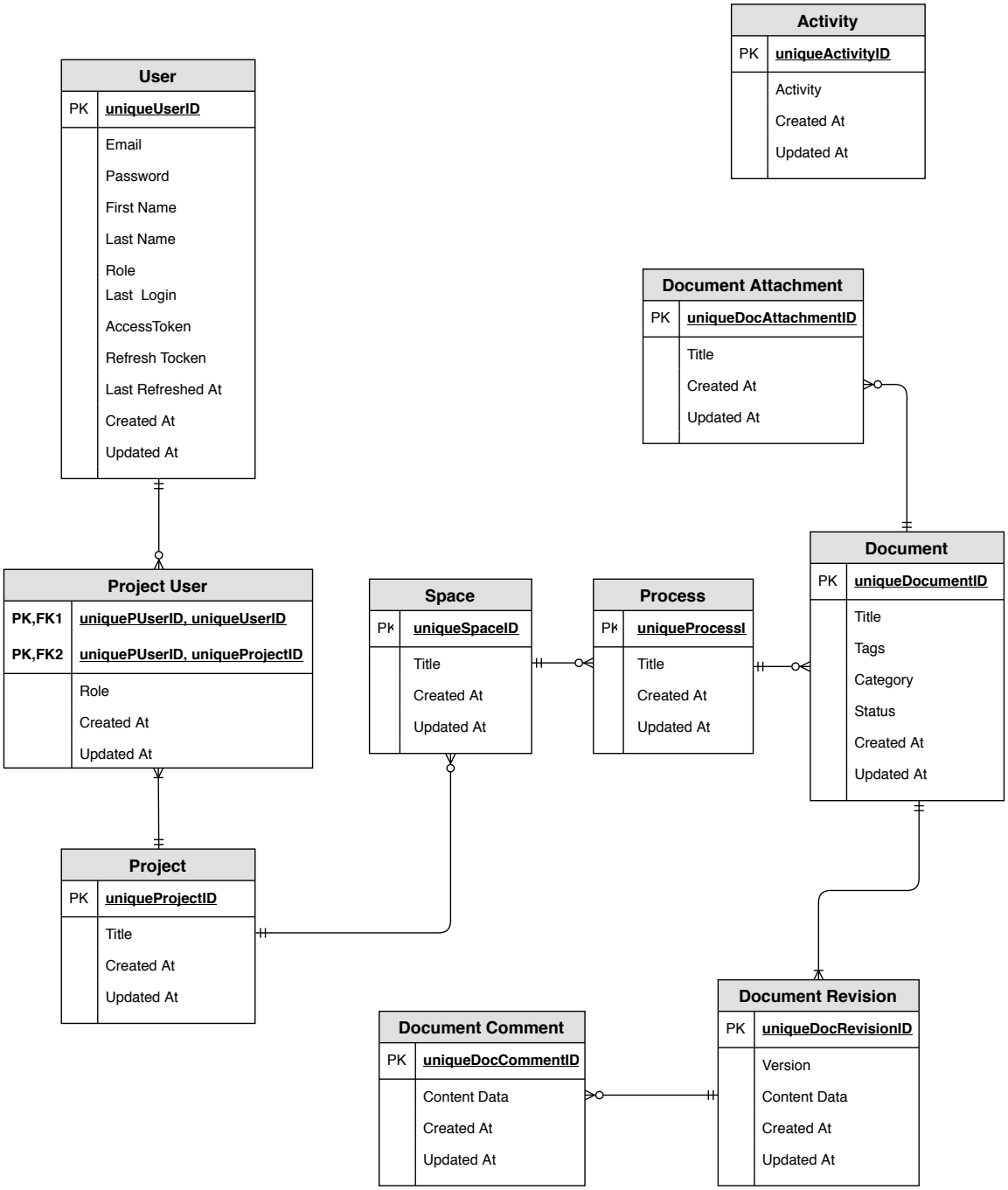
Diagramy při návrhu libovolného projektu hrají důležitou roli. Jejich komplexnost neustále roste, což je známý fakt. V reálném světě jsou již dávno minulostí doby, kdy celý program naprogramoval jeden člověk. Proto již nelze jen začít psát kód, ale je potřeba prvně systém navrhnout, a tím předejít možným budoucím problémům při jeho implementaci v týmu programátorů, protože jeden člověk na to většinou nebude stačit. Na začátku projektu je potřeba pružně reagovat na přibývající požadavky klienta a další problémy vyžadující pro-

vedení úprav. Proto nám diagramy mohou velmi pomoci v řešení problémů ve fázi analýzy, a také designu: během komunikace s klientem a zjišťování, k čemu programovat bude, a také během řešení otázky, jak se naprogramuje. Ve výsledku většina z nich poskytuje možnost představit si, jak bude výsledný program vypadat „z různých uhlů pohledu“ [65].

4.3.1 ERD diagram

Po delším uvažovaní bylo přijato řešení, že výsledný program bude fungovat na základě relační databáze a samotná práce s dokumenty bude na základě NoSQL dokumentové databáze. Proto bude v této části představen návrh relační databáze projektu, který můžete vidět na obrázku 4.8 ve formě entitně vztahového diagramu (z angl. *Entity Relationship Diagram (ERD)*).

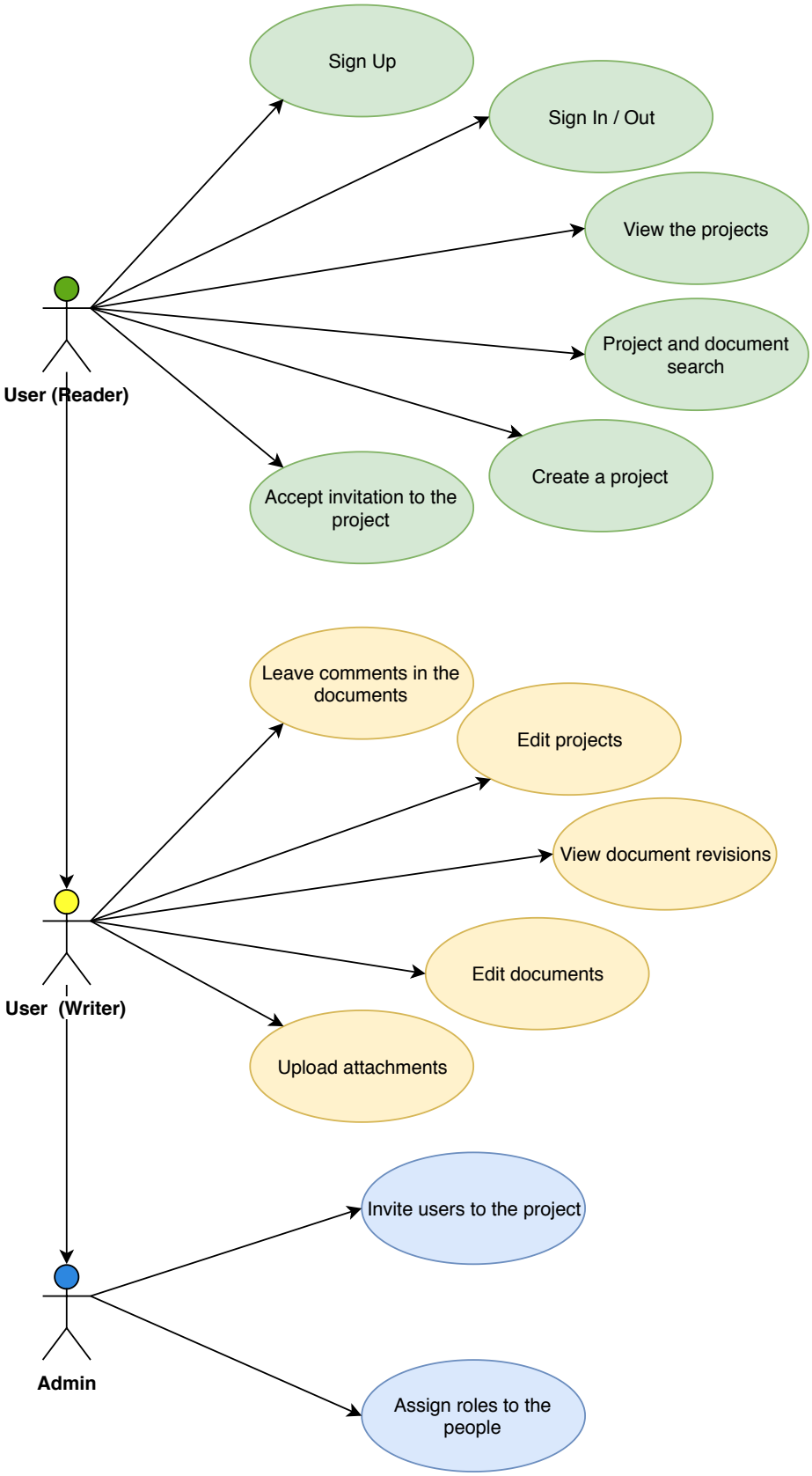
Tento diagram se skládá z deseti entit. Entita **User** představuje uživatele a ukládá si základní autentizační informace o každém uživateli, včetně jeho oprávnění (atribut *Role*) a informace o posledním přihlášení, data vytvoření atd. Entita **Project** obsahuje informace o tom, jaké jméno má projekt, kdy byl vytvořen a zaktualizován. Každý projekt je vytvořen uživatelem, který se u daného projektu stává vlastníkem s rolí administrátora. Uživatele bude možné pozvat do projektů s vybranou rolí, která jim nastaví jejich práva: čtenář, písař, administrátor. Každý uživatel může být pozván do více projektů a každý projekt může mít více pozvaných uživatelů, tedy mezi danými entitami je vazba M:M (*ManyToMany*) a vzniká potřeba vytvořit vazební tabulku. Tuto roli plní entita **ProjectUser**, která si ukládá informace o tom, kdo, kdy, a jak byl připsán ke každému projektu, a s jakým oprávněním. Entita **Space** představuje každou znalostní oblast, která je v rámci projektu unikátní. Entita **Process** představuje každý proces ve znalostních oblastech, který je v rámci každé z nich unikátní. **Document** si ukládá informace o dokumentech: název, datum vytvoření, stav, atd. Schéma dané entity se pravděpodobně změní v konečné variantě, v této chybí položka *Assignee*, která bude zodpovědná za možnost přiražení dokumentů k posouzení zodpovědné osobě a následné změny jeho stavu (například, z *Waiting for review* na *Ready*). Tyto změny se však provedou až při běhu implementačních prací. Entita **DocumentRevision** obsahuje informace provázané s verzemi dokumentů. Ve výsledné variantě se toto schéma entity trochu změní: atribut *ContentData* bude odstraněn až se obsah dokumentů bude ukládat do dokumentové databáze. Entita **DocumentComment** zodpovídá za komentáře, které uživatelé budou přidávat k dokumentům. Daná entita se také změní: zatím chybí uživatel, který komentář vytvořil. Entita **DocumentAttachment** bude ukládat informace o přílohách, které se budou nahrávat do dokumentů. Samotné přílohy se budou ukládat v rámci dokumentové databáze. Entita **Activity** je zodpovědná za zaznamenávaní aktivit.



Obrázek 4.8: Prvotní návrh ERD diagramu výsledné aplikace.

4.3.2 UCD diagram

Diagram případů užití (angl. *Use Case Diagram (UCD)* je v softwarovém inženýrství jeden z diagramů chování definovaných v grafickém jazyce UML (neboli *Unified Modeling Language*). Daný diagram pomáhá zjistit hranice systému a slouží jako základ pro odhad jeho rozsahu. Zachycuje především aktéry komunikující se systémem, vztahy mezi službami a jejími poskytovateli ve vizuální a textové podobě. Výsledek je srozumitelný pro zákazníky (budoucí uživatele) i pro vývojáře systému [37].



Obrázek 4.9: Návrh UCD diagramu výsledné aplikace.

Diagram představený na obrázku 4.9 UCD zobrazuje zúčastněné aktéry v systému rozdělené podle role. Prakticky vzato všichni aktéři představují jediného uživatele s různým stupněm oprávnění. Tak aktér **User (Reader)** s nejnižším stupněm oprávnění (čtenář) zastupuje každého nově zaregistrovaného uživatele, který zatím nemá přístup k úpravě projektů či dokumentů, ale může přijmout pozvání pro práci s některým z nich nebo si vytvořit vlastní projekt. Ve chvíli, kdy se takový uživatel rozhodne, tedy spíše mu bude nařízeno založit nový projekt, stane se z něj automaticky uživatel s oprávněním administrátora v tomto projektu. Jako administrátor si uživatel může pozvat do projektu jiné kolegy, kteří jsou zaregistrováni v systému, ale zatím nebyli do projektu přiraženi. Pro pozvání jiných kolegů do projektu aktér **Admin** posilá pozvánky, ve kterých předem definuje, jaké oprávnění bude mít konkrétní pozvaná osoba: čtenář, písař, administrátor. V bublinách napravo od aktérů jsou detailněji popsány služby, které mohou vykonávat: uživatel Čtenář má právo vykonávat pouze to, co má napsáno ve svých bublinách, zatímco uživatel Písař má právo dělat vše, co uživatel Čtenář plus to, co má obsaženo ve vlastních bublinách, a nakonec uživatel Administrátor má právo na libovolný úkon v rámci systému.

4.4 Návrh dokumentových šablon pro vybrané znalostní oblasti

Pro personál ve firmách, které jsou na začátku zavedení manažerských postupů a pokročilých přístupů k procesům řízení, by bylo velice nápomocné mít vše jasné, a pokud to jde, na jednom místě. Této myšlenky jsem se snažila držet při návrhu dokumentových šablon, které by uživatel mohl použít jako základ při tvorbě procesních dokumentů ze znalostní oblasti projektového řízení. Většina předem připravených šablon dokumentů obsahuje krátký popis toho, k čemu se tento dokument používá, co má obsahovat, a jak má vypadat. Zde je příklad využití z praxe s ukázkovými daty (jako u šablon uvedených v příloze A.1). V plném rozsahu vytvořené šablony zastupují znalostní oblasti řízení kvality, lidských zdrojů a komunikace. Mohou navíc sloužit jako základ pro rozšíření použitelnosti systému v dalších znalostních oblastech projektů tak, jako většina z nich představuje dokumenty, jež se využívají (například, dokument *Issue log* v řadě procesů různých oblastí).

Během návrhu šablon jsem nejprve nebyla spokojená s výsledkem, a absence znalostí z praxe mi překážela v provedení kvalitního návrhu. Rozhodla jsem se si zdroje pro inspiraci k dalšímu postupu v práci vyhledat na internetu. Některými z nich jsem se inspirovala, některé převzala a některé využila při vytváření výsledných šablon[3].

[3]Zdroje použité na návrh šablon: https://www.projectmanagementdocs.com/templates, https://www.projectmanager.com/templates, https://www.smartsheet.com, https://analysistabs.com, https://moorepants.github.io/eme185/pages/team-charter-template.html, https://www.tn.gov/content/dam/tn/finance/tbsm/TBSMOrganizationalProcessAssets.docx.

Kapitola 5

Implementace

Důležitou částí projektů, která se sleduje po návrhu, je implementace. Implementace daného projektu bude nejprve vysvětlena prostřednictvím popisu výsledné architektury aplikace z pohledu její rozdělení na jednotlivé části. Také tato kapitola zahrnuje popis použitých technologií z pohledu serverové a klientské části. Najdete zde i detailnější rozbor implementace zajímavých částí projektů. Pro implementaci celého projektu bylo zvoleno vývojové prostředí Visual Studio Code pro svou jednoduchost v používání, multifunkčnost a příjemné barevné schéma.

5.1 Základní rozdělení architektury

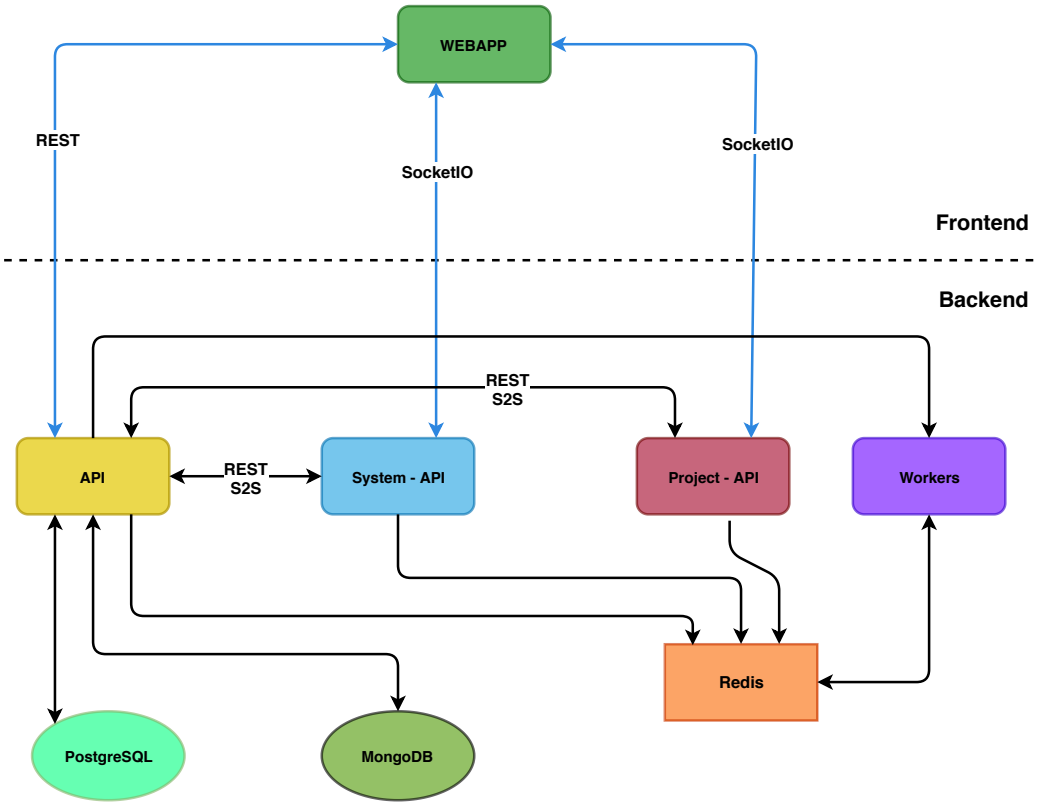
Z hlediska softvérové architektury se software může skládat z mnoha vrstev. Tyto vrstvy nemusí být na stejných strojích nebo přímo u uživatele, ale mohou být rozprostřeny. Popisovaná architektura se skládá ze dvou hlavních částí a funguje na principu *klient-server*. První částí je část, která se zobrazuje na straně webového klienta – **front-end** – a má na starosti prezentační vrstvu. Druhou částí je serverové řešení – **back-end** – poskytující soubor služeb komunikujících mezi sebou. Architektura je zobrazena na obrázku 5.1 a je detailněji popsána v následujících sekcích.

5.1.1 Popis serverové části (Backend)

Na nejspodnější vrstvě se nacházejí databázové systémy, které umožňují perzistenci uživatelských dat. Různé serverové moduly komunikují s rozličnými databázovými systémy. Na ukládání strukturovaných dat se využívá *objektovo-relační* databázový systém **PostgreSQL** popsaný v sekci 5.3.1. Mezi ukládaná data patří například údaje o uživatelích, o projektech, různých právech nebo rolích. Na ukládání dat o dokumentech, samotné dokumenty, přílohy nebo obecně objemnější data, se používá dokumentový *NoSQL* databázový systém **MongoDB**.

Systém dále obsahuje čtyři samostatné komponenty, kde každá plní specifickou roli. Komponenty mezi sebou komunikují dle potřeby. Díky této architektuře je s určitými změnami možné dosáhnout v budoucnu velké škálovatelnosti celého systému. Pak je velmi jednoduché například přesunout každý komponent zvlášť na samostatný server nebo mít *pool* serverů, ze kterých se pak vybírá dle vytíženosti.

Komponenty jsou modulární, to znamená, že odpovědnosti aplikace (jako je například správa uživatelů) jsou od sebe odděleny tak, aby je bylo možné využívat nezávisle. Nově vy-



Obrázek 5.1: Zobrazení základního rozdělení architektury a toku dat.

tvářené moduly se takto dokáží jednoduše zapojit do stávající architektury, přičemž mohou být používány jinými částmi nebo je používat.

Datové aplikační rozhraní (API)

Vrstva `API` slouží jako komponent, který má jako jediný přístup k databázovým systémům. Komponenta je bezestavová (*stateless*). *Stateless* komponenty jsou specifické tím, že si neuchovávají stav konverzace mezi klientem a komponentem samotným. Po odbavení požadavku od jednoho klienta mohou být okamžitě přiděleny na odbavení požadavků klienta jiného.

Jelikož identita bezestavového objektu není klientovi dostupná na ověření, ten neví, zda jeho požadavky obsluhuje stále tentýž session objekt[1] nebo nikoliv. Anonymita bezestavových session objektů způsobuje hlavní rozdíl oproti stavovým (*stateful*). Všechny instance konkrétní třídy konkrétního session objektu jsou identické, což neplatí pro *stateful* instance. To představuje výhodu, kdy se dá daný komponent škálovat horizontálně, to znamená přidávat identické instance na více serverů v případě potřeby.

Komunikace s ostatními moduly probíhá prostřednictvím **REST**[2] rozhraní. Rozhraní REST je používané pro jednotný a snadný přístup ke zdrojům. Zdrojem mohou být data,

[1] **Session object** neboli relace, uchovává informace o relaci uživatele nebo mění nastavení relace uživatele.

[2] **Representational state transfer** (REST) - https://www.service-architecture.com/articles/web-services/representational_state_transfer_rest.html

jako například konkrétní údaje o uživateli. Všechny zdroje mají vlastní identifikátor **URI**[3] a REST definuje čtyři základní metody pro přístup k nim (*GET*, *POST*, *DELETE*, *PUT*). Obsah dat v odpovědi je ve formátu **JSON** [4]

[3] **Uniform resource identifier** (URI) - Jednotný identifikátor prostředku https://aleph.nkp.cz/F/?func=direct&doc_number=000000567&local_base=KTD

[4] **JavaScript Object Notation** (JSON) je způsob zápisu dat nezávislý na počítačové platformě a určený pro přenos dat, která mohou být organizována v polích nebo agregována v objektech https://www.json.org/.

Popis	**Metoda**	**URI**
Dokumenty - document		
Vytvoření přílohy	post	/attachment/upload
Vytvoření komentáře	post	/comment/create
Vytvoření dokumentu	post	/create
Získání přiřazených dokumentů	get	/assigned
Získání dokumentu	get	/fetch/:documentId
Získání dokumentové revize	get	/revision/fetch/:revisionId
Procesy - process		
Vytvoření procesu	post	/create
Projekty - project		
Vytvoření projektu	post	/create
Získání všech a pozvaných uživatelů	get	/listInvitees/:projectId
Získání projektu	get	/fetch/:projectId
Získání projektů	get	/list
Pozvání uživatele do projektu	post	/user/invite
Vymazání uživatele z projektu	post	/user/remove
Veřejné API - public		
Přihlášení uživatele	post	/login
Server2server token	post	/s2stoken
Registrace uživatele	post	/signup
Obnovení uživatelského tokenu	post	/token
Server2Server - s2s		
Uložení dokumentu	post	/document/save
Získání revize	get	/revision/fetch/:revisionId
Získání dokumentů v projektu	get	/documents/:projectId
Získání uživatele	get	/user/profile/:userId
Aktualizace stavu dokumentu	post	/document/status
Aktualizace tagů dokumentu	post	/document/tags
Oblasti - space		
Vytvoření oblasti	post	/create
Uživatel - user		
Získání aktivit uživatele	get	/activities
Získání profilu uživatele	get	/profile
Získání uživatelů	get	/list

Tabulka 5.1: Přehled jednotlivých přístupových bodů pro **API** komponenty. Kompletní cesta ke zdroji se určuje jako kategorie + *URI*. Příklad kompletní cesty pro vytvoření komentáře: `document/comment/create`.

Obrázek 5.2: Detailnější zobrazení komponentu System-API.

Komunikace mezi serverovými komponenty dané architektury (komunikace `API <-> System-API` a `API <-> Project-API`) probíhá prostřednictvím **S2S** (*server-to-server*) rozhraní, které je implementováno jako šifrované *REST API* (HTTPS). Komunikace samotná je zabezpečena pomocí *OAuth 2.0* s využitím *Bearer tokenu* předávaném v autorizační hlavičce dotazu [33, 35]. `API` komponent si tento token validuje a v případě potřeby (expirace *auth tokenu*) si vygeneruje nový, případně zamítne dotaz (*neplatný token*). K těmto přístupovým bodům se díky autorizaci klient nedostane, a tak jsou citlivá data dostupná pouze pro serverové komponenty. Seznam přístupových bodů `API` je znázorněn v tabulce 5.1.

Komponent `API` se skládá z *Express* serveru, který má za úkol při startu aplikace zaregistrovat metody (*handlery*) pro obsluhu jednotlivých koncových bodů (*endpointů*), které slouží pro přístup ke zdrojům. Ve chvíli, kdy přijde dotaz na některý z koncových bodů, určený *handler* ho obslouží. Koncové body jsou registrovány dynamicky, kde při přidání nového adresáře a souboru v `API` komponentu serverový skript (`root.ts`) tuto souborovou strukturu projede a zaregistruje nové obslužné metody.

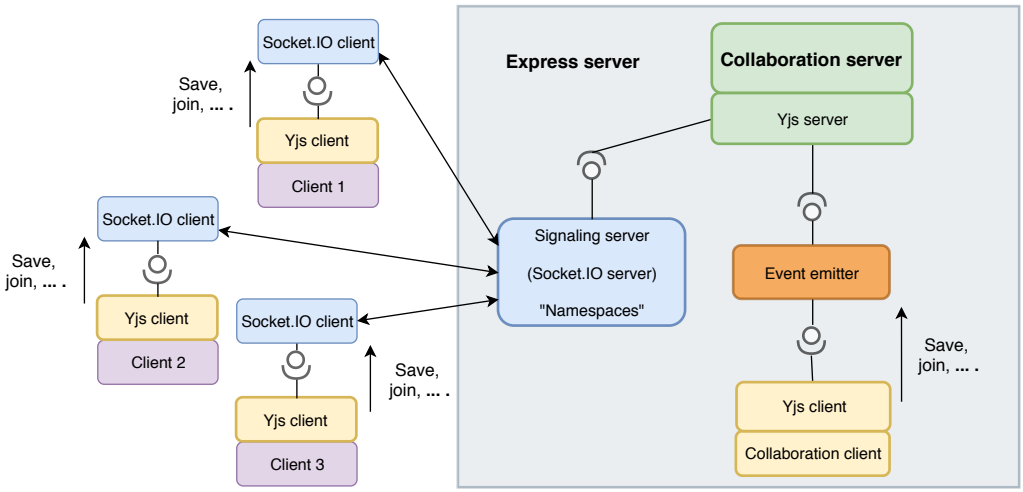
Komponent má propojení na obě databáze (*Postgres*, *MongoDB*). **Postgres** obstarává relační data a **MongoDB** dokumentová data, jako například přílohy nebo obsah dokumentů. Při zasílání příloh se například vytvoří záznam o dokumentu a samotná data dokumentu se vloží do **MongoDB**, která vrátí identifikátor. Ten se pak přiřadí do záznamu. Při stahování přílohy se dokument z **MongoDB** nahraje do veřejného dočasného adresáře a vygeneruje se veřejný odkaz, který se pak vrátí uživateli. V případě, že jde o obrázková data, webová aplikace zobrazí daný náhled obrázku.

Systémové aplikační rozhraní (System-API)

Daný komponent je stavový (**stateful**), z čehož vyplývá, že je pověřen tím, aby si pamatoval stav všech akcí vykonaných klientem mezi jednotlivými voláními metod. Stav je navázán na konkrétního klienta. Všechny metody budou vykonány v rámci jednoho sezení (*session*) a stavy proměnných jsou uchovány mezi jednotlivým voláním metod.

Komponent slouží pro výměnu systémových zpráv mezi klienty webové aplikace. Příkladem je například odstranění uživatele z projektu, nebo jestliže se uživateli změnila role. Díky tomuto řešení se o změně uživatel dozví okamžitě ve webové aplikaci. Pro implementaci tohoto řešení je použitá technologie **socketIO**, která je popsána v sekci 5.3.1.

Na obrázku 5.2 je představen interní pohled na komponent `System-API`. Daný komponent se skládá z *Express* serveru, který hostuje signalizační server *socketIO*. Na rozdíl od signalizačního serveru v `Project-API` daný server nepoužívá jmenné prostory (tzv. namespaces), proto všechno, co přijde na daný server, bude rozposíláno všem klientům. Používá se to v případě, kdy je potřeba poslat určitou zprávu všem uživatelům bez rozdílu toho,



Obrázek 5.3: Detailnější zobrazení prvku Projekt-API.

zda jsou / nejsou v konkrétním projektu. Jako příklad uvedeme následující situaci: uživatel 1 pozve dalšího uživatele do svého projektu. Zpráva o tomto se rozešle všem klientům prostřednictvím daného signalizačního serveru v `System-API`, neboť nevíme, kde se tento pozvaný uživatel nyní nachází.

Důvodem toho, proč v architektuře projektu komponenty `System-API` a `Project-Api` jsou naimplementovány jako dva rozdílné prvky se samostatnými signalizačními servery, je snaha o oddělení rolí, a to, že komponent `Project-API` je mnohem komplexnější z hlediska zodpovědností a funkcionality.

Projektové aplikační rozhraní (Project-API)

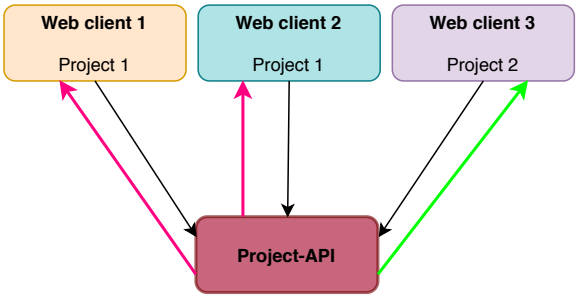
Modul `Project-API` představuje stavový komponent. Stavový komponent je navázán na konkrétní klienty. Pokud klient vytvoří instanci komponentu, všechny volané metody budou prováděny v rámci jedné a téže instance a stavy proměnných budou uchovávány mezi jednotlivými voláními, modul si tedy drží stav.

Komponent slouží taktéž jako signalizační server a pro komunikaci mezi klienty je použita technologie **socketIO** 5.3.1. Signalizace je určena pro výměnu zpráv mezi klienty v rámci daného projektu. To znamená, že ve chvíli, kdy klient začne pracovat s dokumenty v jednom projektu, se na serveru, kde běží komponent, vytvoří v paměti dočasný záznam (*snapshot*) aktuálního stavu (*Collaboration client*). Kdykoliv se do projektu připojí další klient, vidí vše ve stejném stavu jako předchozí klienti.

Přes tento signalizační server probíhá i komunikace v rámci samotného editoru. Cokoliv v editoru píše jeden klient, synchronizuje se i všem ostatním. Mezi další funkcionality, které komponent obstarává, patří například synchronizace obsahu, pozice kurzoru anebo zvýraznění textu.

Další zodpovědností komponentu je přidání nového komentáře, nahrání nebo odebrání přílohy, a rozšíření (*broadcastovat*) této změny přes signalizační server všem klientům ve stejném projektu. Aby se zpráva dostala správným klientům, využívá se v *socketIO* funkcionalitě namespaces, kde je každý projekt reprezentován svým jmenným prostorem.

Na obrázku 5.3 je komponent `Project-API` představen detailněji, včetně zobrazení interních komponent a interakcí mezi nimi pro lepší pochopení dané části aplikace, který



Obrázek 5.4: Schéma komunikace mezi uživateli projektů.

následuje níže. `Project-API` server se skládá z několika komponent. Celé to je *Express server*, který v sobě hostuje signalizační server implementovaný pomocí *socketIO*. Daný express server se liší od express serveru v komponentu `System-API` pouze tím, že používá *namespaces*, které reprezentují jednotlivé projekty.

Signalizační server má v sobě naimplementovány různé obslužné metody, které poslouchají příchozí události od webových klientů. Každá událost má své jméno, které vysvětluje podstatu příchozího signálu: *project-join*, *document-save*, *project-leave* a další. Ve chvíli, kdy z určitého socketIO klienta přijde událost, příslušná metoda, která je registrována na událost s tímto konkrétním názvem, danou událost obslouží. Na daném popisu je postavená logika, která je zodpovědná za další postup při odbavování požadavků typu: uživatel N se připojil do projektu M, uložil určitý dokument, atd.

Další komponenty v `Project-API` serveru jsou ty, které obsluhují kolaborační editor: `Collaboration server` a `Collaboration client`. První komponent implementuje `Yjs server`: na něj se přes socketIO připojují weboví klienti. Druhý komponent funguje jako plnohodnotný klient, stejně tak i weboví klienti. Je zde ale z toho důvodu, že právě tento `Collaboration client` se stará o nahrávání a ukládání dokumentů, a zabezpečuje plnou kontrolu nad daným procesem. Díky tomu, že v rámci `Project-API` běží další klient pro kolaborační editor (`Collaboration client`), máme jistotu, že tam zůstane v chodu, a tedy ve chvíli, kdybychom potřebovali řešit nahrávaní / uložení dokumentu na některém z webových klientů, stačí se obrátit na daný `Collaboration client`. Ten se o to postará tak, že vytáhne potřebné aktuální data z `Yjs serveru` a zapíše je do databáze nebo ve chvíli, kdy se určitý dokument nahrává, tak z databáze vytáhne obsah daného dokumentu a pošle jej do `Yjs serveru`. `Yjs server` se následně postára o to, aby se data dostaly ke všem klientům, kteří jsou připojeni na daný dokument. V případě, že bychom danou úlohu řešili přes webové klienty, neměli bychom možnost mít takovou kontrolu: každý uživatel může bez kontroly a kdykoliv opustit nebo zavřít prohlížeč, například v průběhu ukládání.

Tím, že `Collaboration server` a `Collaboration client` jsou hostovány v rámci jednoho procesu, je zbytečné, aby komunikovali přes *socketIO*, takže komunikace mezi nimi probíhá lokálně, kdy si posílají zprávy přes `Event emmiter`.

Na obrázku 5.4 je zobrazeno schéma komunikace mezi uživateli projektů. Pro její korektní průběh signalizační server komponenty `Project-API` využívá výhod jmenného prostoru, což bude vysvětleno dále na příkladu. První webový klient vstoupí do projektu *Project 1* a v ten okamžik se připojí na signalizační server komponenty `Project-API`, kde se vytvoří jmenný prostor neboli namespace *Project 1*. To způsobí, že se od toho okamžiku všechny události z daného webového klienta budou posílat do jmenného prostoru *Project 1*. `Project-API` server přitom ví, že cokoliv, co přijde do jmenného prostoru *Project 1*, má pře-

poslat všem ostatním klientům (růžové šipky na obrázku), které jsou připojeny do stejného namespacu (v našem případě je takovým klientem webový klient číslo 2 na obrázku 5.4). Když se webový klient číslo 3 připojí do projektu *Project 2*, tak se pro něj v signalizačním serveru komponenty `Project-API` vytvoří nový jmenný prostor *Project 2*. Takže zprávy, které přijdou od webových klientů 1 a 2 se třetímu posílat nebudou.

Yjs využívá na distribuci změn a slučování dat technologii **CRDT** [53]. CRDT je datová struktura, která může být replikována napříč více počítači v síti. Matematicky jsou při CRDT strukturách nesrovnalosti vždy řešitelné. Konkrétní implementaci řeší samotná knihovna *Yjs*, což umožňuje kolaboraci a integraci textových změn vícero klienty. Data, která se přenášejí, jsou popsány formátem, který je kompatibilní s CRDT. V tomto případě se jedná o Quill Delta, popsáno v sekci 5.3.2

Služby (Workers)

Pro granulárnější rozdělení architektury a rozdělení odpovědností mezi moduly vznikl nový komponent zodpovědný za různé obstarávání služeb. Tento komponent je vhodný pro zpracování asynchronních činností různého druhu.

Mezi možné služby (*workery*) můžeme zařadit například obstarávaní registračního procesu, tj. ověření emailové adresy. Další služba zařizuje procesy uživatelského managementu, jako například možnost resetování hesla.

Tento komponent je velmi univerzální a rozšiřitelný. *Workery* mohou být škálovatelné horizontálně, a pro zvýšení propustnosti je možné nasadit více strojů, na kterých by běžely stejné služby.

Mezi další služby, o které by se daly komponenty rozšířit, patří například export dokumentů do různých formátů, nahrávání dokumentů na `Google Drive`[5] a podobně. Systém je však natolik rozsáhlý, že tato konkrétní implementace není součástí aktuálního řešení, ale patří mezi potencionální možnosti dalšího vývoje.

Pomocný server pro cachování dat (Redis)

`Redis`[6] je open source úložiště dat, které své datové struktury ukládá kvůli rychlostním benefitům přímo do paměti *RAM*. Komponent slouží k urychlení některých procesů ostatních komponentů. Příkladem může být redukce dotazů do databáze, kterou používá komponent `API` (5.1.1). Další možností využití tohoto komponentu je sdílení dat mezi službami, ne všechno se však podařilo ve výsledné práci realizovat. Je tedy prostor v práci pokračovat.

5.1.2 Popis částí webového klienta (Frontend)

Frontend byl implementován jako *React* aplikace. Při načtení webové stránky se využívá `server-side rendering`, což znamená, že HTML obsah, CSS, rozložení uživatelského rozhraní a dat se vygeneruje s použitím šablon na straně serveru. Oproti `client-side renderingu` je při prvotním načtení tento přístup rychlejší, jelikož se ušetří počet dotazů na server.

Systém využívá návrhový vzor hloupých (*dumb*) a inteligentních (*smart*) komponent [30]. Inteligentní komponenty jsou komponenty na úrovni aplikace, které vykonávají funkce a spravují data, zatímco hloupé komponenty se zaměřují pouze na uživatelské rozhraní.

Výhody tohoto rozdělení:

[5] **Google Drive** - https://www.google.com/drive/
[6] **Redis** - https://redis.io/

- **Opakovatelnost**: ačkoliv můžeme podobný komponent napsat vícekrát na různých místech aplikace, dá se ušetřit mnoho času vytvořením opakovaně použitelného komponentu, který přijímá vstupy.
- **Flexibilita snadno provádět změny**: oddělení inteligentních a hloupých komponentů umožňuje flexibilitu při provádění malých i velkých změn v aplikaci bez velké práce. Při změně stačí komponent změnit jen na jednom místě a změna se projeví v celé aplikaci.
- **Přehlednost kódu**: čím méně kódu je a čím více je organizovanější, tím jednodušší je nejen ho pochopit pro autora samotného, ale také ostatními, kteří chtějí zjistit, jak aplikace funguje.
- **Poskytuje konzistenci a zabraňuje duplikaci**: díky rozdělení na *dumb* a *smart* existuje možnost umístit formulář pro přihlášení na mnoho stránek bez toho, aby se duplikoval kód.
- **Snadné testování**: hloupý komponent se velmi snadno testuje, jelikož jednoduše bere vstupy a vrací hotový kus UI.

Komunikace s backendem probíhá přes REST API a dotazy jsou zabezpečeny pomocí autorizačního tokenu. Systém je spojen s komponenty `System-API` a `Project-API` přes rozhraní *SocketIO*. Klient se ihned po přihlášení spojuje s komponenty `System-API`, aby byl schopen přijímat systémové zprávy, a v případě přizvání nového uživatele do projektu se vyvolá metoda, která tomuto uživateli stáhne nový projekt a webová aplikace se tak jeví interaktivně. Jestliže klient vstoupí do určitého projektu, je připojen ke komponentu `Project-API` aby byl např. v případě přidání nového dokumentu okamžitě aktualizován seznam dokumentů pro všechny klienty v daném projektu.

V React aplikaci má každý komponent určitý interní stav. Při změně stavu se komponent překreslí. Jestliže však chceme synchronizovat stav mnoha komponent, které sdílejí data, tak bez systému pro stavový manažment bychom museli komponenty složitě synchronizovat. Jak tedy zjistíme, v jakém stavu se nachází aplikace? Můžeme například zkontrolovat DOM a všechny komponenty. Se systémem pro správu stavu jednoduše zkontrolujeme námi definovanou datovou strukturu. DOM by měl odrážet data, ale samotná data slouží jako lepší zdroj (*single source of truth*[7]).

Uživatelské rozhraní webové aplikace

Pro vytvoření uživatelského rozhraní webové aplikace byl použit vývojový framework *Semantic UI React*. Výsledný vzhled aplikace vypadá jednoduše a určitě se na něm dá ještě pracovat a případně jej vylepšit. Avšak při tvorbě své aplikace jsem se zaměřovala spíše na funkčnost, než na vzhled. Také jsem toho názoru, že v jednoduchosti je krása, a toto pravidlo z pohledu uživatelského rozhraní má aplikace splnila.

Do vytvořené webové aplikace je možné se zaregistrovat jako nový uživatel, a také se přihlásit jako registrovaný uživatel (příloha B.1). Ihned po přihlášení je uživatel schopný zjistit, zda má přiřazené projekty: přiřazené projekty se objeví v seznamu levého sloupce aplikace (příloha B.2). V případě, že uživatel nemá přiřazeny žádné projekty (příloha B.3)

[7] **Jediný zdroj pravdy** (Single source of truth) - https://en.wikipedia.org/wiki/Single_source_of_truth

má oprávnění vytvořit vlastní projekt (příloha B.4). Každý nový projekt se vytváří na základě **projektové šablony**, díky čemuž se projekt vytváří s předem definovanou strukturou (příloha B.5). Daná struktura obsahuje tři předem vytvořené znalostní oblasti, na které byla daná práce zaměřena: řízení kvality, řízení lidských zdrojů a řízení komunikace. Zmíněné oblasti obsahují předem připravené rozdělení na procesy, samotné procesy a vstupní / výstupní dokumenty procesů. V případě potřeby uživatel může vytvořit další oblasti a v nich procesy i dokumenty (příloha B.6). Nové oblasti, procesy a dokumenty se pochopitelně vytvářejí prázdné. Do vlastních projektů (nebo do projektů, ve kterých má dostatečné oprávnění) uživatel může pozvat další uživatele, kterým předem nastaví oprávnění v projektu, do kterého je chce přihlásit, pomocí výběru rolí (příloha B.7). Uživateli, který byl pozván do určitého projektu, se tento projekt objeví v levém sloupci aplikace (příloha B.2), a tedy (po přihlášení) daný uživatel může kliknout na přiřazený projekt a začít / pokračovat v práci s dokumenty.

Po přihlášení uživatel vidí aplikace „rozdělenu“ do tří sloupců. Jak bylo zmíněno výše, levý sloupec zobrazuje případně přiřazené (neboli vlastní) projekty. Střední a pravý sloupec aplikace plní úlohu **dashboardu**. V pravém sloupci se zobrazují aktivity, které proběhly v projektech: uložení konkrétního dokumentu, přidání nového komentáře, přidání nové přílohy, změna stavu dokumentu, změna metadat dokumentu (tagů) (příloha B.8). Význam středního sloupce bude vysvětlen dále. Implementace a zobrazení aktivit na dashboardu je v podstatě zavedení prototypu notifikací, které jsou součástí řešení podpory automatizace projektových procesů (workflow) v mé EDM aplikaci.

Pro práci s některým dokumentem uživateli stačí si vybrat konkrétní oblast, proces, a následně si kliknout na dokument. Po rozkliknutí jednoho z předem vytvořených dokumentů se uživateli zobrazí **dokumentová šablona** konkrétního dokumentu (příloha B.9). Prozatím se dokumentové šablony, které jsou ve vytvořené aplikaci, liší od během procesu návrhu aplikace: obsahují pouze textová data, nikoliv tabulky. Je to bohužel dočasným následkem výběru „mladé“ a moderní technologie pro podporu textového editoru v aplikaci *Quill*, která má začít podporovat práci s tabulkami od následujícího vydání. Jakmile bude toto vydání zveřejněno, bude možné přidat navržené tabulky do dokumentových šablon. Kvůli tomu byly předpřipravené dokumentové šablony nahrány do odpovídajících procesů, jako vstupy / výstupy pouze částečně. Všechny dokumentové šablony však budou nahrány do obsahu paměťového média, kde si je bude možné prohlédnout. Použitá technologie *Quill* dovoluje jednoduše a intuitivně tvořit, modifikovat a mazat obsah dokumentů (příloha B.10).

Ve vytvořené aplikaci je umožněna **kolaborativní spolupráce** více uživatelů s dokumenty. Při práci s jedním dokumentem každý uživatel uvidí barevný kurzor jiných uživatelů s popisem jejich jména v případě navedení myši na vybraný kurzor (příloha B.11). Všechny změny, které se provádějí ostatními uživateli, jsou proveditelné v reálném čase a sledovatelné. V případě, že jeden z uživatelů dá dokument, na kterém spolupracuje, uložit, uloží se i změny, které byly provedeny jinými uživateli.

Díky možnosti si vytvořit vlastní oblasti, procesy, dokumenty, použití dokumentových šablon a kolaboračního editoru je tato aplikace rozšiřitelná pro správu dokumentů ostatních (sedmi) znalostních oblastí.

Aplikace také umožnuje **verzování dokumentů**. Verzi dokumentu uživatel vytvoří tak, že jej dá uložit. Zobrazit seznam verzí dokumentu je možné kliknutím na tlačítko *Revisions* na pravé straně aplikace (příloha B.12). V případě, že dokument od svého vytvoření nebyl uložen, v okénku *Revisions* bude pouze záznam nulové verze, což je vlastně původně vytvořený dokument. Po novém uložení dokumentu se ve *Revisions* objeví nový záznam:

Revision 1, který je aktuálně editovanou a uloženou verzí dokumentu. Pro zobrazení dokumentu vybrané verze si stačí kliknout na jednu z verzí v záznamu *Revision* (příloha B.13).

Další součástí řešení podpory automatizace projektových procesů (workflow) ve výsledné aplikaci jsou **stavy dokumentu**. Libovolný dokument se může nacházet ve třech stavech: *Ready, Waiting for review, Done* (příloha B.14). Implicitně se každý nově vytvořený dokument nachází ve stavu *Ready*, což znamená, že je připravený pro práci. Ve chvíli, kdy práce s dokumentem bude provedená, některý z uživatelů může požádat zodpovědnou osobu daný dokument zkontrolovat: „přesune“ dokument do stavu *Waiting for review* a přiřadí ho konkrétnímu uživateli na kontrolu (příloha B.15). Stav dokumentu je viditelný pro všechny uživatele, a tak každý vidí, v jakém stavu je. Uživatel, kterému byl dokument přiřazen, jej uvidí ve středním sloupci aplikace na svém dashbortu (příloha B.16). Poté, co uživatel odpovědný za revize zkontroluje dokument a případně jej opraví, může dát dokument do stavu *Done*. V tomto stavu je dokument uzamčený pro upravování. Kvůli tomu, že se v projektech je často třeba vrátit zpět, a případně něco přidat, upravit, je dokument možné navrátit do stavu *Waiting for review*, a tedy jej odemknout a provést úpravu.

Při týmové spolupráci na projektu, a tedy i s dokumenty, je užitečné používat **komentáře**. V aplikaci je implementována jednoduchá ukázka dané funkce, která umožnuje ke každému dokumentu v aplikaci přidávat komentáře, jež se synchronizují všem uživatelům. Komentář zahrnuje jméno komentátora, časové razítko a samotný obsah komentáře (příloha B.17).

K dokumentům v aplikaci se dají nahrávat **přílohy** (příloha B.18). Je to obzvlášť vhodné, jestliže je potřeba „zálohovat“ dokument, který vznikl na papíře, do aplikace. Zatím je zobrazení příloh v náhledu jen na úrovni obrázků (příloha B.19), otevřená knihovna *React-PDF*, která by to mohla zabezpečit, obsahuje bohužel chybu, kterou sama a v dohledné době nedokáži opravit.

Také v aplikaci funguje vyhledávání dokumentů v rámci projektů podle jména a tagů dokumentů (přílohy B.20, B.21).

5.2 ERD výsledné aplikace

ERD schéma výsledné aplikace se ve srovnání s navrhnutou verzí změnilo (obrázek 5.5). Změny proběhly v entitě **Document**: přibyl atribut *UserID*, který reprezentuje to, komu je dokument přiřazen na recenze. Entita **Document Revision** byla obohacena o atribut *UserID*, který v tomto případě odkazuje na uživatele, který danou revizi vytvořil. Atribut *UserID* přibyl také v entitě **Document Comment**, kdy odkazuje na uživatele, který komentář vytvořil. Do entity **Document Attachment** přibyl atribut *ContentID*, jenž odkazuje na databázi MongoDB, kde je dokument uložen.

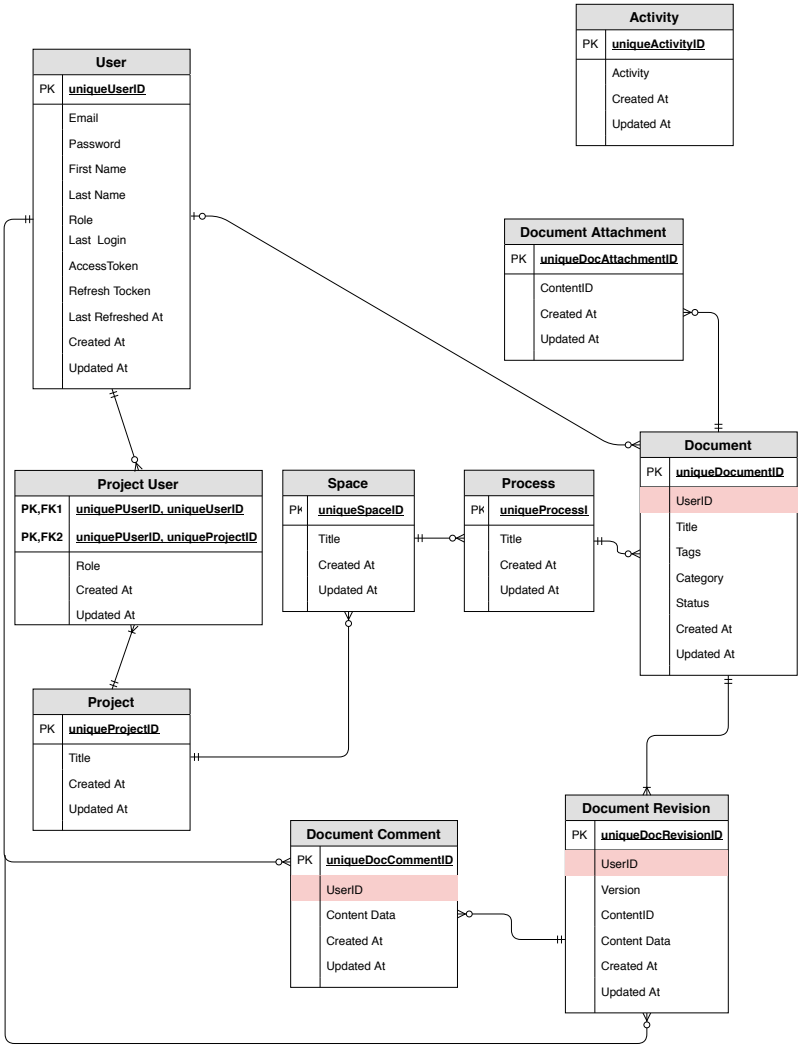
5.3 Použité technologie

JavaScript[8]

JavaScript (často zkrácený jako „JS“) je multiplatformní, vysoce interpretovaný, objektově orientovaný programovací jazyk, který odpovídá specifikaci *ECMAScript*[9]. Spolu s HTML

[8] **JavaScript** - https://www.javascript.com/

[9] **ECMAScript**, což je specifikace skriptovacího jazyka standardizovaná společností Ecma International v ECMA-262 a ISO / IEC 16262, která byla vytvořena za účelem standardizace JavaScriptu tak, aby podporovala více nezávislých implementací.



Obrázek 5.5: Zobrazení ERD diagramu výsledné aplikace.

a CSS je JavaScript jednou z hlavních technologií WWW[10], ale používá se také v mnoha jiných prostředích [19, 16, 27].

JavaScript umožňuje interaktivní webové stránky, a je nezbytnou součástí webových aplikací. Používá ho drtivá většina internetových stránek, a hlavní webové prohlížeče mají k jeho provedení vyhrazený JavaScript nástroj. JavaScrip, jako jazyk s více paradigmaty, podporuje události řízené, funkční a imperativní (včetně objektově orientovaných a prototypových) programovacích stylů. Má rozhraní API pro práci s textem, poli, daty, regulárními výrazy a DOM, ale jazyk sám o sobě neobsahuje žádné vstupy a výstupy, jako je síť, úložiště nebo grafická zařízení. Opírá se o hostitelské prostředí, ve kterém je zabudován, aby tyto funkce poskytoval. Zpočátku byl implementován pouze na straně klienta ve webových prohlížečích, nyní jsou JavaScript nástroje začleněny do mnoha dalších typů hostitelského softwaru: server-side[11] ve webových serverech a databázích, v non-webových programech

[10] **WWW** je zkratkou World Wide Web – celosvětová síť (internet).

[11] **Server-side** skriptování je technika používaná ve vývoji webu, která zahrnuje používání skriptů na webovém serveru, které vytvářejí odpověď přizpůsobenou požadavkům každého uživatele (klienta) na webovou stránku.

(textové procesory, PDF software) a za běhu prostředí, která zpřístupňují JavaScript pro psaní mobilních a desktopových aplikací, včetně widgetů pro pracovní plochy [26, 19].

TypeScript[12]

TypeScript je open-source[13] programovací jazyk. TypeScript je typovaná nadmnožina JavaScriptu, která jej obohacuje o atributy OOP[14] jako třídy, moduly, atd. Projekt byl vyvinut pro vývoj velkých aplikací a transcompilerů[15] na JavaScript [23]. TypeScript kód se kompiluje do prostého JavaScriptu, a s ohledem na výše popsaný vztah těchto dvou jazyků vyplývá, že libovolný kód v JavaScriptu je automaticky validním TypeScript kódem.

Vývojáři TypeScript hledali řešení, které by neporušilo kompatibilitu se standardem. S vědomím, že tehdejší standardní návrh ECMAScript sliboval podporu pro třídové programování, byl TypeScript založen na tomto návrhu. V tomto smyslu byl TypeScript ukázkou toho, co očekávat od standardu ECMAScript § 6. vydání - ECMAScript 2015 [13].

5.3.1 Back-end technologie

PostgreSQL[16]

PostgreSQL je výkonný open-source objektově-relační databázový systém, který využívá a rozšiřuje jazyk SQL v kombinaci s mnoha funkcemi, které bezpečně ukládají a rozšiřují nejsložitější pracovní úlohy. Počátky PostgreSQL sahají do roku 1986 jako součásti projektu POSTGRES na University of California v Berkeley a mají více než 30 let aktivního rozvoje na základní platformě. POSTGRES byl a stále je průkopníkem mnoha konceptů, které byly v některých komerčních databázových systémech k dispozici teprve později.

PostgreSQL je nyní nejpokročilejší dostupná open-source databáze. Její architektura je založena na modelu klient/server. Server, který spravuje databázové soubory, přijímá připojení k databázi z klientských aplikací a provádí akce databází jménem klientů. Program databázového serveru se nazývá *postgres*. Program klienta (*frontend*), který chce provádět operace s databází, může být velmi různorodý: textově orientovaný nástroj, grafická aplikace, webový server, který přistupuje k databázi pro zobrazení webových stránek nebo specializovaný nástroj pro údržbu databáze. PostgreSQL podporuje velkou část standardu SQL a nabízí mnoho moderních funkcí:

- Složité dotazy (z angl. *Complex queries*);
- Cizí klíče (z angl. *Foreign keys*);
- Trigery (z angl. *Triggers*);
- Aktualizovatelné pohledy (z angl. *Updatable views*);
- Transakční integritu (z angl. *Transactional integrity*);

[12] **TypeScript** - https://www.typescriptlang.org/

[13] **Open-source** v překladu z ang. *Otevřený/Svobodný software* počítačový program s otevřeným zdrojovým kódem, který je poskytnut dalším vývojářům, kteří jej mohou zdarma využívat: studovat, upravovat a přispívat k jeho vývoji a také ho distribuovat dále.

[14] **OOP** - objektově orientované programování.

[15] **Transcompiler, transpiler** - kompilátor typu *source-to-source*, který produkuje kód v jiném jazyce, než byl kompilovaný program napsán.

[16] **PosthreSQL** - https://www.postgresql.org

- Řízení víceúrovňové souběžnosti (z angl. *Multiversion concurrency control*);

Také PostgreSQL umožnuje uživateli provést rozšíření daného systému řadou způsobů, například přidáním nových: typů dat, funkcí, operátorů, agregačních funkcí, indexových metod a procedurálních jazyků. PostgreSQL běží na všech hlavních operačních systémech, od roku 2001 je kompatibilní se standardem ACID[17] a má výkonné doplňky, jako je například populární rozšiřující modul databáze PostGIS, který přidává podporu pro geografické objekty.

MongoDB[18]

MongoDB je multiplatformní, výkonný, vysoce škálovatelný open-source databázový systém pro správu databází (DBMS), který používá dokumentově orientovaný databázový model, což podporuje různé formy dat [8]. Od roku 2019 daný systém dosáhl pozice přední databáze NoSQL. V současné době se tato databáze používá jako backendový datový sklad mnoha známých organizací: IBM, Twitter, Zendesk, Forbes, Facebook, Google atd. [31].

Namísto použití tabulek a řádků, jako v relačních databázích, se architektura MongoDB skládá z kolekcí a dokumentů. Záznam v MongoDB je dokument, který je datovou strukturou složenou z párů polí a hodnot. Dokumenty MongoDB jsou podobné objektům JSON, ale používají variantu nazvanou Binary JSON (BSON), která pojme více datových typů. Pole v dokumentech jsou podobná sloupcům v relační databázi a hodnoty, které obsahují, mohou být různé typy dat, včetně dalších dokumentů, polí a polí dokumentů. Kolekce obsahují sady dokumentů a fungují jako ekvivalent relačních databázových tabulek. Mohou také obsahovat libovolný typ dat, ale omezení je, že data v kolekci nelze šířit do různých databází.

Formát pro ukládání a výměnu dokumentů BSON, používaný v MongoDB, poskytuje binární reprezentaci dokumentů typu JSON. Dalším klíčovým prvkem je automatické sdílení, které umožňuje distribuci dat v kolekci MongoDB napříč různými systémy pro horizontální škálovatelnost, jelikož objem dat a požadavky na propustnost se zvyšují.

DBMS NoSQL používá jedinou hlavní architekturu pro konzistenci dat se sekundárními databázemi, které uchovávají kopie primární databáze. Operace jsou automaticky replikovány do těchto sekundárních databází pro automatické převzetí služeb při selhání. MongoDB používá dynamické databázové schéma, které umožňuje vytváření a integraci dat pro aplikace jednodušeji a rychleji. Databázové schéma obsahuje popisy obsahu, struktury a omezení integrity používané k vytvoření a udržování databáze. Jako jeden z představitelů systémů správy databází, MongoDB používá definice dat ve schématu především proto, aby poskytl přístup a kontrolu přístupu k datům v databázi [7].

Node.js[19]

Node.js je platforma založená na běhovém prostředí JavaScriptu pro snadné vytváření rychlých, škálovatelných síťových aplikací. Tento nástroj převede JavaScriptový kód na rychlejší

[17] **ACID** je zkratka od angl. *Atomicity, Consistency, Isolation, Durability*, což je souborem vlastností databázových transakcí určených k zajištění jejich platnosti v případě výskytu chyb, výpadků napájení apod.

[18] **MongoDB** - https://www.mongodb.com, https://searchdatamanagement.techtarget.com/definition/MongoDB

[19] **Node.js** - https://medium.freecodecamp.org/what-exactly-is-node-js-ae36e97449f5, https://nodejs.org/

strojový kód. Strojový kód je nízko úrovňový kód, který může počítač spustit, aniž by jej musel nejprve interpretovat.

Node.js má architekturu řízenou událostmi schopnou asynchronního I/O (input/output), bez blokování, což ho činí lehkým a efektivním, vhodným pro datově náročné aplikace v reálném čase, které běží napříč distribuovanými zařízeními. Cílem návrhů dané platformy je optimalizovat propustnost a škálovatelnost ve webových aplikacích s mnoha vstupními / výstupními operacemi, stejně, jako pro webové aplikace v reálném čase (např. komunikační programy v reálném čase a hry prohlížeče).

Node.js umožňuje vývojářům používat JavaScript pro psaní nástrojů příkazového řádku a pro skriptování na straně serveru – spuštění skriptů na straně serveru pro vytvoření dynamického obsahu webové stránky před odesláním stránky do webového prohlížeče uživatele. V důsledku toho Node.js představuje paradigma "JavaScript everywhere", sjednocující vývoj webových aplikací kolem jednoho programovacího jazyka [25].

Express[20]

Express.js neboli jednoduše *Express, ExpressJS*, je framework pro webové aplikace, který poskytuje jednoduché API pro tvorbu webových stránek, webových aplikací a backendů. ExpressJS poskytuje robustní sadu funkcí pro webové a mobilní aplikace. Při jeho použití není třeba se zabývat prací s protokoly, požadavky, procesy atd. Také tento framework poskytuje minimální rozhraní, potřebné nástroje pro tvorbu aplikací a je flexibilní: je zde k využití mnoho modulů na npm[21], které lze přímo zapojit [4].

Express.js a Node.js „dali“ JavaScriptu (jazyku, který se primárně používal pro tvorbu front endu webových aplikaci) nové funkce backendu - což umožnilo vývojářům poprvé vytvářet backend s JavaScriptem. Tyto dva nástroje společně umožňují vytvořit celý web s využitím JavaScriptu: pomocí Node.js vyvinout aplikace na straně serveru, a poté tuto aplikaci publikovat jako webové stránky s aplikací Express [61].

Typeorm[22]

TypeORM je multiplatformní ORM[23], který může běžet na řadě platforem, a může být použit s TypeScriptem či JavaScriptem. Jeho cílem je vždy podporovat nejnovější funkce JavaScriptu a poskytovat další funkce, které pomohou vyvinout jakoukoli aplikaci, která využívá databáze – od malých aplikací s několika tabulkami až po rozsáhlé podnikové aplikace s více databázemi.

Od nástroje TypeORM se dá očekávat, že vytvoří databázové tabulky za vývojáře, a také vyhledá, vloží, aktualizuje a smaže vybraná data bez nutnosti psaní mnoho těžce udržovatelných SQL dotazů. TypeORM podporuje Active Record a Data Mapper vzory, na rozdíl od všech ostatních existujících JavaScriptových ORM, což znamená, že je zde možnost psát škálovatelné a udržovatelné aplikace produktivnějším způsobem.

[20] **Express** - https://expressjs.com

[21] **NPM**, zkráceně Node Package Manager je správce balíčků pro programovací jazyk JavaScript.

[22] **Typeorm** - https://github.com/typeorm/typeorm

[23] **ORM** zkratka od angl. Object-relational mapping, což je v IT programovací technika pro převod dat mezi systémy nekompatibilního typu pomocí objektově orientovaných programovacích jazyků.

SocketIO[24]

Socket.IO je JavaScriptová knihovna, která umožňuje v reálném čase komunikaci obousměrnou a založenou na událostech mezi prohlížečem a servery. Funguje na každé platformě, prohlížeči nebo zařízení, se zaměřením na spolehlivost a rychlost. Skládá se ze dvou komponent, které mají téměř identické API: Node.js serveru a Javascriptové klientské knihovny pro prohlížeč. Hlavními rysy dané knihovny jsou:

- Spolehlivost (z angl. *Reliability*): připojení jsou navázána i v případě přítomnosti proxi serveru[25], load balanceru[26], firewallu[27] nebo antivirového softwaru pomocí nástrojů Engine.IO;
- Podpora automatického opětovného připojení (z angl. *Auto-reconnection support*): pokud není instruován jinak, odpojený klient se pokusí navázat spojení navždy, dokud nebude server znovu dostupný;
- Detekce odpojení (z angl. *Disconnection detection*): mechanismus prezenčního signálu je implementován na úrovni Engine.IO, což umožňuje serveru i klientovi vědět, kdy ten druhý již neodpovídá;
- Binární podpora (z angl. *Binary support*): dovoluje vysílat všechny serializovatelné datové struktury, včetně: ArrayBuffer a Blob v prohlížeči, a také ArrayBuffer a Buffer v Node.js;
- Podpora multiplexování (z angl. *Multiplexing support*): umožňuje vytvořit několik jmenných prostorů, které budou fungovat jako oddělené komunikační kanály, ale budou sdílet stejné základní spojení;
- Podpora pokojů (z angl. *Room support*): v rámci každého jmenného prostoru se dají definovat libovolné kanály, nazývané *Room*, do kterých se mohou připojit/odpojit koncové body. Pak je možné vysílat do kteréhokoliv „pokoje“, a dosáhnout tak každého koncového bodu, který se k němu připojil.

Socket.IO primárně používá protokol WebSocket s dotazováním jako záložní možnost, a zároveň poskytuje stejné rozhraní. Ačkoli to může být používáno jako jednoduchý wrapper[28] pro WebSocket, poskytuje mnoho dalších funkcí, včetně vysílání do více koncových bodů, ukládání dat spojených s každým klientem a asynchronních I/O.

[24] **SocketIO** - https://socket.io

[25] **Proxi server** je server, který funguje jako prostředník mezi klientem a cílovým počítačem (serverem). Sám vystupuje před cílovým počítačem jako klient.

[26] **Load balancer** je server, který používá techniku pro rozložení zatížení mezi dva nebo více strojů (počítačů, síťových linek, atd.) aby bylo dosaženo optimálního využití, prostupnosti nebo času odezvy.

[27] **Firewall** je síťové zařízení, které slouží k řízení a zabezpečování síťového provozu mezi sítěmi s různou úrovní důvěryhodnosti a zabezpečení.

[28] **Wrapper** je podprogram v softwarové knihovně nebo počítačovém programu, jehož hlavním účelem je volání druhého podprogramu nebo systémové volání s malým nebo žádným dodatečným výpočtem.

PassportJS[29]

PassportJS neboli jednoduše Passport je autentizační middleware[30] pro Node.js. Je navržen tak, aby sloužil jedinému účelu – autentizaci požadavků.

V moderních webových aplikacích může mít autentizace různé formy. Tradičně se uživatelé přihlašují zadáním uživatelského jména a hesla. S nástupem sociálních sítí se stalo populární metodou ověřování, které používají Facebook, Twitter. Služby, které vystavují rozhraní API, často vyžadují pro ochranu přístupu pověření založená na tokenu.

Služba Passport uznává, že každá aplikace má jedinečné požadavky na autentizace. Mechanismy autentizace, známé jako *strategie*, jsou naprogramované a oddělené jako jednotlivé moduly. Aplikace si mohou vybrat, které strategie použít, aniž by vytvářely zbytečné závislosti. Strategie sahají od autentizace uživatelského jména a hesla, přes autentizace pomocí delegace OAuth[31] nebo přes federální autentizace pomocí OpenID[32].

Ve výchozím nastavení, pokud se autentizace nezdaří, bude služba Passport reagovat stavem *401 Unauthorized*, a žádné další obsluhy nebudou vyvolány. Pokud autentizace proběhne úspěšně, bude vyvolán další obslužný program a vlastnost, která bude nastavena na autentizovaného uživatele.

5.3.2 Front-end technologie

React

React[33] (také známý jako React.js nebo ReactJS) je knihovna JavaScriptu pro jednoduché vytváření uživatelských rozhraní. React může být použit jako základ pro vývoj jednostránkových nebo mobilních aplikací. Je to nejefektivnější metoda pro rychle se měnící data, která je třeba zaznamenat. Načtení dat je však pouze začátkem toho, co se děje na webové stránce, což je důvod, proč komplexní aplikace React obvykle vyžadují použití dalších knihoven pro správu stavu (byl použit Mobx), směrování a interakci s rozhraním API. Hlavními rysy a významnými vlastnostmi dané knihovny jsou:

- Určitost (z angl. *Declarative*): navrhnutá vytvořením jednoduchého pohledu pro každý stav v aplikaci React efektivně zaktualizuje a vykreslí správné komponenty, jestliže se změní data. Deklarativní zobrazení činí kód předvídatelnějším a snadnějším během ladění;

- Založení na komponentech (z angl. *Component-Based*): react dává možnost sestavit zapouzdřené komponenty, které spravují svůj vlastní stav, a poté je vytvořit tak, aby spolu vytvářely komplexní uživatelská rozhraní. Vzhledem k tomu je logika komponent napsána v JavaScriptu místo použití šablon. Je možné snadno předávat bohatá data prostřednictvím aplikace.

[29] **PassportJS** - http://www.passportjs.org

[30] **Middleware** je v informatice specializovaný software, který poskytuje aplikacím služby nad rámec služeb poskytovaných operačním systémem. Někdy se označuje jako „softwarové lepidlo“ (z angl. Software glue) [60].

[31] **OAuth** je otevřený protokol, cílem kterého je poskytnout bezpečnou autentizaci a autorizaci oproti API různých služeb.

[32] **OpenID** je otevřený standard popisující decentralizovaný způsob autentizace uživatelů

[33] **React** - https://reactjs.org/

Mobx[34]

Mobx je knihovna, která činí stavový management jednoduchý, škálovatelný pomocí použití transparentního funkčního reaktivního programování (TFRP). Postoj společnosti MobX je velmi jednoduchý: všechno, co se dá z aplikace odvodit, musí byt odvozeno automaticky. To zahrnuje uživatelské rozhraní, serializaci dat, komunikaci se serverem atd.

React a MobX jsou spolu silnou kombinací a poskytují optimální a unikátní řešení běžných problémů ve vývoji aplikací. React vykreslí stav aplikace tím, že poskytne mechanismy, které ji převedou do stromu renderovatelných komponent: mechanismy pro optimální vykreslování uživatelského rozhraní pomocí virtuálního DOM, který snižuje počet nákladných DOM mutací. MobX poskytne mechanismus pro ukládání a aktualizaci stavu aplikace, který pak používá: mechanismy pro optimální synchronizaci stavu aplikací s komponenty React pomocí reaktivního grafu stavu virtuální závislosti, který není nikdy zastaralý, a aktualizuje se pouze v případě nezbytné nutnosti.

Quill[35]

Quill je moderní open-source textový editor WYSIWYG[36] postavený pro kompatibilitu a rozšiřitelnost v moderním webu. Byl navrhnut a vytvořen Jasonem Chenem a Byronem Milliganem. Je aktivně udržovaný společností Slab Inc. Quill je inicializován prvkem DOM, který obsahuje editor. Obsah tohoto prvku se stane počátečním obsahem Quill.

Díky své modulární architektuře a expresivnímu rozhraní API je Quill zcela přizpůsobitelný pro všechny potřeby. Moduly umožňují přizpůsobit chování a funkčnost Quill podle požadavku uživatele. K dispozici je několik oficiálně podporovaných modulů, ze kterých lze vybrat libovolný z nich s dalšími konfiguračními možnostmi a rozhraním API.

Quill umožnuje granulativní přístup k obsahu editoru, změnám a událostem prostřednictvím jednoduchého rozhraní API. Pracuje konzistentně a deterministicky s JSON jako vstup i výstup. Qill má zajímavé rozšíření pod jménem Delta, které bylo využito při implementaci daného projektu.

Quill Delta[37]

Delta je jednoduchý, ale výrazný formát, který lze použít k popisu obsahu Quill a jeho změn. Formát je přísnou podmnožinou JSON, je čitelný pro člověka a snadno strojově tvořený. Deltami může byt popsán jakýkoliv Quill dokument, včetně všech informací o textu a formátování, bez dvojznačnosti a složitosti HTML.

Název Delta představuje dokumenty i změny dokumentů. Delty nejsou instrukce pro přechod z jednoho dokumentu do druhého. Delta představuje dokument vyjádřením pokynů počínajících z prázdného dokumentu.

Delty jsou implementovány jako oddělená a samostatná knihovna, která umožňuje její použití i mimo Quill. Daná knihovna je vhodná pro operační transformaci a lze ji použít v reálném čase, v aplikacích Google Docs.

[34] **Mobx** - https://github.com/mobxjs/mobx

[35] **Quill** - https://github.com/quilljs/quill

[36] **WYSIWYG** je zkratka z angl. *What you see is what you get* – co vidíš, to dostaneš. Tak se zpravidla označují editory, v nichž uživatel přímo upravuje webovou stránku tak, jak bude vypadat v běžném internetovém prohlížeči.

[37] **Quill Delta** - https://quilljs.com/docs/delta/

Yjs[38]

Yjs je framework pro off-line úpravy první verze *peer-to-peer* ve strukturovaných datech, jako je text, formát RTF, JSON nebo XML. Je jednoduché s ním pracovat, jelikož Yjs se stára o většinu problémů kolem umožnění souběžných úprav v rámci jednoho dokumentu. Nemá implementovánu transportní vrstvu, ale poskytuje konektor pro umožnění komunikace mezi Yjs klienty.

[38] **Yjs** - https://github.com/y-js/yjs

Kapitola 6

Testování

Kapitola testování se věnuje ověření funkčnosti a správnosti implementovaného EDM systému. Jsou zde popsány výsledky automatizovaného testování s použitím testovacího softwaru. Dále jsou představeny scénáře a zhodnocení výsledků provedených testů. Kromě zmíněných způsobů testování byla aplikace i její součásti průběžně testovány během vývoje i pomocí debuggování, čímž byly odstraněny nejzásadnější chyby v aplikaci.

6.1 Black box testování

Testování formou černé skříňky (black box) se zaměřuje na vstupy a výstupy programu bez znalosti, jak je naimplementován. Smyslem je analyzovat chování softwaru vzhledem k očekávaným vlastnostem tak, jak ho vidí uživatel.

Pro black box testování byl použit open-source testovací framework **Selenium**[1], který mimo jiné poskytuje rozšíření pro prohlížeč, které dovoluje zaznamenat vykonané akce do formátu vhodného pro zpracování strojem. Avšak některé testy vyžadovaly manuální modifikaci skriptu, jelikož bylo nutné pracovat například s generováním uživatelského jména, emailu nebo hesla.

Některé z testů uvádějí dva uživatele, ale testy byly implementovány tak, že se vykonaly nejprve kroky pro prvního uživatele, ten se odhlásil, a následovaly kroky pro druhého uživatele.

6.1.1 Testovací scénáře

Testovací scénáře zahrnují variace testování takové funkcionality, jak například:

1. Přihlášení s různými variacemi emailu a hesla
2. Registrace uživatele
3. Vyhledávání názvu dokumentů a tagů
4. Vytvoření tagů
5. Nahrávání příloh
6. Vytvoření revize

[1] **Selenium** - https://www.seleniumhq.org/

7. Změny stavů
8. Testování uživatelských práv
9. Zjišťování aktivity uživatele
10. Vytvoření komentáře
11. Tvorba dokumentu, atd.

Pro testování funkčností systému byla vytvořena sada testovacích scénářů. Každý testovací scénář má název, popisuje určitý úkon uživatele spolu s generovanými daty, vstupními podmínkami a očekávanými výsledky. Jako příklad jsou v tabulkách níže zobrazeny první testovací scénáře. Ostatní testovací scénáře najdete v příloze C.

ID: #1	Vytvoření vstupního dokumentu
Název:	Kontrola správného vytvoření vstupního dokumentu ve vybrané oblasti a procesu
Vstupní podmínky:	1. Uživatel je přihlášen 2. Uživatel je v nově vytvořeném projektu s původními šablonami
Postup:	1. Najedeme kurzorem myši na proces "Plan Quality management"v oblasti "Quality management"- Objeví se plovoucí okno se vstupem 2. Klikneme na vstupní pole pro zadání názvu dokumentu a zadáme: "Sample document" 3. V plovoucím okně vybereme typ dokumentu: "Input document" 4. Na pravé straně plovoucího okna klikneme na ikonu šipky v kroužku 5. Klepneme na nově vytvořený dokument s názvem: "Sample document"
Očekáváný výsledek:	1. Uživatel se nachází na obrazovce dokumentu 2. Je zobrazen kolaborativní editor 3. Dokument je zařazen do kategorie vstupního dokumentu 4. Dokument je ve stavu "ready"

Tabulka 6.1: Testovací případ #1.

ID: #2	Změna stavu na "done"
Název:	Kontrola správné funkčnosti při změně stavu na "done"
Vstupní podmínky:	1. Uživatel je přihlášen 2. Uživatel je v nově vytvořeném projektu s původními šablonami 3. Uživatel se nachází v dokumentu s názvem: "Requirements Documentation"
Postup:	1. Klikneme na stav dokumentu: "Ready" 2. Klikneme na možnost: "Done" 3. Klikneme na tlačítko: "Submit"
Očekáváný výsledek:	1. Uživatel není schopný editovat dokument

Tabulka 6.2: Testovací případ #2.

ID: #3	Vytvoření revize
Název:	Kontrola správné funkčnosti vytváření revizí
Vstupní podmínky:	1. Uživatel je přihlášen 2. Uživatel je v nově vytvořeném projektu s původními šablonami 3. Uživatel se nachází v dokumentu s názvem: "Project Management Plan Updates"
Postup:	1. Uživatel vepíše na začátek dokumentu slovo: "Test" 2. Klikneme na tlačítko: "Save" 3. Klikneme na záložku: "Revisions"
Očekáváný výsledek:	1. V seznamu revizí se nachází revize s aktuálním datem a číslem revize 2. Po otevření revize se zobrazí náhled obsahující text "Test"na začátku dokumentu 3. Po otevření revize s číslem 0 se zobrazí původní revize, která neobsahuje text "Test"na začátku dokumentu

Tabulka 6.3: Testovací případ #3.

ID: #4	Registrace
Název:	Kontrola, zda se lze registrovat s neexistujícím jménem a heslem
Generována data:	1. `username` - "user"+timestamp+"email.cz" 2. `password` - "password" 3. `firstname` - "Name"+timestamp 4. `last_name` - "Surname"+timestamp
Vstupní podmínky:	1. Webová aplikace běží na úvodní stránce s přihlášením 2. Formulář na zadání jména a hesla je prázdný 3. Uživatel "random@email.cz"neexistuje
Postup:	1. Klikneme na odkaz: "Sign up here" 2. Klikneme na vstupní pole pro zadání emailu a zadáme: {username} 3. Klikneme na vstupní pole pro zadání hesla a zadáme: {password} 4. Klikneme na vstupní pole pro ověření hesla a zadáme: {password} 5. Klikneme na vstupní pole pro zadání jména a zadáme: {firstname} 6. Klikneme na vstupní pole pro zadání příjmení a zadáme: {last_name} 7. Klikneme na tlačítko "Sign up"
Očekáváný výsledek:	1. Uživatel je přihlášen do systému a registrován 2. Nezobrazí se hláška: "Invalid email or password."

Tabulka 6.4: Testovací případ #4.

6.2 White box testování

Při testování formou bílé skříňky (v angličtině white box), má tester přístup ke zdrojovému kódu a testuje produkt na základě toho, jak vidí nejen co se děje na povrchu skříňky, ale i vnitřní reakce systému. Takový způsob testování lépe odhaluje, kde hledat chyby.

Mezi white box testování můžeme zařadit tvorbu Unit testů, které testují funkcionality nejmenších prvků systému. Jednotkou může být např. modul, třída nebo její metoda. Unit testy pro danou jednotku se vytvářejí při jejím vývoji. Cílem je najít a odstranit co nejvíce logických chyb již při vývoji, a tím usnadnit práci při testování systému. Některé `dumb` nebo `smart` komponenty byly ze začátku otestovány, avšak prototyp aplikace se měnil vnitřně tak často, že nemělo smysl tyto testy psát znovu. Užitečnější bylo Blackbox testování za pomoci *Selenia*, při kterém je skrytá implementace a testuje se pouze „zvenku“ na uživatelském rozhraní.

6.3 Výsledky testování

Testování celkem odhalilo kolem 10 závažných chyb z nichž byly všechny úspěšně vyřešeny a tedy byly odhaleny kritická místa systému. Bohužel systém stále obsahuje občasné chyby, které se nepodařilo doladit z důvodu implementace prioritních funkcí.

Při tvorbě aplikace se ze začátku postupovalo tak, že se vytvořila kostra aplikace se základním rozložením. Pak se vytvořily automatické Selenium testy, které pomohly při urychlení vývoje tím, že po implementaci určité z funkcí bylo možné tuto funkci automaticky přetestovat znovu.

Celkem bylo tedy otestováno 17 scénářů, při kterých se ověřila funkčnost implementovaného EDM systému dle specifikovaných požadavků. Testy byly spouštěny opakovaně, aby se předešlo náhodnému chování.

Selenium neposkytuje možnost, jak vytvořit dvě okna a přepínat mezi nimi. Z tohoto důvodu nebyla možnost implementovat testy pro interaktivní kolaboraci. Kolaborativní editor byl však otestován manuálně s více okny prohlížeče v anonymním režimu. Nanejvýš byl však testován s pěti okny najednou.

Konečné výsledky testovacích scénářů odpovídaly očekávání.

Kapitola 7

Závěr

Tato práce se zabývá systémy na elektronickou správu dokumentů (EDMS) z perspektivy vybraných znalostních oblastí procesů v projektovém řízení.

Cílem této práce bylo seznámit se s teorií z oblasti elektronického řízení projektů a projektového řízení. Poté na základě vhodně vybraného EDM modelu specifikovat požadavky a vytvořit návrh vlastního EDM systému, který by napomáhal v manažování dokumentů vznikajících během procesů řízení v oblastech projektového řízení jako: řízení kvality, řízení lidských zdrojů a řízení komunikace v rámci projektu. Návrh výsledné aplikace měl být vytvořen s ohledem na rozšiřitelnost pro správu dokumentů ostatních znalostních oblastí. Následně jsem měla zvolit vývojové prostředí, a po dohodě s vedoucí naimplementovat funkce prototypu navržené aplikace a na vybraném vzorku dat otestovat a zhodnotit funkčnost prototypu.

Ve výsledku prací se podařilo splnit vytyčené cíle v jejich plném rozsahu. Pro seznámení se s oblastí elektronického řízení projektů a projektového řízení jsem si nastudovala aktuální standard PMI z knihy PMBOK Guide (Project management body of knowledge), a také řadu internetových zdrojů.

Pro model jádra EDM aplikace byl zvolen relační model databáze, a jako jádro pro práci s dokumenty NoSQL dokumentový model databáze.

Specifikace požadavků byla vytvořena na základě porovnání s aplikací Confluence od Atlassian, Trello, atd. Většina požadavků se týkala části konkurenceschopné funkcionality EDM aplikace v porovnání s jinými EDMS. Další požadavky byly zaměřeny na vstupy a výstupy procesů vybraných oblastí projektového řízení. Specifikované požadavky na funkcionalitu z pohledu vybraných znalostních oblastí řízení projektů byly splněny v předem definovaném rozsahu. Specifikované požadavky na funkcionalitu z pohledu EDMS byly splněny v základním rozsahu z každé kategorie:

- Metadata: výsledná EDM aplikace nabízí možnost přidání a editace takových metadat dokumentu, jako jsou tagy. Také aplikace automaticky generuje metadata o tom, kdo a kdy vytvořil projekt, verze dokumentu;
- Vyhledávání: aplikace nabízí možnost vyhledávání dokumentů na základě jména a tagů v rámci každého projektu, což výrazně zjednodušuje nalezení konkrétního dokumentu ve chvíli, kdy projekt obsahuje mnoho procesů se svými dokumentovými vstupy a výstupy;
- Vkládaní: vkládání dokumentů do systému je umožněno prostřednictvím příloh, které se mohou nahrát do dokumentů;

- Automatizace procesu neboli Workflow: je v aplikaci vyřešena prostřednictvím zavedení stavů dokumentu, které „způsobují“ následné shromáždění dokumentů k revizi na vlastním dashboardu osoby zodpovědné za revizi;
- Verzování: je zajištěno, uživatel si může prohlédnout jak dokument vypadal v jedné z vybraných verzí, ale nemůže ho do ní vrátit;
- Dostupnost a sdílení: je vyřešeno prostřednictvím pozvání registrovaného uživatele do projektu;
- Archivace a zálohování: pro aplikaci je částečně vyřešena verzováním dokumentů a zálohováním skenovaných papírových dokumentů jako příloh;
- Přístupová práva a bezpečnost: je řešena pomocí přidělení určité role uživatelům, které zveme k projektu. Role a rozdělení práv byly popsány výše v textu práce. Systém rolí, spolu se zabezpečením speciálního autorizačního systému, zajišťuje vysokou míru bezpečnosti daného programu;
- Integrace: je řešena částečně pro obrázky. Zobrazení PDF dokumentů, které bylo plánováno v návrhu, nebylo možné splnit kvůli chybě ve veřejné knihovně, jež se pro tento úkol měla použít.

Rozšiřitelnost vlastního EDMS se vyřešila prostřednictvím použití projektových a dokumentových šablon. Projektové šablony dovolují vytvářet projekty s předem vytvořenou strukturou pro jednotlivé znalostní oblasti procesního řízení projektů (řízení kvality, řízení lidských zdrojů, řízení komunikace), a také umožnují si vytvářet nové vlastní oblasti, procesy a dokumenty. Dokumentové šablony dovolují díky opakovatelnému použití některých dokumentů v různých oblastech šetřit čas s vytvářením nového dokumentu kopírováním z již navržených šablon.

Jako vývojové prostředí jsem zvolila Visual Studio Code, ve kterém proběhla implementace aplikace především v jazyce JavaScript.

Automatizované testování aplikace provedené pomocí speciálního testovacího IDE odhalilo několik chyb, z nichž většina byla ve finální verzi opravena.

Vzhledem k náročnosti zadání je tedy hlavním výstupem diplomové práce prototyp EDM systému, který napomáhá v manažování dokumentů vznikajících během procesů řízení především v oblastech řízení kvality, řízení lidských zdrojů, řízení komunikace, a který pracuje s dokumenty na základě dokumentového databázového modelu.

7.1 Možnosti dalšího vývoje EDM aplikace

Budoucí rozšíření lze rozdělit na několik kategorií. První kategorií jsou vylepšení týkající se správy uživatele. Mezi taková vylepšení patří: ověření elektronické adresy uživatele při registraci (zasláním emailu na uvedenou adresu s verifikačním kódem / odkazem), umožnění obnovení zapomenutého hesla do aplikace, změna hesla, úprava profilu (změna jména uživatele), použití profilové fotografie.

Do druhé kategorie spadají vylepšení tykající se správy dokumentů: možnost obnovení dokumentu do vybrané verze, přejmenování dokumentu, přesunu dokumentu mezi procesy, exportu dokumentu (např. do PDF), sdílení dokumentu pomocí vygenerování unikátního odkazu odkazujícího přímo na dokument, možností vidět, kteří uživatelé se právě dívají na konkrétní dokument.

Do třetí kategorie patří vylepšení tykající se správy komentářů: možnost odpovídat na komentáře (vytvoření vláken komentářů), citovat vybraný obsah dokumentu do komentáře, editovat / mazat komentáře, umožnit richtext editor pro psaní komentářů.

Do čtvrté kategorie spadají vylepšení týkající se práce s kolaborativním editorem dokumentů: rozšíření možnosti richtext editoru (změna písma, velikosti, atd.), vložení multimediálního obsahu (videa), vložení a možnosti správy tabulek, možností anotace v dokumentu a zobrazení toho, který uživatel upravil vybranou část dokumentu.

Do poslední kategorie patří vylepšení tykající se správy projektů: možnosti přesunu procesů (a všech přiřazených dokumentů) do jiné oblasti, možnosti přesunu (sdílení, duplikace) procesů / oblasti do jiného projektu.

Výsledný projekt představuje výborný základ pro stavbu rozličných vylepšení, a proto vylepšení zmíněná výše nejsou finální, ale jsou poměrně důležitá pro zlepšení uživatelské přívětivosti a zkušenosti, které v dnešní době hrají jednou z klíčových rolí v úspěšnosti jakéhokoliv produktu.

Literatura

[1] *Databáze Národní knihovny ČR*. NK ČR, [Online; navštíveno 14.11.2018]. URL https://aleph.nkp.cz/F/4EI81PL8QLG54PJA6M7X6XAAHB8CL76G4X3USKS18LH6BFH3X5-02754?func=find-b&find_code=WTD&x=0&y=0&request=databaze&adjacent=N

[2] *Document Management Systems*. [Online; navštíveno 09.11.2018]. URL https://www.tdx.cat/bitstream/handle/10803/6160/04Nfm04de12.pdf;sequence=4

[3] *EDMS - Electronic Document Management System*. EDMS, [Online; navštíveno 07.11.2018]. URL http://www.edms.net/

[4] *ExpressJS Tutorial*. TutorialsPoint, [Online; navštíveno 6.04.2019]. URL https://www.tutorialspoint.com/expressjs/expressjs_overview.htm

[5] *Introduction to ODBMS: Definition*. [Online; navštíveno 16.11.2018]. URL http://www.odbms.org/introduction-to-odbms/definition/

[6] *IS ALeX Document Management System*. [Online; navštíveno 13.11.2018]. URL https://www.isalex.cz/document-management-system/license/

[7] *ISO/IEC TR 10032:2003 Information technology - Reference Model of Data Management*. International Organization for Standardization, [Online; navštíveno 29.03.2019]. URL https://www.iso.org/standard/38607.html

[8] *MongoDB Tutorial: About MongoDB*. Quackit.com, [Online; navštíveno 29.03.2019]. URL https://www.quackit.com/mongodb/tutorial/about_mongodb.cfm

[9] *Relační databáze*. [Online; navštíveno 15.11.2018]. URL http://vyuka.greendot.cz/materialy/material-4.pdf

[10] *Se správou dokumentů pomohou systémy DMS nebo EDM*. E15.cz, [Online; navštíveno 07.11.2018]. URL https://www.e15.cz/magazin/se-spravou-dokumentu-pomohou-systemy-dms-nebo-edm-978835

[11] *Objektově orientované systémy databází*. 2005, [Online; navštíveno 20.11.2018]. URL https://wikisofia.cz/wiki/Objektov%C4%9B_orientovan%C3%A9_syst%C3%A9my_datab%C3%A1z%C3%AD

[12] *MS ACCESS 2010 relační databáze*. DocPlayer, 2010, elektronická učebnice.
URL https://docplayer.cz/4209030-Ms-access-2010-relacni-databaze-elektronicka-ucebnice.html

[13] *ECMAScript 2015 Language Specification*. Ecma International, Červen 2015, [Online; navštíveno 12.03.2019].
URL http://www.ecma-international.org/ecma-262/6.0/index.html

[14] *Electronic Document Management Cookbook*. University of Toronto: Information Technology Services, Červenec 2017, [Online; navštíveno 10.11.2018].
URL https://easi.its.utoronto.ca/wp-content/uploads/2017/07/Document_Management_Cookbook.pdf

[15] *„Kdo vlastní informace ten vlastní svět“ (N. Rothschild)*. Obshchestvoznanie-online, Leden 2017, [Online; navštíveno 01.01.2019].
URL http://xn----7sbbbfrcoknutbddbdh1cu8l.xn--p1ai/info/news/kto-vladeet-informatsiey-tot-vladeet-mirom-n-rotshild-/

[16] *ECMAScript 2018 Language Specification*. Ecma International, Červen 2018, [Online; navštíveno 21.03.2019].
URL http://www.ecma-international.org/ecma-262/9.0/index.html

[17] *Základy relačních databází, jejich využití v programování webu*. Srpen 2018, [Online; navštíveno 15.11.2018].
URL https://kme.vse.cz/wp-content/uploads/page/534/4.-Z%C3%A1klady-rela%C4%8Dn%C3%ADch-datab%C3%A1z%C3%AD-jejich-vyu%C5%BEit%C3%AD-v-programov%C3%A1n%C3%AD-webu.pdf

[18] *ACID properties of transactions*. Leden 2019, [Online; navštíveno 20.11.2018].
URL https://www.ibm.com/support/knowledgecenter/en/SSGMCP_5.4.0/product-overview/acid.html

[19] *What is JavaScript?* Mozilla and individual contributors, Březen 2019, [Online; navštíveno 20.03.2019].
URL
https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript

[20] Antonín Carda, R. K.: *Workflow : Nástroj manažera pro řízení podnikových procesů*. Grada Publishing, 2003, ISBN ISBN 80-247-0666-0, 2., rozšířené a aktualiz.

[21] Bayer, T.: *Úvod do OOP*. [Online; navštíveno 20.11.2018].
URL https://web.natur.cuni.cz/~bayertom/images/courses/Prog2/prog2_0.pdf

[22] Boček, R.: *Zmena informačního systému podniku jako reakce na stávající problémy*. Diplomová práce, Masarykova univerzita, Duben 2008, str. 27, "Odpor lidí".
URL https://is.muni.cz/th/a2jyf/DP_Radim_Bocek_100568.odt

[23] Bright, P.: *Microsoft TypeScript: the JavaScript we need, or a solution looking for a problem?* Ars Technica, Condé Nast., Říjen 2012, [Online; navštíveno 12.03.2019].
URL https://arstechnica.com/information-technology/2012/10/microsoft-typescript-the-javascript-we-need-or-a-solution-looking-for-a-problem/

[24] Cejpek, J.: *Informace, komunikace a myšlení : úvod do informační vědy*. Karolinum, 2005, ISBN 80-246-1037-X.

[25] Cuomo, J.: *JavaScript Everywhere and the Three Amigos (Into the wild BLUE yonder!)*. Září 2013, [Online; navštíveno 10.03.2019].
URL https://www.ibm.com/developerworks/community/blogs/gcuomo/entry/javascript_everywhere_and_the_three_amigos?lang=en

[26] Dr.Rauschmayer, A.: *Speaking JavaScript*. O'Reilly Media, 2014, ISBN 978-1449365035.

[27] Flanagan, D.: *JavaScript: The Definitive Guide, 6th Edition*. O'Reilly Media, 2011, ISBN 978-0596805524.

[28] Fleissig, S.: DMS: systémy pro správu a oběh dokumentů. *IT Systems*, Říjen 2004, [Online; navštíveno 07.11.2018].
URL https://www.systemonline.cz/clanky/dms-systemy-pro-spravu-a-obeh-dokumentu.htm

[29] Geyer, J.: *Porovnání objektových a relačních databázových systémů*. Diplomová práce, Jihočeská univerzita v Českých Budějovicích, 2012, [Online; navštíveno 10.11.2018].
URL https://theses.cz/id/6304h3/Porovnani_RDBMS_a_OODBMS.pdf

[30] Girouard, H.: *Categorizing Components Into Smart Dumb Components, in React*. Alligator.io LLC, Listopad 2018, [Online; navštíveno 10.05.2019].
URL https://alligator.io/react/smart-dumb-components/

[31] Goel, A.: *What is MongoDB? Introduction, Applications, Advantages and Examples*. hackr.io, [Online; navštíveno 29.03.2019].
URL https://hackr.io/blog/mongodb-introduction-applications-advantages-examples

[32] Grehan, R.; Barry, D.; Zicari, R. V.: *Introduction to ODBMS: Short History*. [Online; navštíveno 16.11.2018].
URL http://www.odbms.org/Introduction/history.aspx

[33] Hardt, D.: *The OAuth 2.0 Authorization Framework*. Internet Engineering Task Force (IETF), [Online; navštíveno 12.04.2019].
URL https://tools.ietf.org/html/rfc6749

[34] Institute, P. M.: *A guide to the project management body of knowledge (PMBOK guide)*. Newtown Square, Pennsylvania, USA: Project Management Institute, Inc., 6 vydání, 9 2017, ISBN 978-1-62825-184-5.

[35] Jones, M.: *The OAuth 2.0 Authorization Framework: Bearer Token Usage*. Internet Engineering Task Force (IETF), [Online; navštíveno 12.04.2019].
URL https://tools.ietf.org/html/rfc6750#section-2.1

[36] Komloši, J.: *Objektově orientované geografické databáze*. Diplomová práce, Vysoké učení technické v Brně, 2010, [Online; navštíveno 21.11.2018].
URL http://www.fit.vutbr.cz/study/DP/DP.php.cs?id=8164&file=t

[37] Kočí, R.; Křena, B.: *Úvod do softwarového inženýrství, 2.přednáška*. Září 2018, [Online; navštíveno 2.03.2019].
URL https://wis.fit.vutbr.cz/FIT/st/cfs.php?file=%2Fcourse%2FIUS-IT%2Flectures%2FIUS2.pdf&cid=9410

[38] Krčál, M.: Document management systems. *Inflow*, Duben 2008, [Online; navštíveno 09.11.2018].
URL http://www.inflow.cz/document-management-systems

[39] Kunstová, R.: *Efektivní správa dokumentů: Co nabízí Enterprise Content Management*. Grada Publishing, 2010, ISBN ISBN 978-80-247- 3257-2.

[40] Lechner, T.: *Elektronické dokumenty v právní praxi*. Leges, 2013, ISBN 978-80-87576-41-0.

[41] Štěpán Martínek: *Document management system*. IT SYSTEMS, Září 2012, [Online; navštíveno 05.11.2018].
URL https://www.systemonline.cz/sprava-dokumentu/dms-moderni-trend-v-praci-s-dokumenty.htm

[42] Mazáč, M.: *Přínosy zavedení elektronického systému pro správu dokumentů*. Diplomová práce, České vysoké učení technické v Praze, Leden 2015, [Online; navštíveno 10.11.2018].

[43] McCreary, D.; Kelly, A.; Shaw, T.: *Making Sense of NoSQL: A guide for managers and the rest of us*. Shelter Island: Manning, 2013, ISBN 978-161-7291-074.

[44] Oppel, A.: *SQL demystified*. Computer press, 2012, ISBN ISBN 97-880-2511-7071, vyd. 1.

[45] Outlet, P.: *Traité de documentation. Le livre sur le livre, théorie et pratique*. Palais mondial, 1934, ISBN XIII s.

[46] *Databáze*. [Online; navštíveno 15.11.2018].
URL https://cs.wikipedia.org/wiki/Datab%C3%A1ze

[47] *Document management system*. [Online; navštíveno 05.11.2018].
URL https://sk.wikipedia.org/wiki/Document_management_system

[48] České republiky, P.: *Zákon č. 499/2004 Sb: o archivnictví a spisové službě*. Červen 2004, [Online; navštíveno 06.11.2018].
URL http://www.ahmp.cz/page/docs/zakon-499-2004.html

[49] Ritchie, C.: *Relational Database Principles*. Cengage Learning EMEA, 2002, ISBN 0-8264-5713-4.

[50] Rouse, M.: *DEFINITION: Class*. Srpen 2005, [Online; navštíveno 17.11.2018].
URL https://whatis.techtarget.com/definition/class

[51] Rouse, M.: *EDM (Electronic Document Management)*. Září 2005, [Online; navštíveno 07.11.2018].
URL https://searchsqlserver.techtarget.com/definition/EDM

[52] Rychlý, M.; Kolář, D.: *NoSQL databáze*. [Online; navštíveno 25.11.2018].
URL http://www.fit.vutbr.cz/~rychly/public/docs/slides-nosql-databases/slides-nosql-databases.pdf

[53] Shapiro, M.; Preguiça, N.; Baquero, C.; aj.: *Conflict-free Replicated Data Types*. Archive ouverte HAL, Leden 2014, [Online; navštíveno 10.05.2019].
URL https://hal.inria.fr/hal-00932836/file/CRDTs_SSS-2011.pdf/

[54] Strauch, C.: *NoSQL Databases*. [Online; navštíveno 25.11.2018].
URL https://www.christof-strauch.de/nosqldbs.pdf

[55] team, B.: *Document Databases Explained*. [Online; navštíveno 25.11.2018].
URL http://basho.com/resources/document-databases/

[56] Tonar, J.: *Relační a nerelační datový model v kontextu business intelligence*. Červen 2013, [Online; navštíveno 25.11.2018].
URL http://www.web-integration.info/cs/blog/relacni-a-nerelacni-datovy-model-v-kontextu-business-intelligence/

[57] Tutorials, T. J.: *What Is an Object?* [Online; navštíveno 20.11.2018].
URL https://docs.oracle.com/javase/tutorial/java/concepts/object.html

[58] Valášek, M.: *Čo znamená elektronická správa dokumentovTEX*. [Online; navštíveno 05.11.2018].
URL https://www.etrend.sk/trend-archiv/rok-/cislo-Marec/co-znamena-elektronicka-sprava-dokumentov.html

[59] Vieru, T.: *KSC Gets Orion Mock-Up for Testing*. Leden 2009, [Online; navštíveno 2.02.2019].
URL https://news.softpedia.com/news/KSC-Gets-Orion-Mock-Up-for-Testing-103300.shtml

[60] Wodehouse, C.: *What is Middleware?* Middleware.org, [Online; navštíveno 15.04.2019].
URL http://web.archive.org/web/20120629211518/http://www.middleware.org/whatis.html

[61] Wodehouse, C.: *Express.js: A Server-Side JavaScript Framework*. Upwork Global Inc., Červen 2015, [Online; navštíveno 6.04.2019].
URL https://www.upwork.com/hiring/development/express-js-a-server-side-javascript-framework/

[62] Zendulka, J.: *Databázové systémy a návrh databází – 2 Konceptuální modelování a návrh databáze* . [Online; navštíveno 13.11.2018].
URL https://www.fit.vutbr.cz/study/courses/DSI/public/pdf/nove/2_2.pdf

[63] Zendulka, J.: *Pokročilé databázové systémy – 2 Objektově-relační databáze*. [Online; navštíveno 16.11.2018].
URL https://www.fit.vutbr.cz/study/courses/PDB/private/lectures/2_ORdatabaze_2.pdf

[64] Zendulka, J.; Rudolfová, I.: *Databázové systémy (IDS) – Studijní opora*. Jul 2006, [Offine; výukový text v pdf].

[65] Čápka, D.: *Lekce 1 - Úvod do UML*. Duben 2013, [Online; navštíveno 22.02.2019].
URL
https://www.itnetwork.cz/navrh/uml/uml-uvod-historie-vyznam-a-diagramy

[66] Švec, M.: Závěrečné práce studentů z předmětu Vybrané problémy informačních systémů: Objektové databáze. Technická zpráva, FIT VUT v Brně, Jun 2003, [Online; navštíveno 15.11.2018].
URL http://www.fit.vutbr.cz/study/courses/VPD/public/0203VPD-Svec.pdf

Příloha A

Příklad návrhu šablon k dokumentům

ASSUMPTION LOG

PROJECT NAME

COMPANY NAME
STREET ADDRESS
CITY

DATE

The Assumption Log is a document which the project manager and team use to capture, document, and track assumptions throughout a project's lifecycle. Assumptions are an important part of any project. Assumptions usually require some type of follow-up or validation in order to determine whether or not they will impact the project. Many assumptions may actually be project risks, or may become risks during the life of the project. In addition to creating an assumption log, be sure to perform proper risk management on your project with a risk management plan and risk register. The assumption log should be used to augment your risk register - it should never be used in place of the risk register.

Each assumption should have an owner or team member responsible for following up and validating the assumption. The Assumption Log assigns each assumption an ID or reference number, a name and description of each assumption, responsible person, due date, status, closure date, and any actions which might be required as part of the follow-up or validation.

The Assumption Log should be updated as items are closed or more information is obtained. The project team should also review the Assumption Log regularly to ensure all project team members are informed and any additional actions or information is captured.

Standard Assumption Log Template:

Assumption Log						
Project:					**Date:**	
ID	**Category**	**Assumption**	**Responsibility**	**Due Date**	**Status**	**Actions**
Each assumption should have a corresponding ID number.	This should list what portion of the project the assumption impacts.	Define the assumption in this column.	Assumptions should be assigned to a team member to validate.	This is the date the assumption should be validated by.	This tracks the assumption validation and if it's open or closed.	Any actions associated with the assumption or validating the assumptions should be listed here.

Example with Sample Data:

Assumption Log						
Project:					**Date:**	
ID	**Category**	**Assumption**	**Responsibility**	**Due Date**	**Status**	**Actions**
001	Manufacturing	There is adequate capacity on manufacturing lines to produce prototypes for the project.	J. Brown	6/1/20xx	Open	J. Brown to meet with operations manager on 4/15 to discuss line capacity.
002	Design	Cable dimensions will not deviate significantly from existing product lines.	P. White	5/15/20xx	Open	Design team is currently verifying planned cable dimensions.
003	Supply	There is adequate warehousing space for additional materials needed for this product.	T. Black	5/1/20xx	Closed	T. Black has verified that there is ample warehousing space available for the Xwave Fiber Project.
004	Planning/ Execution	All project resources will remain available throughout the project lifecycle.	A. Green	4/15/20xx	Open	A. Green will meet with Sponsor and all functional managers on 3/13 to discuss this topic.

Assumption Log						
Project:					**Date:**	
ID	**Category**	**Assumption**	**Responsibility**	**Due Date**	**Status**	**Actions**

Obrázek A.1: Návrh šablony k dokumentu *Assumption Log*.

STAKEHOLDER REGISTER

PROJECT NAME

COMPANY NAME
STREET ADDRESS
CITY

DATE

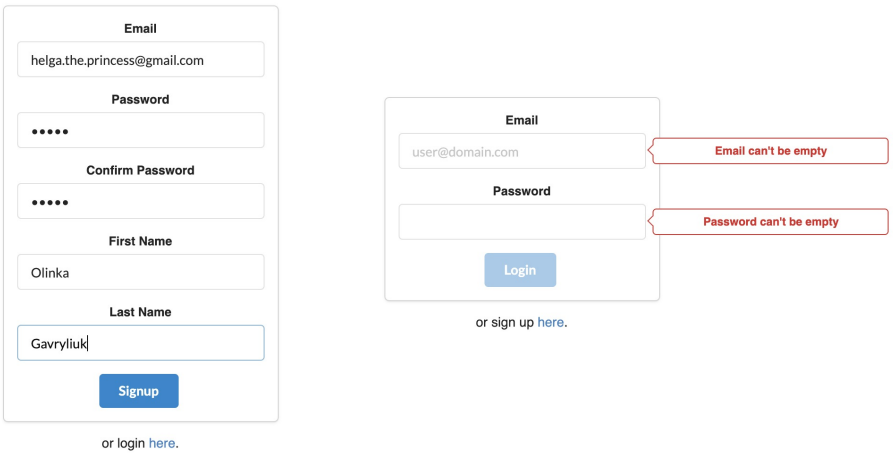
The Stakeholder Register is created by the project manager; however, it may be started by the project sponsor.
The Stakeholder Register is used when identifying stakeholders. It's merely a place for listing them all. The register is an input to performing stakeholder analysis and for creating a Stakeholder Management Plan. Some project managers prefer to keep the stakeholder register separate from the Stakeholder Analysis document. Unlike stakeholder analysis documentation, the Stakeholder Register can be handed out without sharing sensitive information. Like most project management documents, the Stakeholder Register should be consulted and updated throughout the project life cycle.

Stakeholder Register				
Project Name			**Date**	
Project Manager			**Document Number**	
Project Owner/Client				
#	**Stakeholder Name**	**Title and Project Role**	**Contact information**	**Notes**
1				
2				
3				
4				
5				
6				

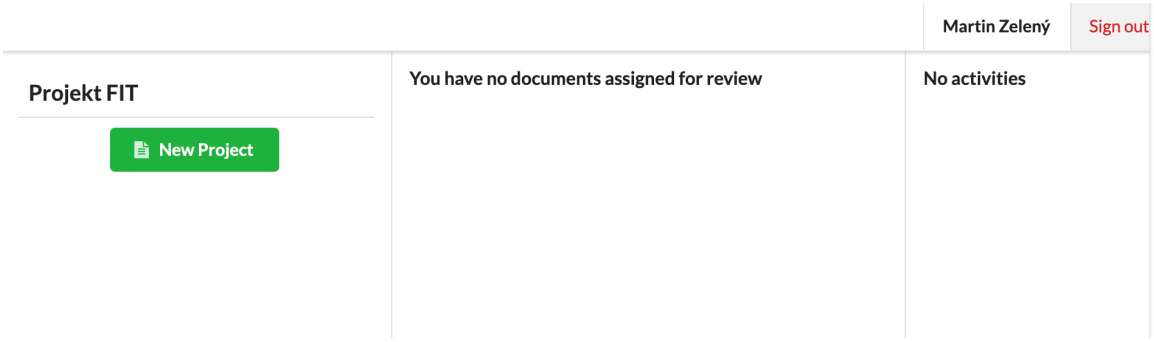
Obrázek A.2: Návrh šablony k dokumentu *Stakeholder register*.

Příloha B

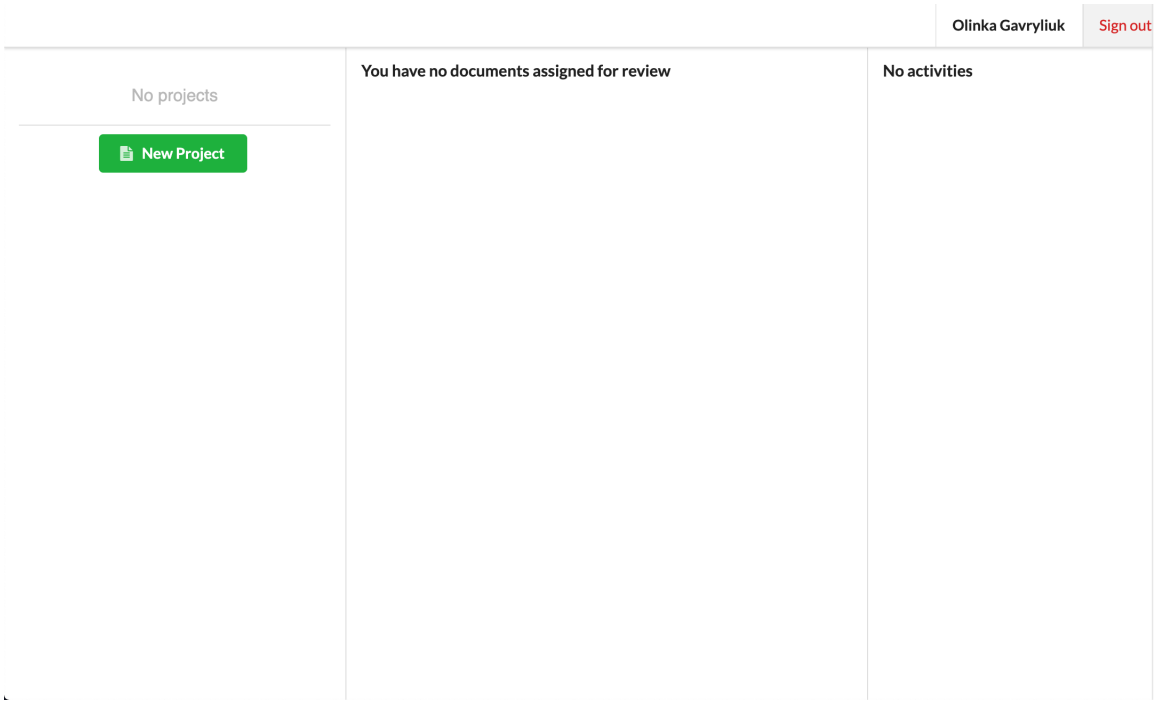
Ukázky z výsledné aplikace



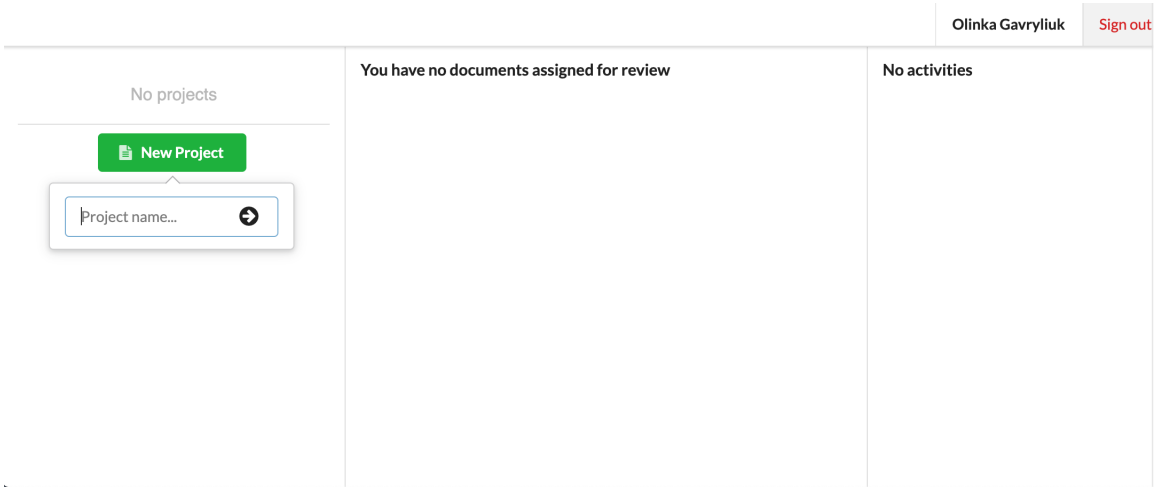
Obrázek B.1: Ukázka registrace a přihlášení do aplikace.



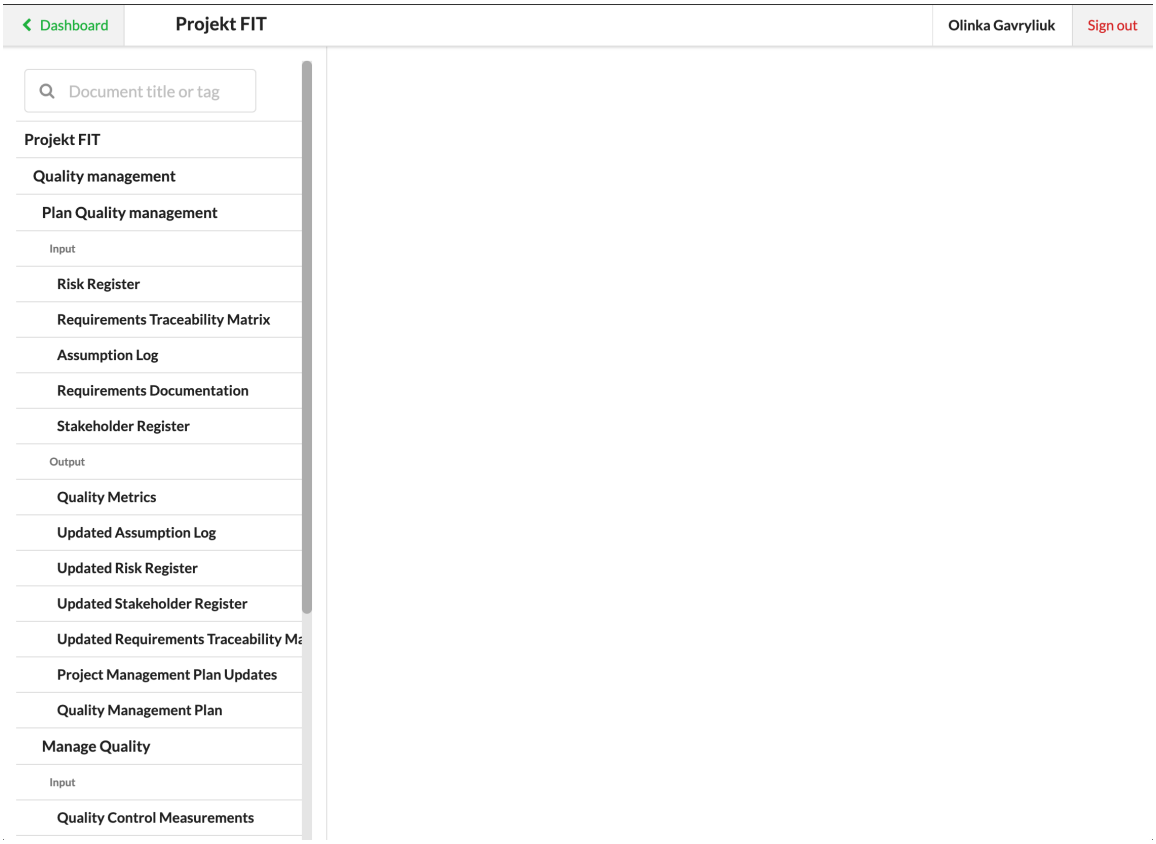
Obrázek B.2: Ukázka přiřazených projektů k danému uživateli (levý sloupec v aplikaci).



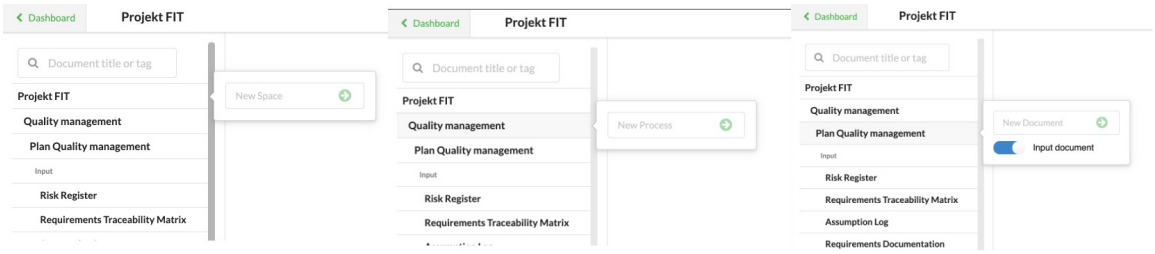
Obrázek B.3: Ukázka případu, kdy uživatel nemá přiřazené žádné projekty („prázdný“ levý sloupec v aplikaci).



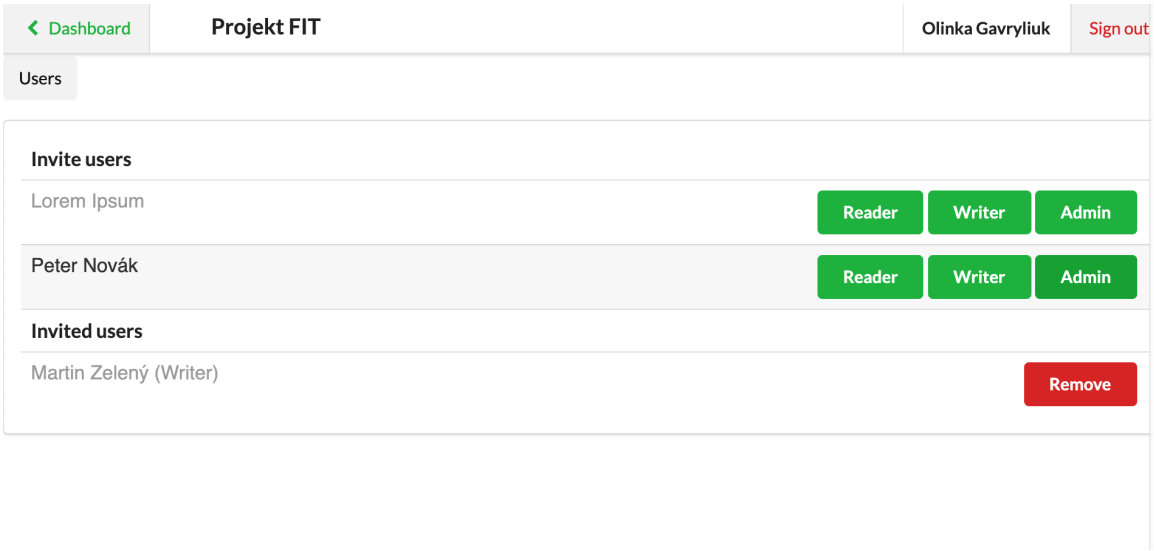
Obrázek B.4: Ukázka vytvoření vlastního projektu uživatelem.



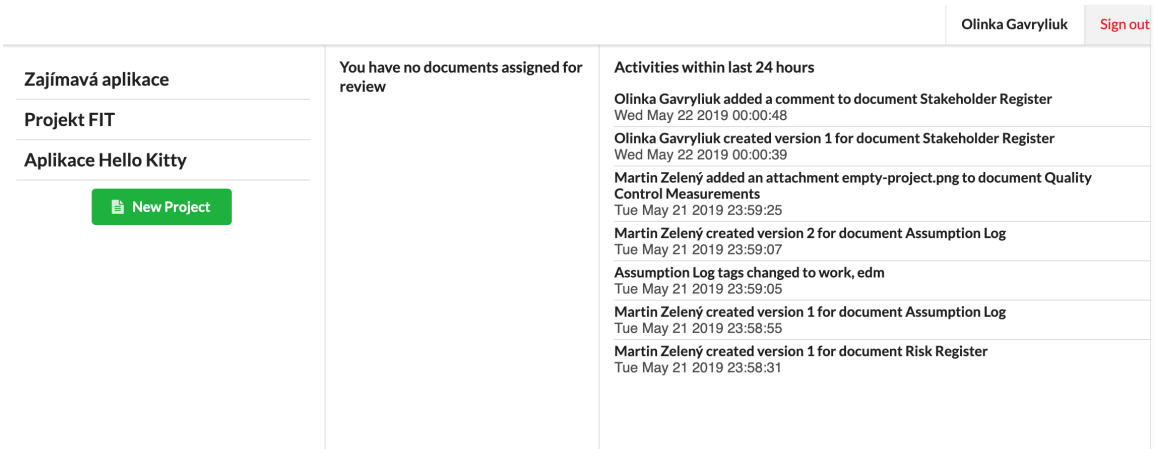
Obrázek B.5: Ukázka struktury nově vytvořeného projektu.



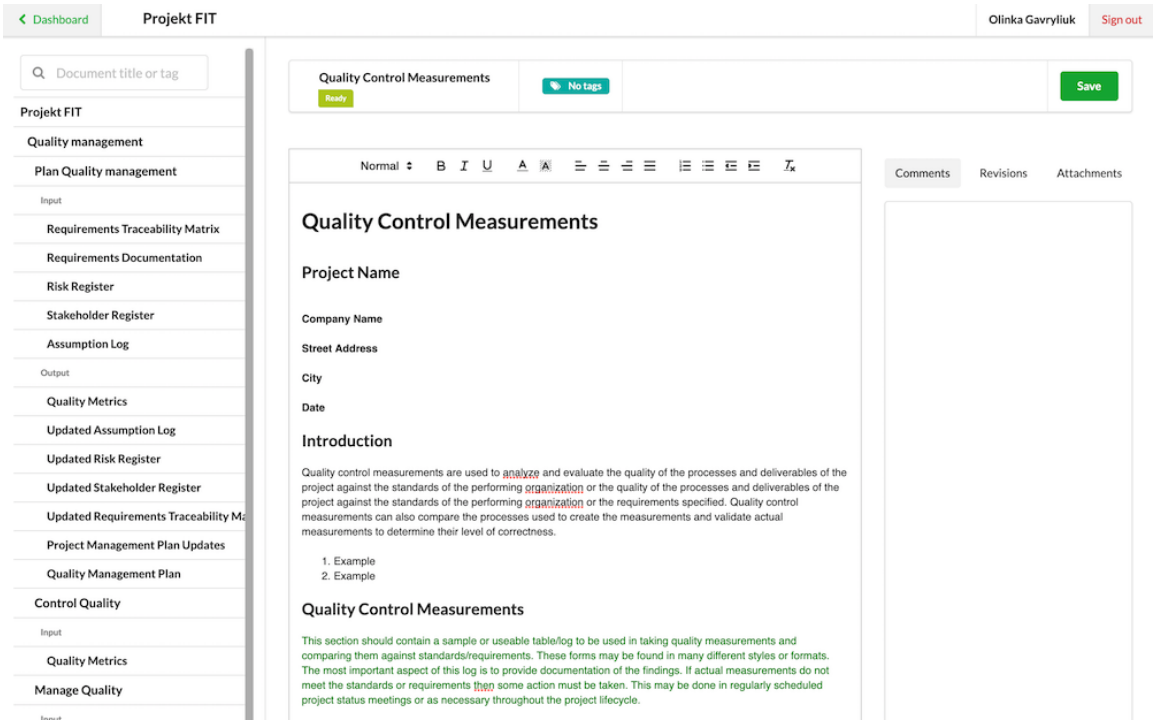
Obrázek B.6: Ukázka vytvoření nové oblasti, procesu a dokumentu v projektě.



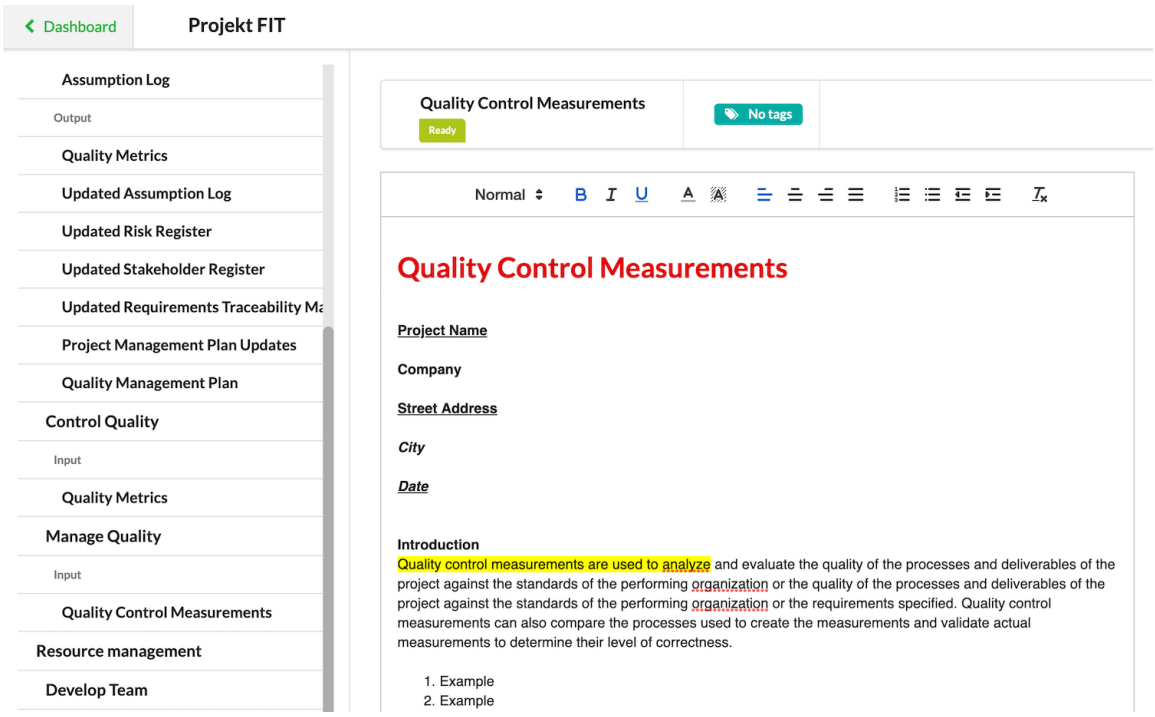
Obrázek B.7: Ukázka přihlášení dalších uživatelů do projektu a výběr rolí pro ně.



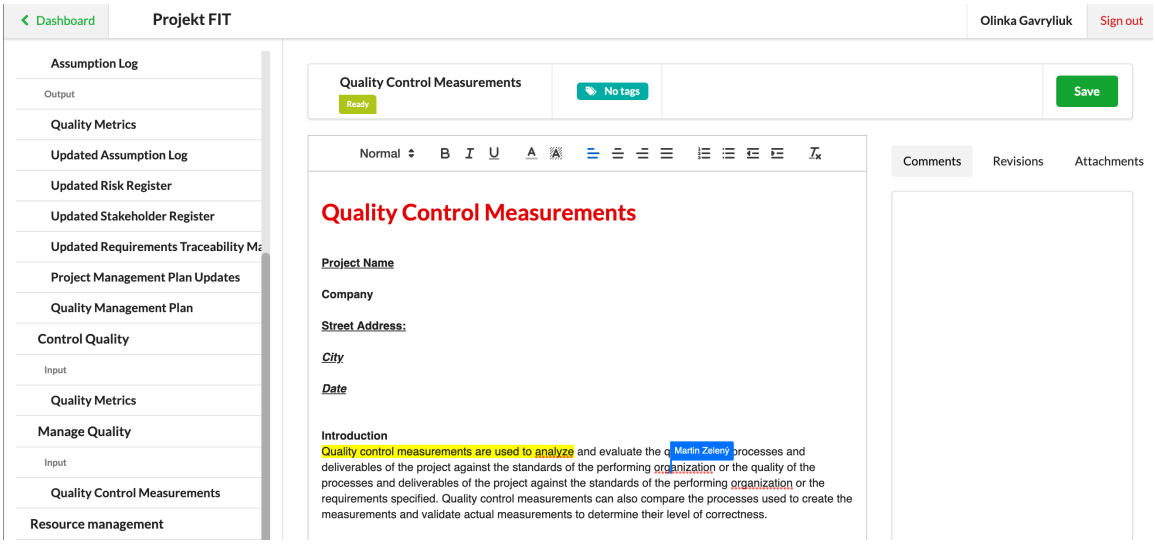
Obrázek B.8: Ukázka dashboardu s zobrazenými aktivitami v aplikaci.



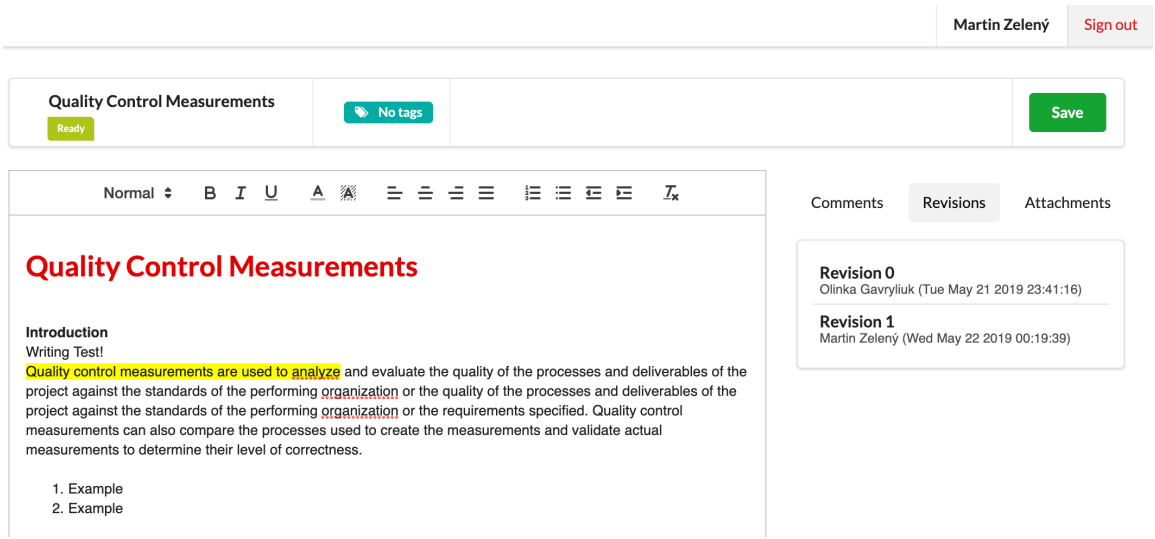
Obrázek B.9: Ukázka předem vytvořené dokumentové šablony nějakého dokumentu v aplikaci.



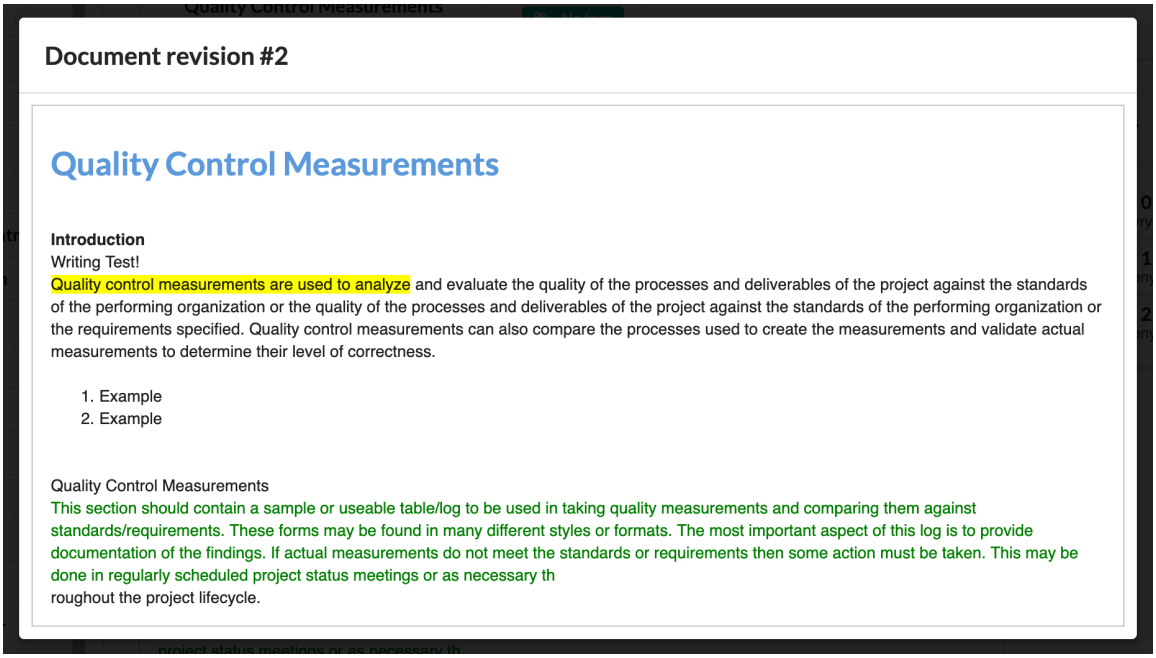
Obrázek B.10: Ukázka práce s obsahem dokumentu v textovém editoru aplikace.



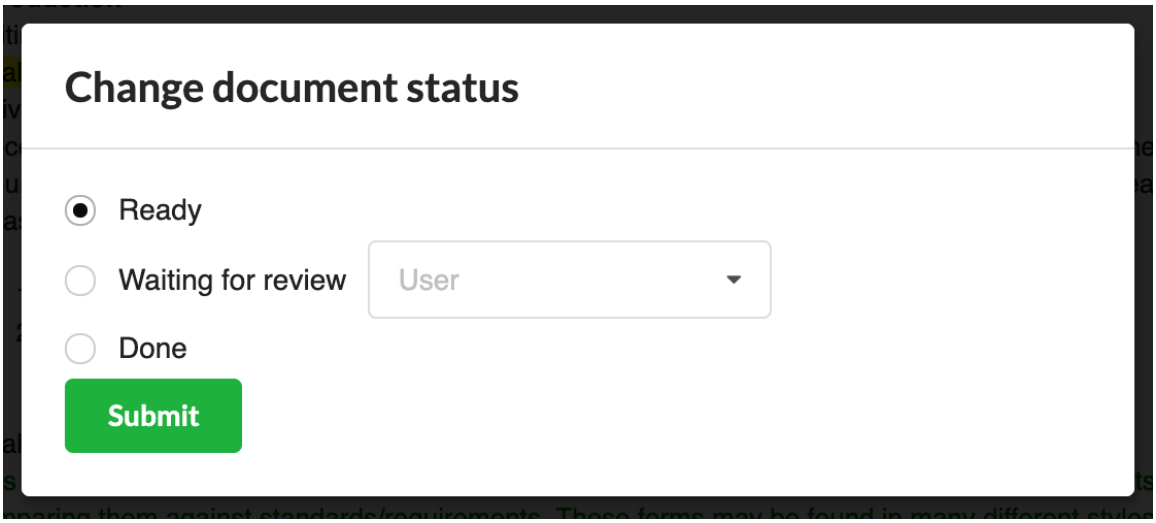
Obrázek B.11: Zobrazení kolaborativní spolupráce několika uživatelů nad jedním dokumentem.



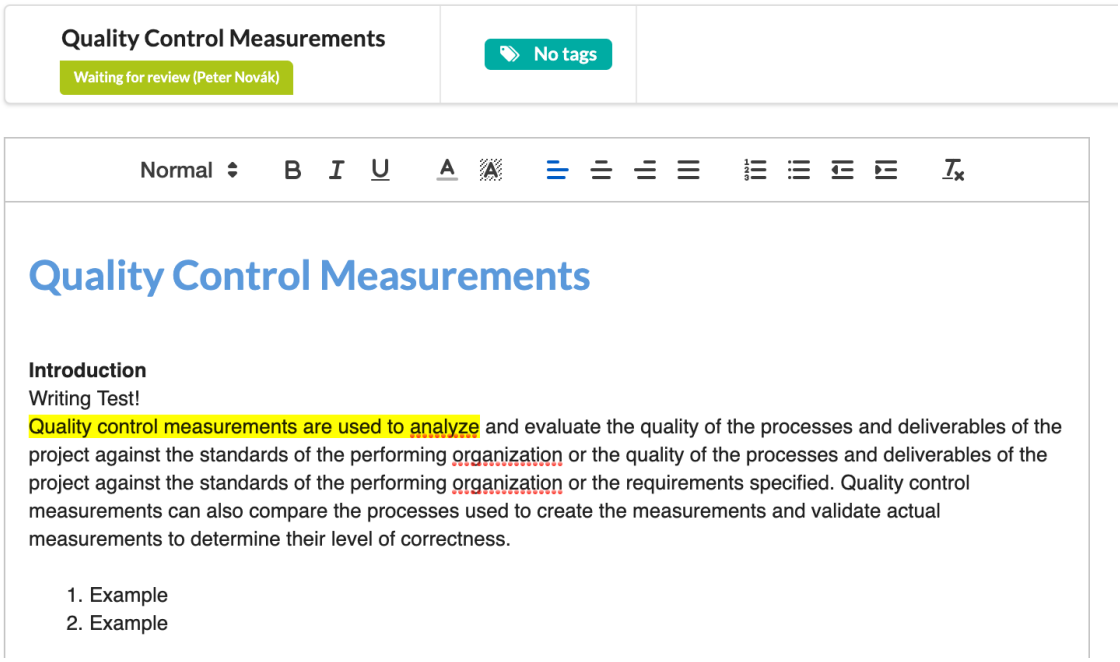
Obrázek B.12: Zobrazení záznamu verzí dokumentu.



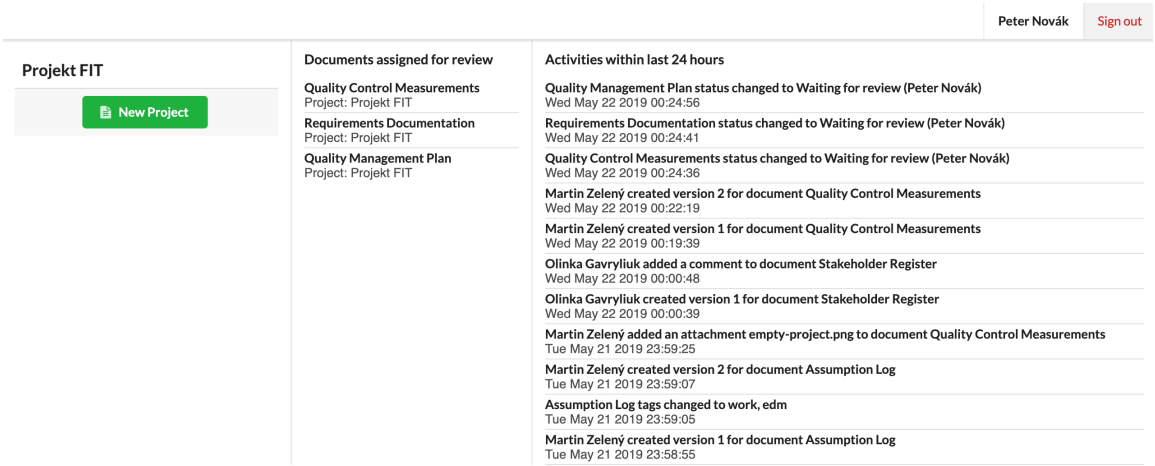
Obrázek B.13: Zobrazení verze vybraného dokumentu.



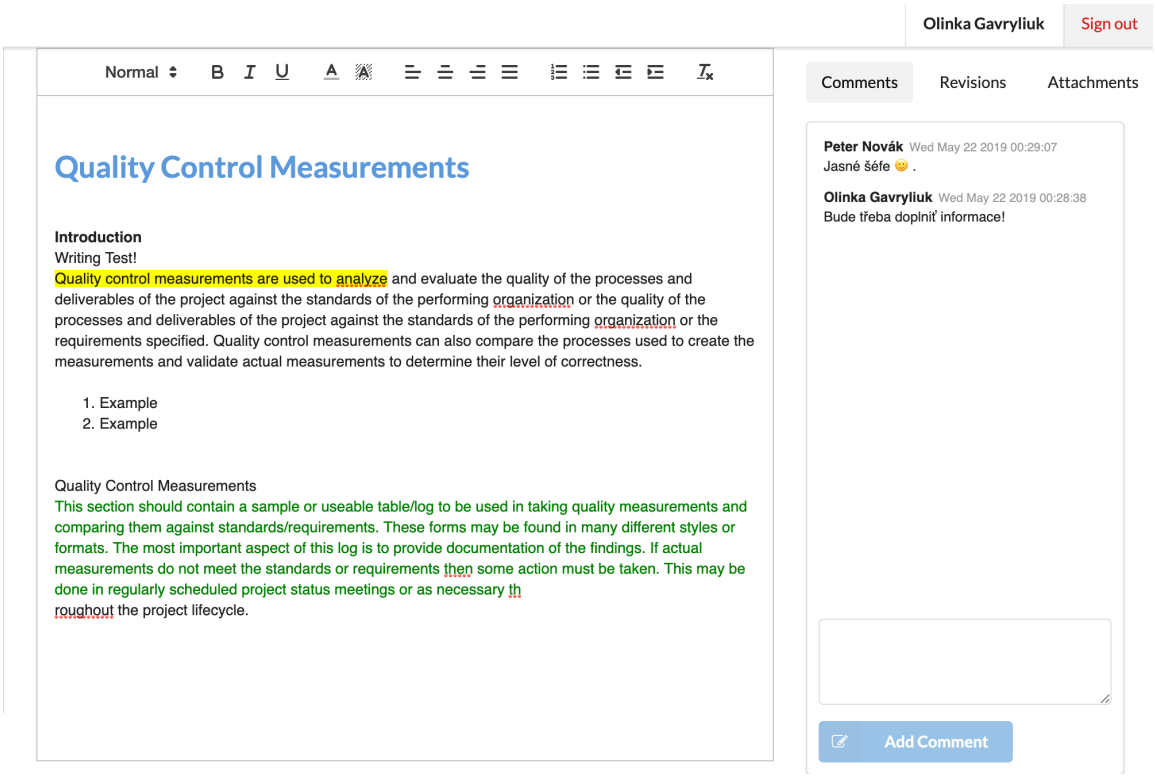
Obrázek B.14: Zobrazení menu stavů dokumentu.



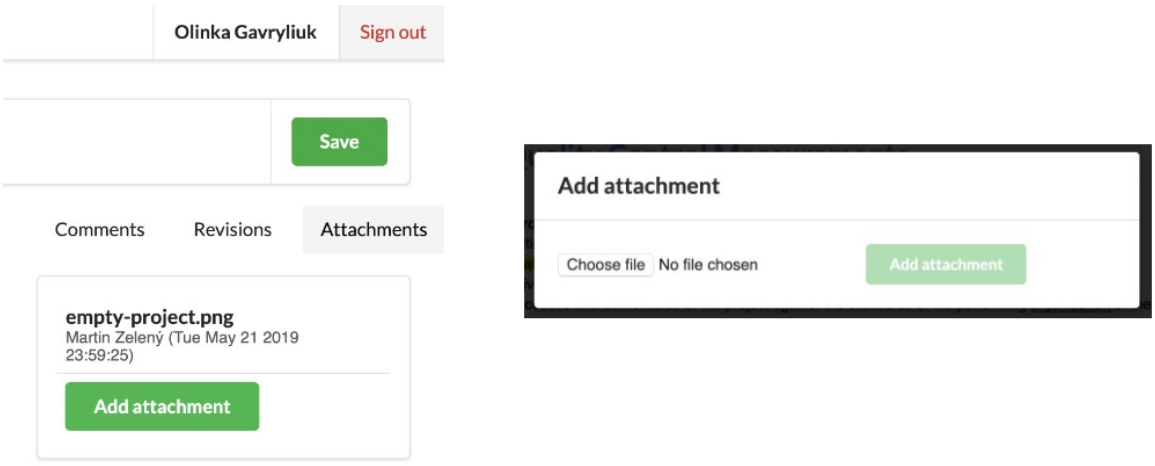
Obrázek B.15: Zobrazení stavu *Waiting for review* s přiřazeným uživatelem.



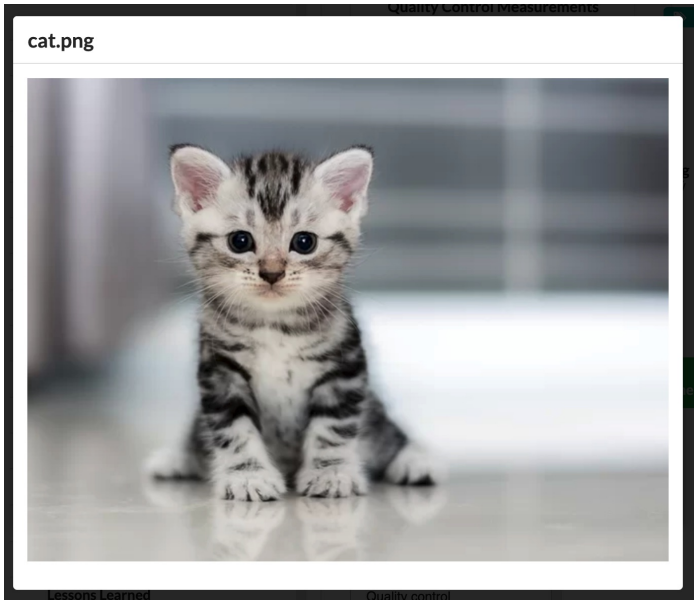
Obrázek B.16: Zobrazení přiřazeného dokumentu na dashboardu zodpovědného uživatele.



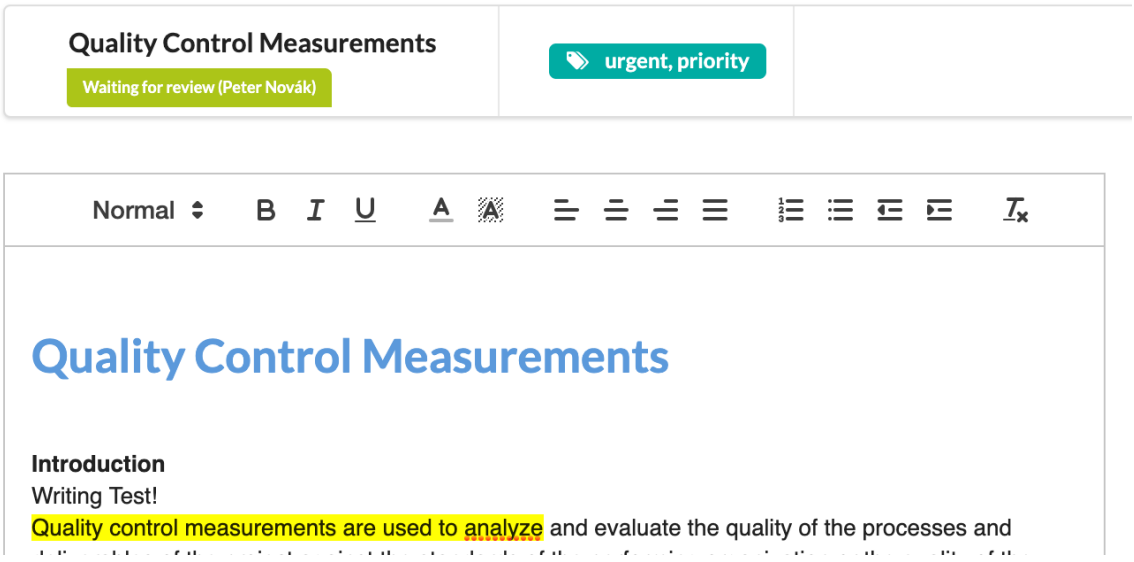
Obrázek B.17: Zobrazení komentářů k dokumentům.



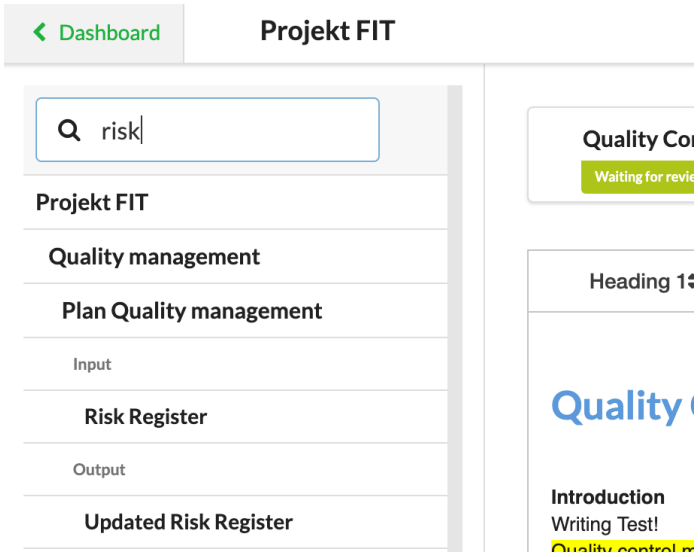
Obrázek B.18: Zobrazení možnosti nahrávat přílohy k dokumentům.



Obrázek B.19: Zobrazení přílohy k dokumentu.



Obrázek B.20: Zobrazení tagů dokumentu.



Obrázek B.21: Zobrazení vyhledávaní dokumentů.

Příloha C

Testovací scénáře

ID: #5	Nahrání přílohy
Název:	Kontrola správné funkčnosti nahrávání příloh
Vstupní podmínky:	1. Uživatel je přihlášen 2. Uživatel je v nově vytvořeném projektu s původními šablonami 3. Uživatel se nachází v dokumentu s názvem: "Requirements documentation"
Postup:	1. Klikneme na záložku: "Attachments" 2. Klikneme na tlačítko: "Add attachment" 3. Ve vyskakovacím okně klikneme na tlačítko: "Choose file" 4. Do plovoucího vstupního pole napíšeme název: "Test" 5. Vybereme zvolený obrázek a potvrdíme systémový dialog 6. Klikneme na potvrzovací tlačítko: "Add atachment"
Očekáváný výsledek:	1. V seznamu příloh u dokumentu se nachází příloha s obrázkem 2. Příloha obsahuje autora a aktuální datum 3. Po kliknutí na přílohu se v plovoucím okně zobrazí náhled obrázku

Tabulka C.1: Testovací případ #5.

ID: #6	Neexistující email a heslo
Název:	Kontrola, zda se jde přihlásit se špatným jménem a heslem
Vstupní podmínky:	1. Webová aplikace běží na úvodní stránce s přihlášením 2. Formulář na zadání jména a hesla je prázdný 3. Uživatel "random@email.cz"neexistuje
Postup:	1. Klikneme na vstupní pole pro zadání emailu a zadáme: "random@email.cz" 2. Klikneme na vstupní pole pro zadání hesla a zadáme: "password" 3. Klikneme na tlačítko "Login"
Očekáváný výsledek:	1. Uživatel není přihlášen 2. Zobrazí se hláška: "Invalid email or password."

Tabulka C.2: Testovací případ #6.

ID: #7	Přihlášení
Název:	Kontrola, zda se jde přihlásit s existujícím jménem a heslem
Generována data:	1. `username` - "user"+timestamp+"email.cz" 2. `password` - "password"
Vstupní podmínky:	1. Webová aplikace běží na úvodní stránce s přihlášením 2. Uživatel se registroval s emailem: {username} a heslem: {password}
Postup:	1. Klikneme na vstupní pole pro zadání emailu a zadáme: {username} 2. Klikneme na vstupní pole pro zadání hesla a zadáme: {password} 3. Klikneme na tlačítko "Login"
Očekáváný výsledek:	1. Uživatel je přihlášen 2. Nezobrazí se hláška: "Invalid email or password."

Tabulka C.3: Testovací případ #7.

ID: #8	Přihlášení se špatným heslem
Název:	Kontrola, zda se není možné přihlásit s existujícím účtem a špatným heslem
Generována data:	1. `username` - "user"+timestamp+"email.cz" 2. `password` - "password"
Vstupní podmínky:	1. Webová aplikace běží na úvodní stránce s přihlášením 2. Uživatel se registroval s emailem: {username} a heslem: {password}
Postup:	1. Klikneme na vstupní pole pro zadání emailu a zadáme: {username} 2. Klikneme na vstupní pole pro zadání hesla a zadáme: {password} 3. Klikneme na tlačítko "Login"
Očekáváný výsledek:	1. Uživatel není přihlášen 2. Zobrazí se hláška: "Invalid email or password."

Tabulka C.4: Testovací případ #8.

ID: #9	Vytvoření projektu
Název:	Kontrola správného vytvoření projektu s předvolenými šablonami
Vstupní podmínky:	1. Uživatel je přihlášen
Postup:	1. Klikneme na tlačítko: "New Project"- Objeví se plovoucí okno se vstupem 2. Klikneme na vstupní pole pro zadání názvu projektu a zadáme: "Sample project" 3. Na pravé straně plovoucího okna klikneme ikonu šipky v kroužku 4. Klikneme na nově vytvořený projekt s názvem: "Sample project"
Očekáváný výsledek:	1. Uživatel se nachází na obrazovce projektu 2. V seznamu se nacházejí předvytvořené oblasti, procesy se vstupními a výstupními dokumenty

Tabulka C.5: Testovací případ #9.

ID: #10	Vyhledávání podle názvu dokumentu
Název:	Kontrola správné funkčnosti vyhledávání podle názvu dokumentu
Vstupní podmínky:	1. Uživatel je přihlášen 2. Uživatel je v nově vytvořeném projektu s původními šablonami 3. Mezi šablonami se nachází dokument: "Stakeholder Register" 4. Mezi šablonami se nachází dokument: "Assumption Log"
Postup:	1. Klikneme na vstupní pole pro vyhledávání a bez potvrzení zadáme: "Stakeholder"
Očekáváný výsledek:	1. V seznamu dokumentů se nachází dokument: "Stakeholder Register" 2. V seznamu dokumentů se nenachází dokument: "Assumption Log"

Tabulka C.6: Testovací případ #10.

ID: #11	Vytvoření a vyhledání tagu
Název:	Kontrola správné funkčnosti vytvoření a vyhledání tagu
Vstupní podmínky:	1. Uživatel je přihlášen 2. Uživatel je v nově vytvořeném projektu s původními šablonami 3. Mezi šablonami se nachází dokument: "Risk Register" 4. Mezi šablonami se nachází dokument: "Quality Metrics"
Postup:	1. Klikneme na dokument s názvem: "Quality Metrics" 2. Klikneme na značku: "No tags" 3. Do plovoucího vstupního pole napíšeme název: "Test" 4. Potvrdíme plovoucí okno 5. Klikneme na vstupní pole pro vyhledávání a bez potvrzení zadáme: "Test"
Očekáváný výsledek:	1. V seznamu dokumentů se nachází dokument: "Quality Metrics" 2. V seznamu dokumentů se nenachází dokument: "Risk Register"

Tabulka C.7: Testovací případ #11.

ID: #12	Přidělení dokumentu na revizi
Název:	Kontrola správné funkčnosti přidělení dokumentu na revizi
Vstupní podmínky:	1. Uživatel1 je přihlášen 2. Uživatel1 je v nově vytvořeném projektu s původními šablonami 3. Uživatel1 se nachází v dokumentu s názvem: "Quality Management Plan" 4. Uživatel2 je přihlášen 5. Uživatel2 je přizvaný do dokumentu 6. Uživatel2 se nachází na obrazovce: "Dashboard"
Postup: Uživatel1	1. Uživatel1 klikne na stav dokumentu: "Ready" 2. Klikneme na možnost: "Waiting for review" 3. Klikneme na kontextové menu a vybereme uživatele: "Uživatel2" 4. Klikneme na tlačítko: "Submit"
Očekáváný výsledek:	1. Uživatel2 vidí v levém sloupci přidělený dokument s názvem: "Quality Management Plan"

Tabulka C.8: Testovací případ #12.

ID: #13	Právo "reader"
Název:	Kontrola správné funkčnosti přístupového práva "reader"
Vstupní podmínky:	1. Uživatel byl přizván do vytvořeného projektu "Sample Project"s právy "reader" 2. Uživatel se nachází v projektu "Sample Project" 3. Projekt obsahuje původní šablony
Postup:	1. Klikneme na dokument: "Ready" 2. Klikneme na možnost: "Done" 3. Klikneme na tlačítko: "Submit"
Očekáváný výsledek:	1. Uživatel není schopný editovat dokument 2. Uživatel nemůže zaslat komentář 3. Uživatel vidí projekt 4. Uživatel vidí dokumenty 5. Uživatel vidí revize

Tabulka C.9: Testovací případ #13.

ID: #14	Právo "writer"
Název:	Kontrola správné funkčnosti přístupového práva "reader"
Vstupní podmínky:	1. Uživatel byl přizván do vytvořeného projektu "Sample Project"s právy "writer" 2. Uživatel se nachází v projektu "Sample Project" 3. Projekt obsahuje původní šablony
Postup:	1. Klikneme na dokument: "Risk Register"
Očekáváný výsledek:	1. Uživatel je schopný editovat dokument 2. Uživatel může zaslat komentář 3. Uživatel nemůže přizvat nikoho do projektu

Tabulka C.10: Testovací případ #14.

ID: #15	Právo "admin"
Název:	Kontrola správné funkčnosti přístupového práva "admin"
Vstupní podmínky:	1. Uživatel byl přizván do vytvořeného projektu "Sample Project"s právy "admin" 2. Uživatel se nachází v projektu "Sample Project" 3. Projekt obsahuje původní šablony
Postup:	1. Klikneme na tlačítko: "Dashboard" 2. Najedeme kurzorem myši na projekt""Sample Project" 2. V plovoucím okně klikneme na tlačítko: "Edit"
Očekáváný výsledek:	1. Uživatel vidí seznam uživatelů k pozvání 2. Uživatel může přizvat jakéhokoliv uživatele s jakoukoliv roli

Tabulka C.11: Testovací případ #15.

ID: #16	Zobrazení aktivity
Název:	Kontrola správné funkčnosti vytvoření aktivity
Vstupní podmínky:	1. Uživatel je přihlášen 2. Uživatel je v nově vytvořeném projektu s původními šablonami 3. Uživatel vytvořil tři prázdné dokumenty 4. Uživatel se nachází v projektu
Postup:	1. Klikneme na tlačítko: "Dashboard"
Očekáváný výsledek:	1. V pravém sloupci se nacházejí tři záznamy o aktivitě se jménem aktuálního uživatele a aktuálním datem

Tabulka C.12: Testovací případ #16.

ID: #17	Vytvoření komentáře
Název:	Kontrola správné funkčnosti vytváření komentářů
Vstupní podmínky:	1. Uživatel je přihlášen 2. Uživatel je v nově vytvořeném projektu s původními šablonami 3. Uživatel se nachází v dokumentu s názvem: "Updated Stakeholder Register"
Postup:	1. Klikneme na záložku: "Comments" 2. Klikneme na vstupní pole pro zadání komentáře a zadáme: "Ahoj EDM" 3. Klikneme na tlačítko: "Add Comment"
Očekáváný výsledek:	1. V seznamu komentářů se nachází komentář s aktuálním datem, autorem a textem: "Ahoj EDM"

Tabulka C.13: Testovací případ #17.

Příloha D

Obsah paměťového média

Obsah přiloženého CD má následující adresářovou strukturu:

- **README.md** - návod ke spuštění aplikace
- **source.zip** - zdrojové soubory implementované aplikace
- **templates/** - předvytvořené dokumentové šablony
- **thesis.pdf** - tato diplomová práce ve formátu PDF