FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Aplikace pro projekt Open-Data
Student:	Kyrylo Bulat
Vedoucí:	Ing. Miroslav Balík, Ph.D.
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2018/19

Pokyny pro vypracování

Cílem práce je návrh a implementace nového webového portálu pro přístup k původně nedostupným datům pro podporu projektu "Open data“ v České Republice.

1. Analyzujte stávající řešení pro sběr a zpracování dat z webů ministerstev České Republiky.
2. Analyzujte hotové portály pro práci s open daty (volně dostupnými daty). Získejte požadavky na nový portál, který bude pracovat s open daty.
3. Navrhněte a doplňte následující funkcionality:
 - zobrazení seznamu veřejných zakázek.
 - zobrazení detailu zakázky.
 - zobrazení seznamu dodavatelů a odběratelů.
 - zobrazení detailu dodavatele/odběratele.
 - veřejné API pro přístup k Open data.
 - zobrazení statistik.
4. Implementujte tyto funkcionality.
5. Otestujte navržené a implementované funkcionality.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

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

V Praze dne 12. února 2018

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Aplikace pro projekt Open-Data

Kyrylo Bulat

Katedra softwarového inženýrství
Vedoucí práce: Ing. Miroslav Balík, Ph.D.

9. května 2018

Poděkování

Chtěl bych poděkovat Ing. Miroslavu Balíku, Ph.D., vedoucímu bakalářské práce, za snahu a pomoc při vytvoření této práce. Mé díky také patří Ing. Marku Sušickému za pomoc s výběrem tématu bakalářské práce a uvedením do kontextu tohoto tématu. Děkuji svým rodičům za podporu během celého studia.

Prohlášení

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

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. Dále prohlašuji, že jsem s Českým vysokým učením technickým v Praze uzavřel dohodu, na základě níž se ČVUT vzdalo práva na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona. Tato skutečnost nemá vliv na ust. § 47b zákona č. 111/1998 Sb., o vysokých školách, ve znění pozdějších předpisů.

V Praze dne 9. května 2018 .....

Odkaz na tuto práci

Bulat, Kyrylo. *Aplikace pro projekt Open-Data*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.

Abstrakt

Ve své práci se zaměřuji na analýzu stávajících řešení pro přístup k informacím o veřejných zakázkách v České republice, vytvoření portálu pro práci s daty, který by podporoval rozvoj projektu Otevřená data v České republice.

Cílem práce je návrh, implementace a testovaní nového webového portálu, který by poskytoval přístup k novým otevřeným datům, které nebyly předtím volně dostupné. Pro vytvoření webového portálu byla vyvinuta webová aplikace v programovacím jazyce Java verze 8 s použitím frameworku Spring a Thymeleaf.

Vytvořené řešení poskytuje možnost používat aplikaci běžnému uživateli a splňuje požadavky uvedené v zadaní bakalářské práci.

Klíčová slova otevřená data, analýza, případ užití, Spring, Spring Boot, Thymeleaf, unit testování, návrh aplikace

Abstract

Bachelor thesis focuses on analyzing existing solutions for access to information about public procurement in the Czech Republic and also on creating a data portal that would support the development of Open Data project in the Czech Republic.

The goal is to design, implement and test a new web portal that would provide access to new open data that was not previously freely available. A web application was developed in programming language Java version 8 using the Spring and Thymeleaf frameworks.

The created solution provides the ability to use the application to a regular user and meets the requirements specified in the bachelor's thesis.

Keywords open data, analysis, use case, Spring, Spring Boot, Thymeleaf, unit testing, application desing

Obsah

Seznam obrázků

Úvod

V dnešní době je kladen velký důraz na snadný přístup k datům a jejich použití pro vlastní účely. Téměř v každé oblasti života jsou různé portály, pomocí kterých lze najít, prohlédnout a stáhnout data, které nás zajímají. Projekt Otevřená data se zabývá vybudováním otevřené datové infrastruktury v České republice. Cílem projektu je zvýšit využití veřejně dostupných dat o zakázkách, smlouvách a fakturách firem v České republice.

Hlavním sponzorem a iniciátorem projektu Otevřená data je Fond Otakara Motejla, který byl založen v roce 2012. Fond podporuje neveřejné organizace, které přispívají k rozvoji právního státu a omezení korupce. Jedním z nejznámějších projektů Fondu je projekt Otevřená data a od roku 2017 Fond dosáhl prosazení zákona o svobodném přístupu k informacím.

První myšlenkou pro podporu a rozvoj projektu bylo navržení a implementace jednoduchého API, které by poskytovalo stáhnutá data ve specifikovaném formátu. Toto nezajišťuje snadný přístup pro všechny občany, ale jenom pro organizace a struktury, které vědí, jak s tím nadále pracovat. Cílem bakalářské práce je vytvoření webového portálu, který by umožnil jednoduše prohlédnout, analyzovat, kombinovat a stáhnout data ve formátu, který vyhovuje uživateli.

Práce obsahuje čtyři části – analýzu, návrh, implementaci a testování. První část obsahuje informaci o hotových řešeních pro veřejný přístup k datům o smlouvách, zakázkách a fakturách v České republice, popis dat, se kterými portál pracuje a způsob sbíraní dat pro jejich následující zpracování. Analýza obsahuje sběr a analýzu požadavků na nový portál. Druhá část práce obsahuje návrh aplikace a popisuje případy užití aplikace, doménový model, se kterým aplikace pracuje a návrh GUI pro portál. Třetí část se zabývá implementací webové aplikace a popisem technologií, které jsou použité při implementaci. Poslední část práce obsahuje popis testovacích scénářů a samotné testování aplikace.

Kapitola 1

Cíl práce

Cílem práce je navrhnout a vytvořit základ funkční aplikace ve formě webového portálu, který bude poskytovat přístup k novým veřejně dostupným datům z různých ministerstev České republiky.

Cíle bakalářské práce jsou:

- analýza stávajících řešení pro přístup k veřejným datům,
- analýza dat, se kterými bude pracovat nová aplikace,
- sběr a zpracování funkčních i nefunkčních požadavků,
- návrh a zpracování dokumentace ve formě diagramů,
- implementace základních modulů aplikace,
- testování aplikace.

Kapitola 2

Analýza

2.1 Analýza stávajících řešení

V současné době se projekt Otevřená data rychle v České republice rozvijí a díky tomu existují již vyvinuté portály a webové aplikace, které poskytují přístup k zdrojům informace. Následující kapitola obsahuje popis a rozbor konkurenčních aplikací.

Webová stránka `www.otevrenadata.cz` je zdrojem informací o existujících aplikacích, která se zabývá podporou projektu Otevřená data. Analýza každého portálu obsahuje základní popis a přehled funkcionalit, informaci o datech poskytovaných aplikací, výhody a nevýhody řešení, náměty na zlepšení.

2.1.1 `www.opendata.praha.eu`

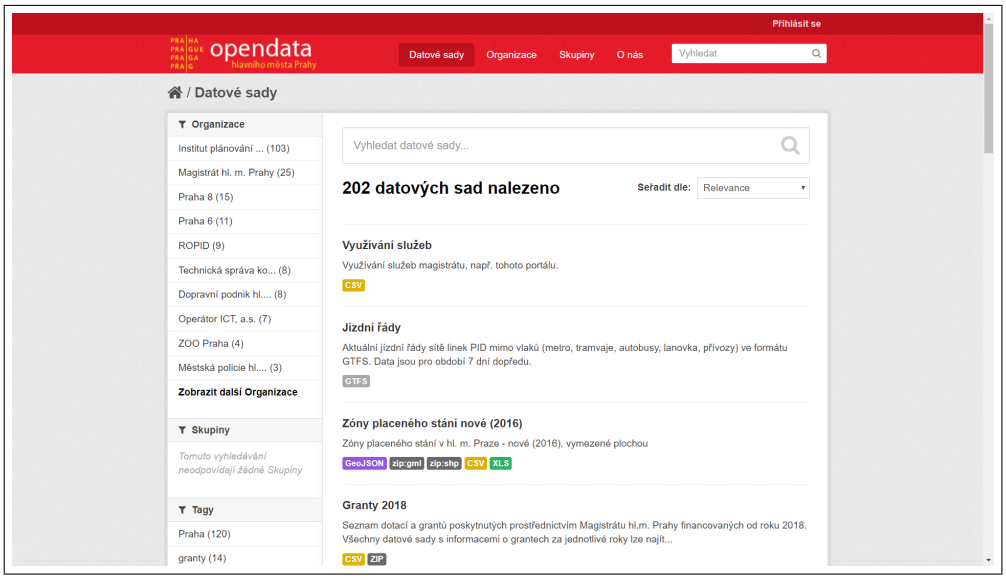
Portál Opendata hlavního města Prahy jsem vybral k analýze, protože se nachází na první pozici v odpovědi od vyhledávače Google[1] na dotaz „open data praha“. Portál `http://opendata.praha.eu` poskytuje informace z téměř 200 datových sad, které aktuálně spravují 15 organizací z hlavního města Prahy. Mezi ně patří:

- magistrát hlavního města Prahy,
- příspěvkové organizace,
- městské části Prahy,
- další související subjekty.

Data pro portál se zveřejňují ve strojově čitelných formátech.

Hlavní funkcionalitou webové aplikace je možnost prohlížení a stáhnutí dat z různých datových sad v různých formátech. Formát každé datové sady

[1] https://www.google.cz



Obrázek 2.1: Portál `www.opendata.praha.eu`

je odlišný a není unifikovaný. Mezi formáty patří ZIP: SHP[2] - 99 datových sad a ZIP: GML[3] - 96 datových sad, GeoJSON[4] – 81, jsou ale i jiné jako CSV[5], XLS, XML[6], HTML[7]. Na stránce s přehledem datových sad je možné použít sadu filtrů podle různých položek. V sadě najdeme:

- organizace,
- skupiny,
- klíčového slova,
- formátu,
- licence.

Z přehledu datových sad je možné přejít na detail vybrané sady. Detail obsahuje detailnější informaci o datové sadě ve formě tabulky a seznam dostupných formátů pro její stáhnutí.

Hlavní menu webové aplikace obsahuje položku „Organizace“, která vede na stránku se seznamem organizací, které publikují dané datové sady. Stejně

[2] Shapefile
[3] Geography Markup Language
[4] Geographical JavaScript Object Notation
[5] Comma-separated values
[6] Extensible Markup Language
[7] HyperText Markup Language

jako u datových sad, je možné přejít na detail organizace, kde se zobrazí související datové sady k této organizaci a krátký popis organizace.

Tento portál využívá hotové softwarové řešení CKAN[8]. CKAN – je open source projekt, napsaný v programovacím jazyku Python na serverové straně a v programovacím jazyku JavaScript na frontendu, okamžitě připravený k nasazení po stáhnutí [2]. CKAN je určen poskytovatelům dat, kteří se snaží o to, aby jejich data byla volně dostupná pro všechny. Mezi nejznámějšími uživateli CKANu patří portály:

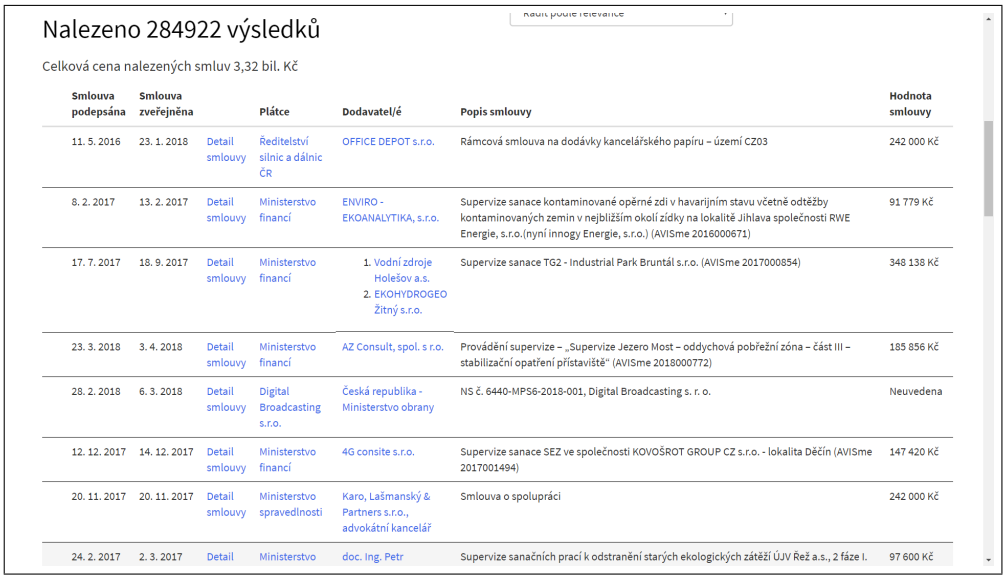
- Velké Británie: data.gov.uk,
- Evropské Unie: publicdata.eu,
- Brazílie: dados.gov.br,
- veřejné správy v Holandsku: amsterdamopendata.nl,
- Afriky: africaopendata.org,
- Německa: daten.berlin.de,
- a mnoho dalších.

Nevýhody portálu:

- jednoduchý(zastaralý) design,
- malý počet organizací,
- data nejsou poskytována v unifikovaném formátu,
- poskytovaná data nejsou předem předzpracována pro uživatele,
- náhledy datových sad nejsou přehledné.

Výhody portálu:

- využití známého open source projektu CKAN — jistota funkčností,
- možnost registrace a nahraní datových sad pro organizace,
- různorodost geografických formátu.



Nalezeno 284922 výsledků

Celková cena nalezených smluv 3,32 bil. Kč

Smlouva podepsána	Smlouva zveřejněna		Plátce	Dodavatel/é	Popis smlouvy	Hodnota smlouvy
11. 5. 2016	23. 1. 2018	Detail smlouvy	Ředitelství silnic a dálnic ČR	OFFICE DEPOT s.r.o.	Rámcová smlouva na dodávky kancelářského papíru – území CZ03	242 000 Kč
8. 2. 2017	13. 2. 2017	Detail smlouvy	Ministerstvo financí	ENVIRO - EKOANALYTIKA, s.r.o.	Supervize sanace kontaminované opěrné zdi v havarijním stavu včetně odtěžby kontaminovaných zemin v nejbližším okolí zídky na lokalitě Jihlava společnosti RWE Energie, s.r.o.(nyní innogy Energie, s.r.o.) (AVISme 2016000671)	91 779 Kč
17. 7. 2017	18. 9. 2017	Detail smlouvy	Ministerstvo financí	1. Vodní zdroje Holešov a.s. 2. EKOHYDROGEO Žitný s.r.o.	Supervize sanace TG2 - Industrial Park Bruntál s.r.o. (AVISme 2017000854)	348 138 Kč
23. 3. 2018	3. 4. 2018	Detail smlouvy	Ministerstvo financí	AZ Consult, spol. s r.o.	Provádění supervize – „Supervize Jezero Most – oddychová pobřežní zóna – část III – stabilizační opatření přístaviště" (AVISme 2018000772)	185 856 Kč
28. 2. 2018	6. 3. 2018	Detail smlouvy	Digital Broadcasting s.r.o.	Česká republika - Ministerstvo obrany	NS č. 6440-MPS6-2018-001, Digital Broadcasting s. r. o.	Neuvedena
12. 12. 2017	14. 12. 2017	Detail smlouvy	Ministerstvo financí	4G consite s.r.o.	Supervize sanace SEZ ve společnosti KOVOŠROT GROUP CZ s.r.o. - lokalita Děčín (AVISme 2017001494)	147 420 Kč
20. 11. 2017	20. 11. 2017	Detail smlouvy	Ministerstvo spravedlnosti	Karo, Lašmanský & Partners s.r.o., advokátní kancelář	Smlouva o spolupráci	242 000 Kč
24. 2. 2017	2. 3. 2017	Detail	Ministerstvo	doc. Ing. Petr	Supervize sanačních prací k odstranění starých ekologických zátěží ÚJV Řež a.s., 2 fáze I.	97 600 Kč

Obrázek 2.2: Portál `www.hlidacstatu.cz`

2.1.2 `www.hlidacstatu.cz`

Portál `https://www.hlidacstatu.cz` je také účastníkem projektu Otevřená data. Server vznikl v srpnu roku 2016 a bere za cíl zpřístupnit registr smluv, identifikovat plýtvání a zneužití moci na úřadech, analyzovat a umožnit veřejnosti data z rejstříku analyzovat.

„Hlídač" se neomezuje jenom na hledání smluv, ale také se zabývá poskytováním údajů o politicích a sponzorech, prohledáváním transparentních účtů, hledáním v EET[9] atd. Aktuálně (duben 2018) portál `www.hlidacstatu.cz` obsahuje 899045 platných smluv a 81932 zneplatněných smluv.

Webový portál obsahuje přehled všech smluv ve formě tabulky s základními informacemi o záznamu. Každý řádek má v sobě odkaz na detail vybraného záznamu. V detailu smlouvy je uveden den publikace, odkazy na jednací strany, záznam v registru smluv (`https://smlouvy.gov.cz`), hodnota a další základní informace. Položka „Přílohy" obsahuje odkazy na originál dokument ve formátu PDF[10] a přepracovanou kopii v textovém formátu. U každé smlouvy je stálé URL[11] tohoto záznamu na serveru „hlídače", které umožňuje stáhnout informaci ve strojově čitelném formátu JSON[12]. Pro použití a stáhnutí JSON smlouvy musí být uživatel přihlášen. Pro přihlášené uživatele webový portál nabízí větší funkcionalitu, speciálně možnost stáhnutí všech veřejně

[8] Comprehensive Knowledge Archive Network
[9] Elektronická evidence tržeb
[10] Portable Document Format
[11] Uniform Resource Locator
[12] JavaScript Object Notation

dostupných záznamů z databáze aplikace „hlídače“ přes API[13].

Portál obsahuje stránky se statistikami ve formě grafů, na kterých jsou vidět různé výsledky výpočtů, které by mohly zajímat uživatele, například „Hodnota uzavřených smluv po měsících“, „Nejaktivnější plátci podle počtu smluv“, atd. Tyto grafy mají v sobě odkazy na framework Highcharts[14], který se využívá pro konstruování interaktivních grafů. Nevýhody portálu:

- není implementováno rychlé filtrování dokumentů podle formátu a organizace,
- složitá navigace portálem,
- mezi dokumenty patří jen smlouvy, nejsou jiné typy dokumentů.

Výhody portálu:

- velký počet záznamů v databázi,
- aktuální data,
- dostupné API pro vývojáře.

2.1.3 `www.mapasamospravy.cz`

Účastníkem projektu Otevřená data je také `www.mapasamospravy.cz`. Tento webový portál se zaměřuje na téma nemovitostí. Všechny smlouvy a dokumenty, které se evidují v databázi aplikace se týkají nějaké budovy nebo nemovitosti v České republice.

Hlavní výjimečností této webové aplikace je její grafické uživatelské rozhraní. Stránky jsou navrženy ve formě mapy České republiky, na které je každá budova označená svou barvou. Objekty na mapě lze rozkliknout pro zobrazení doplňkového menu se seznamem smluv, které se týkají tohoto objektu. Každá smlouva v seznamu má odkaz na detail smlouvy. V detailu je zobrazen text dokumentu, odkaz na zdroj stáhnuté informace, tlačítko pro zobrazení originálu dokumentu. Zobrazení originálu dokumentu je realizováno ve formě vestavěného PDF prohlížeče.

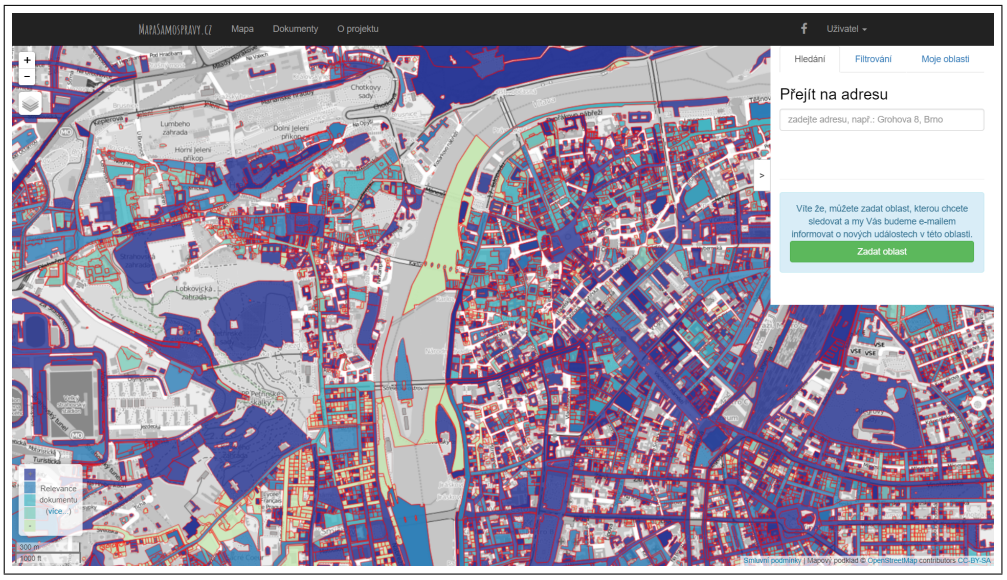
Portál má záložku „Dokumenty“, kde se zobrazí úplný seznam všech dokumentů po 30 dokumentech na stránku. Každý záznam v seznamu obsahuje odkaz na detail vybraného dokumentu. Webový portál poskytuje informace na základě údajů stáhnutých z `www.edesky.cz` a prezentuje úřední desky z více než 1200 úřadů.

Nevýhody portálu:

- zastaralé stylování stránek portálu,

[13] Application Programming Interface

[14] `https://www.highcharts.com/`



Obrázek 2.3: Portál `www.mapasamospravy.cz`

- detail dokumentu neobsahuje žádné užitečné informace kromě odkazu na originál dokumentu,
- dokumenty se tykají jen jedné oblasti, nejsou dokumenty z jiných oblastí a firem,
- webový portál obsahuje implementační chyby, nefunkční přechod na poslední stránku v seznamu dokumentů.

Výhody portálu:

- nestandardní uživatelské rozhraní ve formě mapy,
- aktuální data v dokumentech,
- možnost registrace.

2.1.4 Shrnutí aktuálních řešení

V této kapitole jsem diskutoval jenom nejzajímavější projekty, se kterými jsem se setkal během rešerše tohoto tématu. Počet aplikací, vytvořených pro poskytování údajů o statní správě je obrovský a každá má své výhody a nevýhody. Z analýzy stávajících řešení vyplývá, že většina aplikací se zaměřuje jenom na jednu určitou oblast statní správy a pracuje s jedním typem dokumentů. Probrané aplikace neobsahují takové typy záznamů jako platby, faktury a objednávky. Data o dokumentech, které obsahuje databáze mojí aplikace budou novým a užitečným zdrojem informací pro veřejnost. V úvahu také připadají

chyby v návrhu uživatelských rozhraní hodnocených aplikací, které by šlo odstranit při vývoji nové aplikace.

S ohledem na počet nedostatků stávajících řešení uvažuji o tom, že stojí za to navrhnout a implementovat novou webovou aplikaci pro podporu projektu Otevřená data.

2.2 Popis dat, se kterými aplikace bude pracovat

Hlavní funkčností mojí aplikace je sdílení informací poskytovaných různými státními orgány v České republice. Pro účely bakalářské práce byla použita jíž vytvořená databáze, naplněná daty o statní správě.

Data do databáze se dostanou pomocí externí aplikace. Přístup k databázi jsem získal od svých kolegů z firmy Profinit[15]. Cílem tohoto software je nalezení dat z veřejně dostupných zdrojů, jejich stáhnutí a následně předzpracování dat do podoby, které by vyhovovalo pro další použití. Tento proces se nazývá „crawling“ nebo „data scraping“. Tyto dva pojmy mají mezi sebou odlišnosti.

Pojmem crawling se většinou označuje web crawling, což znamená sbírání informací na internetových stránkách a procházení webových stránek do hloubky za účelem indexování stránek a následné vytvoření databáze těchto indexů [3]. Tato technologie se často používá různými vyhledávacími systémy pro rychlejší odezvu na dotazy uživatelů a zvýšení kvality svých služeb.

Na druhou stranu pojem data scraping se častěji používá, když se jedná o získaní informací z různých zdrojů, zejména z počítači vytvořených výstupů čitelných člověkem. Toto získání dat nemusí byt nutně na internetu, ale z jakýchkoliv zdrojů. [4]

Externí aplikace funguje s daty, které jsou dotažená z různých ministerstev České republiky. Většina ministerstev poskytuje přístup ke svým datům pro veřejnost ve formě nepřehledných a neúplných excel[16] souborů, ve kterých není možné rychle najít potřebnou informaci a zanalyzovat ji. V následujících podkapitolách jsem popsal způsoby zveřejnění a formát dat u různých ministerstev České republiky, se kterými pracuje software pro sbírání dat.

2.2.1 Ministerstvo financí České republiky

Na `www.data.mfcr.cz` Ministerstvo financí České republiky poskytuje JSON API, kde pravidelně aktualizuje informace o nových datových sadách. Rozhraní poskytuje odkazy pro stažení souborů a informaci o jejich aktuálnosti a platnosti dat v nich. Ukázku API je vidět na obrázku 2.4.

MFČR poskytuje čtyři základní typy excel dokumentů. První obsahuje informaci o objednávkách, které jsou evidovány ministerstvem od roku 2007.

[15] Profinit EU, s.r.o.

[16] Microsoft Excel

```
{
    "help": "",
    "success": true,
    "result": {
        "id": "5c216da1-ab1f-4cb9-9580-514d8767c5f6",
        "name": "seznam-objednavek-minsterstva-financi",
        "title": "Seznam objednávek Minsterstva financí",
        "publisher_name": "Ministerstvo financí České republiky",
        "publisher_uri": "http://www.mfcr.cz",
        "maintainer_email": "otevrenadata@mfcr.cz",
        "ruian_code": "1",
        "ruian_type": "ST",
        "license_title": "Jedná se o volné dílo.",
        "license_link": "https://portal.gov.cz/portal/ostatni/volny-pristup-k-ds.html",
        "notes": "<p>Sada obsahuje seznam objednávek Minsterstva financí.</p>",
        "url": "http://data.mfcr.cz/cs/dataset/seznam-objednavek-minsterstva-financi",
        "state": "Active",
        "metadata_created": "2015-03-23T10:53:48+01:00",
        "metadata_modified": "2017-10-04T15:02:11+02:00",
        "type": "dataset",
        "resources": [
            {
                "id": "263cd4e6-bf7f-4250-afdd-dab5251b8af6",
                "url": "http://data.mfcr.cz/sites/default/files/Platne_neplatne_smlouvy_01092017_0.csv",
                "description": "",
                "format": "csv",
                "mimetype": "text/csv",
                "state": "Active",
                "name": "Objednávky 2007 - 2017",
                "size": "4210472",
                "created": "2015-03-23T10:55:55+01:00",
                "last_modified": "2017-10-04T13:12:13+02:00"
            },
```


Obrázek 2.4: Ukazka datového API MFČR

Druhý typ dokumentu obsahuje faktury od roku 2010 do 2014, kde každý rok je rozdělen do vlastního dokumentu. V třetím typu excel souboru jsou uloženy faktury od roku 2015, kde každý rok je také rozdělen do vlastního dokumentu. Formáty faktur pro roky 2010–2014 a 2015–2018 se liší, proto se pro tyto datové sady se používá odlišný formát. Poslední datová sada obsahuje platné i neplatné smlouvy v jednom souboru.

2.2.2 Ministerstvo spravedlnosti České republiky

Oficiálním webem ministerstva spravedlnosti je `https://data.justice.cz`. Tento portál slouží pro přehled všech oficiálních dokumentů, kde ministerstvo hraje jednu z rolí plátce nebo dodavatele služeb. Na portálu lze nalézt seznam všech datových sad, které publikuje ministerstvo, a podmínky jejich použití.

Ministerstvo spravedlnosti České republiky publikuje všechny aktuálně platné smlouvy v jednom excel souboru, který byl naposledy změněn na začátku roku 2017. URL pro stáhnutí datové sady se smlouvami je do aplikace pro sbírání dat vloženo manuálně. Soubor obsahuje základní informace o smlou-

vě jako název smlouvy, název smluvního partnera, IČO[17], stručný popis, platnost a částka v korunách vč. DPH[18].

MSČR také publikuje informace o souvisejících fakturách od roku 2009, kde pro každý rok existuje vlastní excel soubor. Pro zpracování těchto datových sad je do aplikace odkaz na dokument z roku 2009 vložen manuálně a nové odkazy pro následující roky se generují automaticky a liší se jen rokem. Tyto dokumenty obsahují detailnější informaci o fakturách. Struktura dokumentu je podobná dokumentu se smlouvami, ale navíc obsahuje položky variabilní symbol a datum splatnosti faktury.

2.2.3 Ministerstvo životního prostředí České republiky

Na svém oficiálním webu `https://www.mzp.cz/cz/otevrena_data` Ministerstvo životního prostředí České republiky publikuje informace o fakturách, objednávkách i smlouvách. Data se nacházejí v tabulkách na stránkách webu s odkazy pro vygenerování souhrnných CSV, JSON nebo XLS dokumentů. Software pro sbíraní dat dotazuje přímo tyto odkazy pro generování dat pro všechny typy dokumentů. Odpovídající odkazy na datové instance jsou do aplikace vloženy ručně.

Všechny záznamy ministerstva životního prostředí mají podobnou strukturu. Důležitými položkami jsou:

- předmět smlouvy/faktury/objednávky,
- dodavatel smlouvy/faktury/objednávky,
- datum platby/objednávky/splatnosti,
- celková částka.

2.2.4 Ministerstvo kultury České republiky

Ačkoliv Ministerstvo kultury České republiky má svůj vlastní web pro otevřená data `https://data.mkcr.cz`, stejně udržuje nejméně kvalitní datové sady. API pro poskytování informace o nových datových instancích není, proto informaci o tom, zda se objevily nové dokumenty pro zpracování je potřeba kontrolovat manuálně a pak vložit do aplikace pro zpracování. MKČR poskytuje dokument se smlouvami, výše kterých přesahuje 50 000 korun a dokumenty s fakturami. Dokumenty s fakturami se dělí do různých excel souborů podle roku a typu:

- zálohové faktury,
- došlé faktury,

[17] Identifikační číslo osoby
[18] Daň z přidané hodnoty

- platební poukazy,
- dobropisy.

2.2.5 Ministerstvo pro místní rozvoj

Ministerstvo pro místní rozvoj publikuje informaci o souvisejících dokumentech na svém oficiálním webu `http://data.mmr.cz`. Tento portál využívá software CKAN, o kterém jsem psal v kapitole 2.1.1. Portál obsahuje 68 datových sad, ze kterých většina jsou přehledy faktur, smluv a objednávek od roku 2015.

Ministerstvo pro místní rozvoj kromě svého portálu neposkytuje žádné API, přes které by šlo stáhnout požadováné soubory s daty. Proto se musí odkazy na soubory pro zpracování přidávat ručně pro každý běh aplikace pro sběr dat.

Struktura dat poskytovaných ministerstvem pro místní rozvoj, je podobná datům, která poskytují ministerstva popsáná výše. Při zkoumání datových sad jsem pozoroval rozbité kódování u českých popisů a názvů smluv, proto některé soubory nejde zatím pořádně zpracovat.

2.2.6 Ministerstvo dopravy a Státní fond dopravní infrastruktury

Ministerstvo dopravy a Státní fond dopravní infrastruktury poskytují datové sady s fakturami a smlouvami na svém oficiálním webu `https://www.mdcr.cz/Ministerstvo/Otevrena-data`. Tato dvě ministerstva poskytují různá data, ale ve stejném formátu, což ulehčuje jejich zpracování pro aplikaci.

Smlouvy a faktury od roku 2015 se publikuji v excel souborech, které lze stáhnout z webu ministerstva dopravy. Soubory se publikují s měsíční periodicitou a vždy k 20. dni kalendářního měsíce. Odkazy na soubory pro zpracování se do aplikace vkládají manuálně.

2.2.7 Shnutí popisu dat

Z popisu dat, se kterými pracuje aplikace je vidět, že otevřená data, která poskytují ministerstva nejsou vždy v dostupném a pro zpracování validním formátu. Vyhledávání nových datových sad pro zpracování a nalezení dostupných URL pro jejich stáhnutí je stále hlavním problémem. Jedním z kvalitních poskytovatelů otevřených dat je Ministerstvo financí České republiky. Tato organizace udržuje a aktualizuje JSON rozhraní, ze kterého jde zjistit všechny potřebné údaje o nových datových sadách.

Rozšíření počtů datových zdrojů pro získání otevřených dat není hlavním cílem bakalářské práce, proto se nebudeme dále zabývat.

2.3 Požadavky

Požadavky na nový webový portál vychází z analýzy stávajících řešení v předchozí kapitole a nápadů, které jsem ve spolupráci s vedoucím navrhli pro zlepšení přístupu běžného uživatele k otevřeným datům a jejich používání. Při návrhu požadavků jsem bral v úvahu data, se kterými pracuje externí aplikace pro sbíraní dat, aby všechny požadavky bylo možné splnit.

Podle Laplante inženýrství požadavků je „*podobor systémového inženýrství a softwarového inženýrství, který se zabývá stanovením cílů, funkcí a omezení hardwarových a softwarových systémů*“ [5, str. 44]. Specifikace požadavků zahrnuje úkoly, které vyvíjený produkt musí splnit pro dosažení cílů a splnění podmínek určených zákazníkem. Požadavky lze rozdělit do několika základních kategorií [6, str. 35-36]:

- požadavky od zákazníka,
- funkční požadavky,
- výkonnostní požadavky,
- designové požadavky,
- odvozené požadavky,
- alokované požadavky.

V této práci se zaměřuji na rozdělení požadavků na funkční a nefunkční. Funkční požadavky určují povinné úkoly, akce a činnosti, které musí být splněny vyvinutým systémem [6, str. 36]. Nefunkční požadavky se na druhou stranu zaměřují na specifikaci kriterií, podle kterých se dá měřit kvalita aplikace a kterých si zákazník cení nejvíce. „*The nonfunctional requirements are sometimes referred to as ‚nonbehavioral requirements‘ or ‚software quality attributes‘*“ [7, str. 113].

2.3.1 Funkční požadavky

F1 Seznam veřejně dostupných dokumentů (smlouvy, faktury, platby, objednávky)

Uživatel webového portálu bude moci získat rychlý přehled všech dokumentů, které jsou evidovány v aplikaci. Z přehledu dokumentů bude mít uživatel možnost přejít na detail vybraného záznamu. Přehled musí obsahovat název, typ, částku a datum zpracování dokumentu. Uživatel bude moci filtrovat záznamy podle jejich typu (smlouvy, faktury, platby, objednávky).

F2 Zobrazení detailu vybraného dokumentu

Uživatel bude moci zobrazit detail vybraného dokumentu. Detail umožní zobrazit následující informace o dokumentu:

- název,
- typ,
- částka,
- měna,
- variabilní symbol,
- periodicita platby,
- role publikující strany (nepovinný),
- popis (nepovinný),
- jednoznačný identifikátor dokumentu,
- datum zpracování,
- datum platby (nepovinný),
- datum vytvoření dokumentu,
- datum splatnosti (nepovinný),
- název publikující organizace,
- název související oragnizace.

Informace nacházející se v detailu může lišit v závislosti na typu dokumentu. V detailu dokumentu bude uživatel moci zobrazit krátký přehled o plátci a dodavateli, souvisejících dokumentech a přejít na detail těchto organizací.

F3 Evidence organizací

Portál umožní uživateli zobrazit přehled organizací evidovaných v systému. Na přehledu bude zobrazena základní informace o organizaci jako název, typ a IČO. Z přehledu bude uživatel moci přejít na detail vybrané firmy, ministerstva nebo podnikatele. Záznamy na přehledu bude možné odfiltrovat podle typu organizace.

F4 Detail organizace

Uživatel webového portálu bude mít možnost zobrazení detailu organizace. Detail vybrané firmy, ministerstva, nebo podnikatele bude obsahovat následující informace:

- název,
- IČO,
- DIČ[19],
- počet publikovaných dokumentů,
- počet souvisejících dokumentů,
- typ,
- veřejná (ANO/NE).

Zároveň uživateli bude zobrazen krátký přehled souvisejících dokumentů s touto organizací.

F5 Poskytnutí API

Portál bude umožňovat prohlédnout a stáhnout data o dokumentech a organizacích evidovaných v databázi webového portálu přes JSON API. Uživatel bude moci dotazovat na seznam dokumentů, detail dokumentu podle unikátního identifikátoru, seznam organizací, detail organizace podle unikátního identifikátoru, detail organizace podle IČO. Aby uživatel věděl jak pracovat s API, portál bude poskytovat návod k použití API endpointů a popis poskytovaných dat.

2.3.2 Nefunkční požadavky

NF1 Grafické uživatelské prostředí

Základním cílem aplikace je zjednodušení přístupu běžného uživatele k otevřeným datům, proto vytvoření uživatelsky přívětivého grafického prostředí je jedním ze základních nefunkčních požadavků.

NF2 Přístup z mobilních zařízení

Všechny funkčnosti poskytovány webovým portálem budou dostupné nejenom z počítače, ale i z mobilních zařízení. Webový portál bude poskytovat stejnou kvalitu funkcionalit pro mobilní zařízení.

[19] Daňové identifikační číslo

NF3 Použité technologie

Pro implementaci aplikace budou použity moderní technologie. Mezi moderní technologie patří programovací jazyk Java 8, framework Spring nebo Spring Boot pro serverovou stranu aplikace, HTML/XHTML[20] verze 5 pro vytvoření obrazovek, CSS[21] framework Bootstrap pro stylování obrazovek, JavaScript framework JQuery, Apache Tomcat nebo Glassfish aplikační server, PostgreSQL nebo Oracle Database databáze pro ukládání dat.

NF4 Rozšiřitelnost

Jak jsem již zmiňoval, Otevřená data je projekt, který se postupem času rozvijí víc a víc a počet nových zdrojů dat se zvětšuje. Počet dat a funkčností poskytovaných webovým portálem se bude v budoucnu zvětšovat. Proto je třeba při návrhu architektury aplikace počítat se s možností rozšíření a přidaní nových funkčností.

NF5 Výkonnost

Portál dokáže zpracovat dotazy od 200 uživatelů zároveň.

[20] Extensible hypertext markup language
[21] Cascading Style Sheets

Kapitola 3

Návrh

3.1 Případy užití

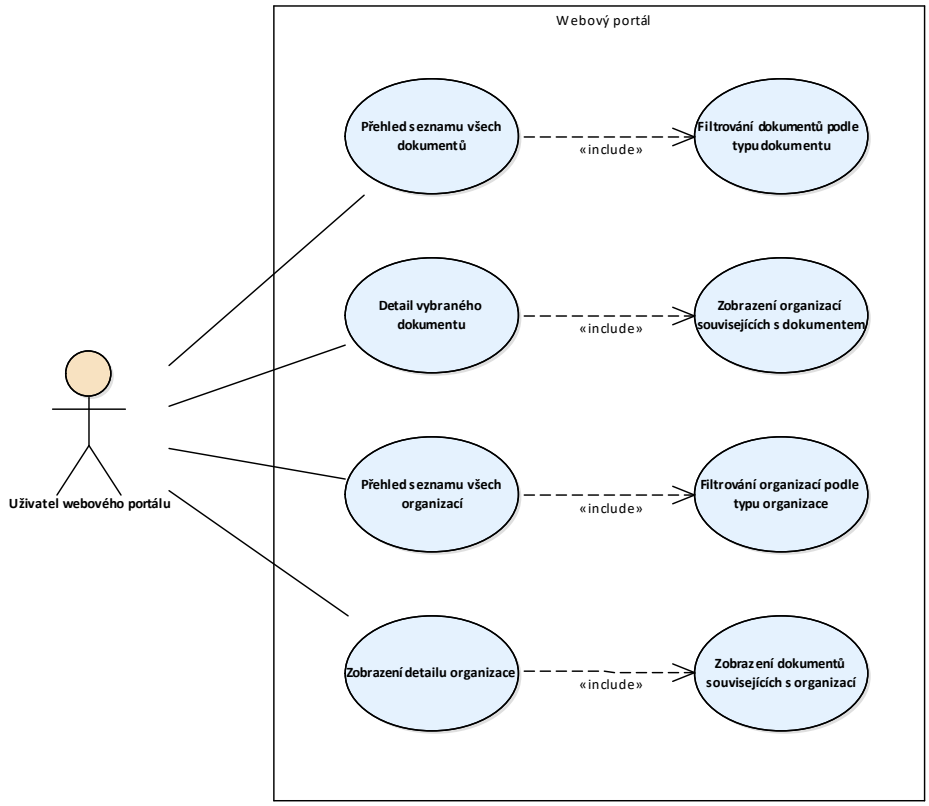
V softwarovém inženýrství je případ užití seznam akcí nebo kroků, které definují interakci mezi uživatelem aplikace a aplikací samotnou. Alistair Cockburn definuje případ užití tímto způsobem: „*případ užití zachycuje smlouvu mezi účastníky systému o jeho chování. Případ užití popisuje chování systému za různých podmínek, neboť reaguje na žádost jednoho ze zúčastněných stran, nazývaného primárním hráčem. Primární aktér iniciuje interakci se systémem pro dosažení určitého cíle. Systém reaguje a chrání zájmy všech zúčastněných stran*“ [8, str. 15].

Pro napsaní případů užití existuje spoustu různých možností a každý softwarový inženýr sám stanoví ten nejvhodnější způsob pro svůj projekt. Časem bylo stanoveno několik vzorů pro napsaní vysoce kvalitních případů užití:

- Cockburn style - vzor definovány Alistair Cockburnem v jeho populární knize „Writing Effective Use Case“ [8]:
  - „fully dressed“ - rozšíření a detailní popis struktury doporučený Alistair Cockburnem. Obsahuje název, cíl, rozsah, úroveň, zúčastněné strany a zájmy, minimální záruky, úspěšné záruky, spouštěč, hlavní scénář, rozšíření, technologie a seznam dat, související informace.
  - „casual“ - zjednodušený způsob, který zachycuje informaci o cíli, primárním hráči, rozsahu, úrovni a popisu případů užití.
- „Fowler style“ - vzor definovány Martinem Fowlerem, který zachycuje cíl, hlavní úspěšný scénář, popis jiných možných výsledků případů užití. [9, str. 100-101]

Ve své bakalářské práci pro znázornění případů užití používám UML[22] diagramy s detailním popisem každého z nich. Při návrhu případů užití jsem vycházel ze základních funkčních požadavků kladených na webový portál a API. V rámci bakalářské práce se omezujeme jenom na běžného nepřihlášeného uživatele portálu nebo vystaveného API.

3.2 Případy užití portál



Obrázek 3.1: Případy užití portál

3.2.1 UC 1. Přehled seznamu všech dokumentů

Rozsah:	Webový portál
Úroveň:	Záměr uživatele

[22]Unified Modeling Language

Hlavní účastník:	Uživatel portálu
Předpoklady:	Uživatel se nachází na domovské stránce portálu

Scénář úspěšného plnění:

1. Uživatel portálu si chce zobrazit seznam všech dokumentů.
2. Uživatel stiskne tlačítko „Dokumenty“.
3. Uživateli se načte nová stránka „Dokumenty“ s přehledem dokumentů.
4. Uživateli se zobrazí 10 dokumentů a možnost přechodu na další stránku s přehledem.
5. Uživatel stiskne tlačítko pro přechod na další stránku seznamu.
6. Opakuje se krok číslo 4.

Výjimky ze standardního průběhu:

4.a Uživateli se nenačte žádný dokument.

1. Aplikace zobrazí hlášku o tom, že není žádný dokument k zobrazení.
2. Uživateli se nezobrazí možnost přechodu na další stránky s přehledem dokumentů.

3.2.1.1 UC 1.1. Filtrování dokumentů podle typu dokumentu

Rozsah:	Webový portál
Úroveň:	Záměr uživatele
Hlavní účastník:	Uživatel portálu
Předpoklady:	Uživatel se nachazí na stránce „Dokumenty“ portálu

Scénář úspěšného plnění:

1. Uživatel portálu si chce nastavit filtrování dokumentů podle jejich typu.
2. Uživatel si zvolí jeden nebo více checkboxů na filtrování dokumentů.
3. Uživateli se zobrazí 10 dokumentů vybraného typu a možnost přechodu na další stránku s přehledem.
4. Uživatel postupně mění typy filtrování dokumentů.
5. Opakuje se krok číslo 3.

Výjimky ze standardního průběhu:

3.a Uživateli se nenačte žádný dokument.

1. Aplikace zobrazí hlášku o tom, že není žádný dokument k zobrazení.
2. Uživateli se nezobrazí možnost přechodu na další stránky s přehledem dokumentů.

3.2.2 UC 2. Detail vybraného dokumentu

Rozsah:	Webový portál
Úroveň:	Záměr uživatele
Hlavní účastník:	Uživatel portálu
Předpoklady:	Uživatel se nachází na stránce „Dokumenty“ portálu a přehled obsahuje aspoň jeden dokument

Scénář úspěšného plnění:

1. Uživatel portálu si chce zobrazit detail dokumentu.
2. Uživatel si zvolí jeden z dokumentů z přehledu a klikne na aktivní odkaz v názvu dokumentu.
3. Uživateli se zobrazí detail vybraného dokumentu.
4. Detail dokumentu obsahuje všechny informace o dokumentu popsáné v požadavku č.2.
5. Detail dokumentu obsahuje kratký přehled o souvisejících organizacích.

3.2.3 UC 3. Přehled seznamu všech organizací

Rozsah:	Webový portál
Úroveň:	Záměr uživatele
Hlavní účastník:	Uživatel portálu
Předpoklady:	Uživatel se nachází na stránce „Domů“ portálu

Scénář úspěšného plnění:

1. Uživatel portálu si chce zobrazit seznam všech organizací.
2. Uživatel stiskne tlačitko „Organizace“.
3. Uživateli se načte stránka „Organizace“ s přehledem organizací.
4. Uživateli se zobrazí 10 organizací a možnost přechodu na další stránku s přehledem.
5. Uživatel stiskne tlačítko pro přechod na další stránku seznamu.
6. Opakuje se krok číslo 4.

Výjimky ze standardního průběhu:

4.a Uživateli se nenačte žádná organizace.

1. Aplikace zobrazí hlášku o tom, že není žádná organizace k zobrazení.
2. Uživateli se nezobrazí možnost přechodu na další stránky s přehledem organizací.

4.b Uživateli se načte meně než 10 organizací.

1. Uživateli se nezobrazí možnost přechodu na další stránku.

3.2.3.1 UC 3.1. Filtrování organizací podle typu

Rozsah:	Webový portál
Úroveň:	Záměr uživatele
Hlavní účastník:	Uživatel portálu
Předpoklady:	Uživatel se nachází na stránce „Organizace“ portálu

Scénář úspěšného plnění:

1. Uživatel portálu si chce nastavit filtrování organizací podle jejich typu.
2. Uživatel si zvolí jeden nebo více checkboxů (zaškrtávací políčko), typ organizace, na filtrování organizací.
3. Uživateli se zobrazí 10 záznamů vybraného typu a možnost přechodu na další stránku s přehledem.
4. Uživatel postupně mění typy filtrování organizací.
5. Opakuje se krok číslo 3.

Výjimky ze standardního průběhu:

3.a Uživateli se nenačte žádná organizace.

1. Aplikace zobrazí hlášku o tom, že není žádná organizace k zobrazení.
2. Uživateli se nezobrazí možnost přechodu na další stránky s přehledem organizací.

4.a Uživatel stiskne tlačítko pro přechod na další stránku.

1. Uživateli se zobrazí dalších 10 organizací vybraného typu.

3.2.4 UC 4. Zobrazení detailu organizace

Rozsah:	Webový portál
Úroveň:	Záměr uživatele
Hlavní účastník:	Uživatel portálu
Předpoklady:	Uživatel se nachazí na stránce „Organizace“ portálu a přehled obsahuje aspoň jednu organizaci

Scénář úspěšného plnění:

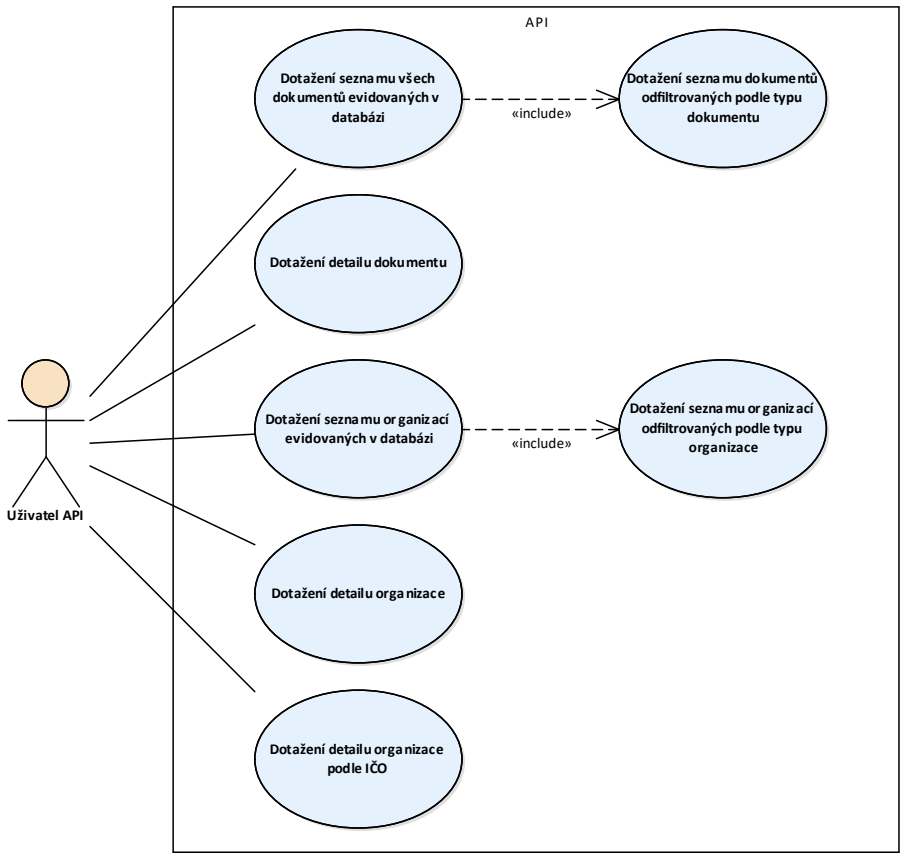
1. Uživatel portálu si chce zobrazit detail organizace.
2. Uživatel si zvolí jednu z organizací z přehledu a klikne na aktivní odkaz v názvu organizace.
3. Uživateli se zobrazí detail vybrané organizace.
4. Detail organizace obsahuje všechny informace o organizaci popsáné v požadavku č.4.
5. Detail organizace obsahuje krátký přehled o souvisejících dokumentech.

Výjimky ze standardního průběhu:

5.a Uživateli se nezobrazí související dokumenty.

1. Aplikace zobrazí hlášku o tom, že k této organizaci není žádný související dokument.

3.3 Případy užití API



Obrázek 3.2: Případy užití API

3.3.1 UC 5. Dotažení seznamu všech dokumentů evidovaných v databázi

Rozsah:	API portálu
Úroveň:	Záměr uživatele
Hlavní účastník:	Uživatel API portálu

Předpoklady:	Uživatel získal přístup k endpointu pro dotažení seznamu všech dokumentů evidovaných v databázi a přečetl si návod na použití

Scénář úspěšného plnění:

1. Uživatel API chce udělat dotaz na seznam všech dokumentů.
2. Uživatel si nastaví parametry „rowStart“ a „rowEnd“ GET požadavku podle návodu.
3. Uživatel odešle GET požadavek na server.
4. Uživatel dostane odpověď ze serveru.
5. Odpověď ze serveru obsahuje seznam požadovaných dokumentů podle podmínek nastavených v požadavku.

Výjimky ze standardního průběhu:

5.a Odpověď ze serveru neobsahuje žádný dokument.

1. Aplikace nevrátí žádnou hlášku, ale vrátí prázdný seznam dokumentů.

3.3.1.1 UC 5.1. Dotažení seznamu dokumentů odfiltrovaných podle typu dokument

Rozsah:	API portálu
Úroveň:	Záměr uživatele
Hlavní účastník:	Uživatel API portálu
Předpoklady:	Uživatel získal přístup k endpointu pro dotažení seznamu všech dokumentů evidovaných v databázi a přečetl si návod na použití

Scénář úspěšného plnění:

1. Uživatel API chce udělat dotaz na seznam všech dokumentů odfiltrovaných podle typu.
2. Uživatel si nastaví parametr „recordTypes“ GET požadavku podle návodu.
3. Uživatel odešle GET požadavek na server.
4. Uživatel dostane odpověď ze serveru.
5. Odpověď ze serveru obsahuje seznam s dokumenty vybraného typu.
6. Uživatel opakuje krok 3 až 5 s různými kombinacemi typů dokumentu.

Výjimky ze standardního průběhu:

4.a Dotaz na endpoint selže.

1. Uživatel dostane chybovou/validační hlášku s popisem problému.
2. Uživatel může opakovat akci s jinými parametry.

5.a Odpověď ze serveru neobsahuje žádný dokument.

1. Aplikace nevrátí žádnou hlášku, ale vrátí prázdný seznam dokumentů.

3.3.2 UC 6. Dotažení detailu dokumentu

Rozsah:	API portálu
Úroveň:	Záměr uživatele
Hlavní účastník:	Uživatel API portálu
Předpoklady:	Uživatel získal přístup k endpointu pro dotažení detailu dokumentu a přečetl si návod na použití

Scénář úspěšného plnění:

1. Uživatel API chce udělat dotaz na detail dokumentu.
2. Uživatel si nastaví parametr „id“ GET požadavku podle toho, detail jakého dokumentu chce dotáhnout.
3. Uživatel odešle GET požadavek na server.
4. Uživatel dostane odpověď ze serveru.
5. Odpověď ze serveru obsahuje detail dokumentu, na který se ptal uživatel.

Výjimky ze standardního průběhu:

5.a Databáze aplikace neobsahuje požadovány dokument.

1. Aplikace nevrátí žádnou hlášku, ale vrátí prázdnou odpověď.

3.3.3 UC 7. Dotažení seznamu organizací evidovaných v databázi aplikace

Rozsah:	API portálu
Úroveň:	Záměr uživatele
Hlavní účastník:	Uživatel API portálu
Předpoklady:	Uživatel získal přístup k endpointu pro dotažení seznamu organizací evidovaných v databázi aplikace a přečetl si návod na použití

Scénář úspěšného plnění:

1. Uživatel API chce udělat dotaz na dotažení seznamu organizací evidovaných v databázi aplikace.
2. Uživatel si nastaví parametry „rowStart“ a „rowEnd“ GET požadavku podle návodu.
3. Uživatel odešle GET požadavek na server.
4. Uživatel dostane odpověď ze serveru.
5. Odpověď ze serveru obsahuje seznam požadovaných organizací podle podmínek nastavených v požadavku.

Výjimky ze standardního průběhu:

5.a Odpověď ze serveru neobsahuje žádnou organizaci.

1. Aplikace nevrátí žádnou hlášku, ale vrátí prázdný seznam organizací.

3.3.4 UC 7.1. Dotažení seznamu organizací odfiltrovaných podle typu

Rozsah:	API portálu
Úroveň:	Záměr uživatele
Hlavní účastník:	Uživatel API portálu
Předpoklady:	Uživatel získal přístup k endpointu pro dotažení seznamu organizací evidovaných v databázi aplikace a přečetl si návod na použití

Scénář úspěšného plnění:

1. Uživatel API chce udělat dotaz na dotažení seznamu organizací odfiltrovaných podle typu.
2. Uživatel si nastaví parametr „organizationTypes“ GET požadavku podle návodu.
3. Uživatel odešle GET požadavek na server.
4. Uživatel dostane odpověď ze serveru.
5. Odpověď ze serveru obsahuje seznam organizací vybraného typu.
6. Uživatel opakuje krok 3 až 5 s různými kombinacemi typů organizace.

Výjimky ze standardního průběhu:

4.a Dotaz na endpoint selže.

1. Uživatel dostane chybovou/validační hlášku s popisem problému.
2. Uživatel může opakovat akci s jinými parametry.

5.a Odpověď ze serveru neobsahuje žádnou organizaci.

1. Aplikace nevrátí žádnou hlášku, ale vrátí prázdný seznam organizací.

3.3.5 UC 8. Dotažení detailu organizace

Rozsah:	API portálu
Úroveň:	Záměr uživatele
Hlavní účastník:	Uživatel API portálu
Předpoklady:	Uživatel získal přístup k endpointu pro dotažení detailu organizace a přečetl si návod na použití

Scénář úspěšného plnění:

1. Uživatel API chce udělat dotaz na detail organizace.
2. Uživatel si nastaví parametr „id“ GET požadavku podle toho, detail jaké organizace chce dotáhnout.
3. Uživatel odešle GET požadavek na server.
4. Uživatel dostane odpověď ze serveru.
5. Odpověď ze serveru obsahuje detail organizace, na kterou se ptal uživatel.

Výjimky ze standardního průběhu:

5.a Databáze aplikace neobsahuje požadovanou organizaci.

1. Aplikace nevrátí žádnou hlášku, ale vrátí prázdnou odpověď.

3.4 Wireframe aplikace

Následující podkapitola se zabývá návrhem obrazovek webové aplikace. Wireframes aplikace byly vytvářeny v programu Enterprise Architect[23] jako Webpage Wireframe Diagram. Při návrhu obrazovek byl kladen velký důraz na minimalistický, jednoduchý design, který by splnil všechny funkční a nefunkční požadavky kladené na webovou aplikaci.

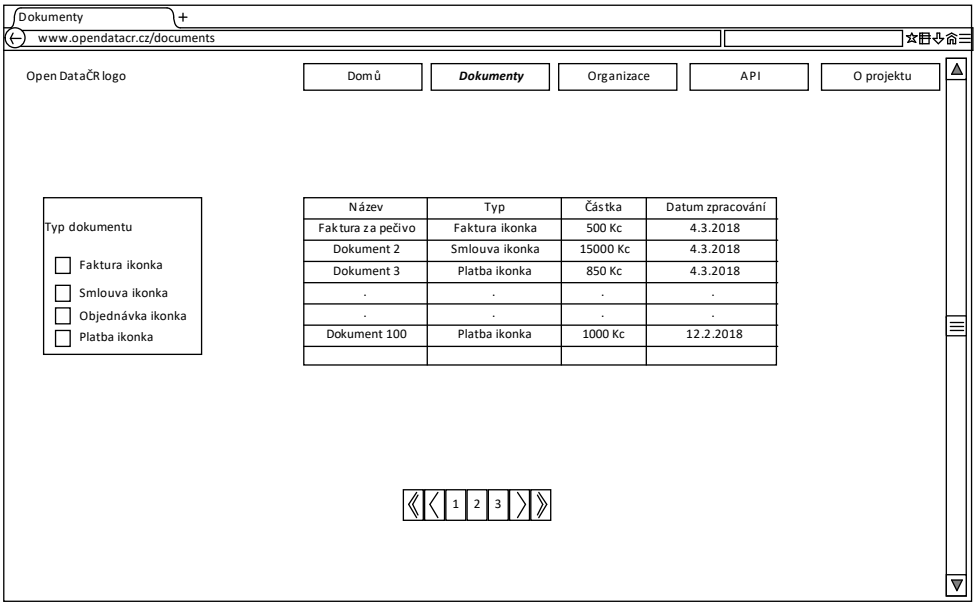
Grafický design aplikace se bude skládat ze sedmi hlavních obrazovek:

- Dokumenty,
- Detail dokumentu,
- Organizace,
- Detail organizace,
- API,
- Domů,
- O projektu.

3.4.1 Obrazovka - Dokumenty

Obrazovka **Dokumenty** umožňuje uživateli portálu zobrazit seznam všech dokumentů, které aplikace eviduje v databázi a zároveň mít rychlý přehled

[23] Sparx Systems Enterprise Architect



Obrázek 3.3: Wireframe „Dokumenty“

o vybraném záznamu. Jak je vidět na obrázku 3.3, základem stránky je tabulka, která obsahuje název, typ, částku a datum zpracování dokumentu.

Aby byly splněny funkční požadavky kladené na aplikaci, na tuto stránku byla přidána možnost filtrování záznamu podle jejich typu. Toto bylo zajištěno pomocí checkboxů vlevo od tabulky, které umožňují uživateli zaškrtnout požadované typy dokumentů.

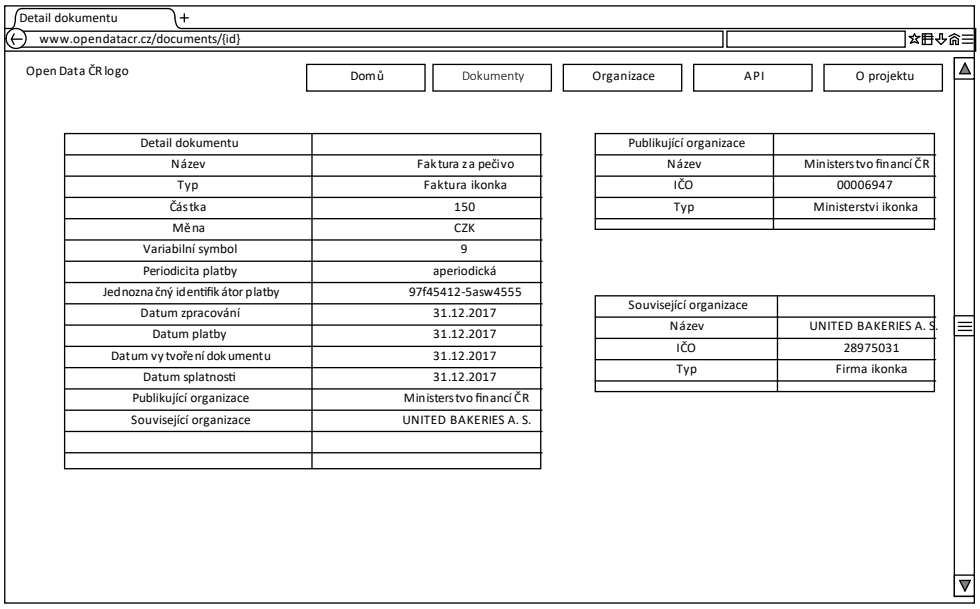
Každý název záznamu v tabulce je aktivním odkazem, který umožňuje uživateli přejít na detail dokumentu.

3.4.2 Obrazovka - Detail dokumentu

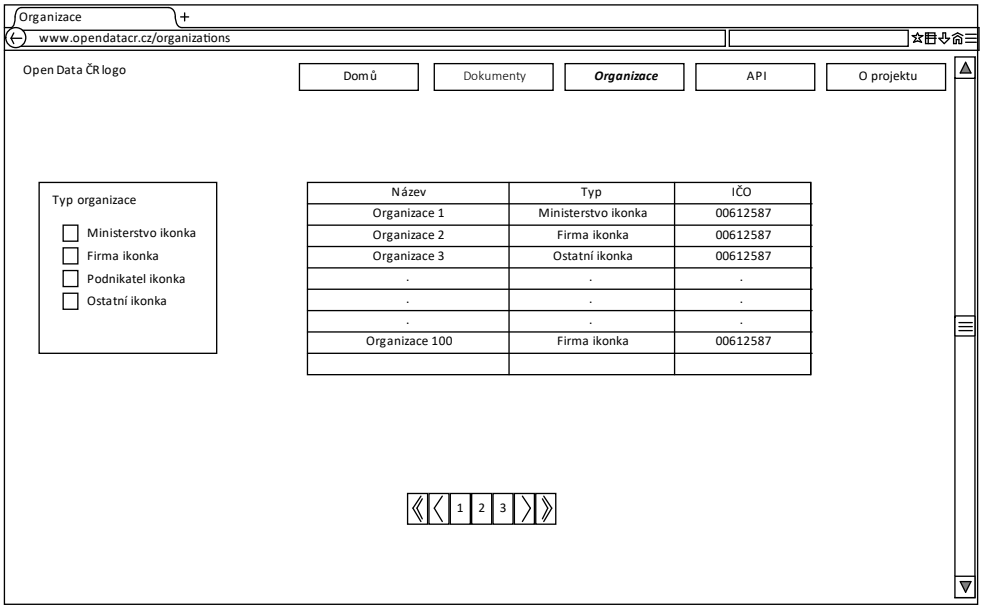
Obrazovka **Detail dokumentu** obsahuje podrobnější informaci o vybraném dokumentu. Informace o vybraném záznamu se bude nacházet v tabulce, která se skládá ze dvou sloupců vlastnost–hodnota. Tabulka také obsahuje informace o organizacích, kterých se tento dokument tyká. Vpravo od tabulky s informací o detailu dokumentu se nachází dvě tabulky s krátkým přehledem o organizacích, které souvisejí s tímto dokumentem. Název organizace je aktivním odkazem, který umožní uživateli přejít na detail organizace.

3.4.3 Obrazovka - Organizace

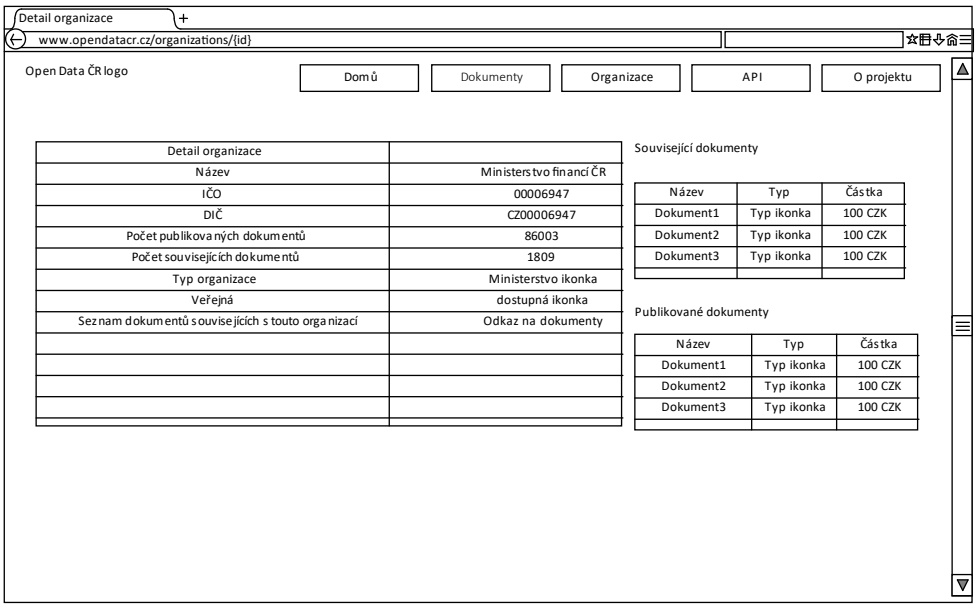
Obrazovka **Organizace** umožňuje uživateli procházet seznam všech organizací evidovaných v databázi aplikace. Uživatel bude mít rychlý přehled



Obrázek 3.4: Wireframe „Detail dokumentu“



Obrázek 3.5: Wireframe „Organizace“



Obrázek 3.6: Wireframe „Detail organizace“

o názvu, typu a IČO organizace s možností přechodu na detail vybraného záznamu pomocí odkazu propojeného s názvem organizace.

Kvůli počtu organizací evidovaných v databázi jsem rozhodl, že na každé stránce bude zobrazeno maximálně 10 záznamů. Dolní část obrazovky obsahuje stránkování, pomocí kterého uživatel bude moci přejít na následující, předchozí, první a poslední stránku s záznamy o organizacích. Toto zajistí maximální přehlednost obrazovky a množství informace na jedné stránce nezatíží uživatele.

3.4.4 Obrazovka - Detail organizace

Obrazovka **Detail organizace** poskytuje uživateli podrobnější informaci o vybrané organizaci. Struktura této stránky je velmi podobná se stránkou **Detail dokumentu**, levá půlka stránky obsahuje detailní informaci o vybrané organizaci a pravá půlka zobrazuje informaci o souvisejících dokumentech.

3.5 Technologie pro implementaci

3.5.1 Vývojové prostředí

Pro vývoj webové aplikace bylo použito vývojové prostředí IntelliJ IDEA[24]. Toto IDE[25] bylo vybráno z důvodu velké nabídky vývojářských nástrojů a pluginů, které lze nainstalovat do prostředí a lehce s nimi pracovat.

Podstatnou částí práce na projektu tvoří práce s daty a s databází. Jednou z klíčových funkčností, které nabízí IntelliJ IDEA, je připojení prostředí do schématu databáze, které používá programátor. Toto přináší spoustu výhod: programátor nemusí přepínat mezi aplikací pro správu databáze a vývojovým prostředím, IntelliJ IDEA nabízí pohodlné GUI[26] pro práci se záznamy v tabulkách, umožňuje práci v databázové konzoli.

IntelliJ IDEA je integrována s takovými nástroji jako Maven, Ant, Gradle, GIT, SVN, Spring Framework, JUnit Test. Část z těchto nástrojů jsem použil při vývoji webové aplikace pro moji bakalářskou práci.

3.5.2 Jazyk programování

Daný projekt je implementován v jazyce Java verze 8. „*The Java programming language is a general-purpose, concurrent, classbased, object-oriented language*“ [10, str. 1]. Tento jazyk patří do skupiny objektově orientovaných programovacích jazyků. Toto znamená, že při řešení úloh se programátor snaží modelovat objekty reálného světa v počítači, pokud možno jedna ku jedné. Jednou z výhod Javy je přenositelnost, což umožňuje používat programy napsáné v tomto jazyce na různých systémech. Přenositelnost mezi různými platformami je dosažena pomocí existence JVM[27]. Java Virtual Machine je sada počítačových programů a datových struktur, která využívá modul virtuálního stroje ke spuštění dalších počítačových programů a skriptů vytvořených v jazyce Java.

Java je jeden z nejpopulárnějších jazyků pro napsaní webových aplikací. Toto je dosaženo díky existenci spousty nástrojů a volně dostupných frameworků pro řešení úloh daného typu.

Konkretně pro moji aplikaci jsem použil technologii Java Enterprise Edition. JEE je součást platformy Java určená pro vývoj a provoz podnikových aplikací a informačních systémů. JEE má podporu pro vývoj webových aplikací, které budou běžet na webovém serveru, což je můj případ.

[24] www.jetbrains.com/idea

[25] Integrated Development Environment (vývojové prostředí)

[26] Graphical user interface

[27] „*The Java Virtual Machine is an abstract computing machine. Like a real computing machine, it has an instruction set and manipulates various memory areas at run time. It is reasonably common to implement a programming language using a virtual machine*“ [11, str. 2]

3.5.3 Návrhové vzory

Jedním ze základních principů vývoje webových aplikací je jejich udržitelnost a rozvoj do budoucna. Jelikož mým cílem je nejenom vývoj aplikace a i její další rozšíření o moduly, návrh aplikace musí být lehce pochopitelný a dostatečně sofistikovaný. V současné době pro tyto účely při vývoji aplikace se aplikují různé návrhové vzory. Návrhové vzory mohou být jak na úrovni implementace tak i na úrovni návrhu architektury. Ve své práci používám oba způsoby. Následující podkapitola obsahuje popis hlavního návrhového vzoru, který byl použit při návrhu architektury aplikace.

3.5.3.1 Model–View–Controller

Model–**View**–**Controller** (dale jen MVC) je softwarová architektura, která rozděluje datový model aplikace, uživatelské rozhraní a řídicí logiku do tří nezávislých komponent tak, že modifikace některé z nich má jen minimální vliv na ostatní. Z definice pojmů vyplývá několik základních výhod používání této architektury:

- rychlý vývoj několika modulu zároveň,
- možnost vývoje několika verzí prezentačních vrstev pro tu samou datovou a bussines vrstvu,
- snadnější testování jednotlivých modulů,
- zabránění existence tzv. „špagetového kódu“.

Model - je komponenta aplikace, která je zodpovědná za data v aplikaci a za aplikační logiku, která je spojená s těmito daty.

View - je zodpovědná za zobrazení a reprezentaci dat z modelu.

Controller - komponenta aplikace, která reaguje na události, typicky akce od uživatele a propaguje změny do modelu a view. Dá se to představit jako vrstva mezi **Model** a **View**.

3.5.3.2 Dependency Injection

Podle Martina Fowlera dependency injection (vkládání závislostí) lze považovat za implementaci nebo konkretní techniku IoC[28]. Hlavní výhodou vkládání závislostí je to, že třída nemusí vytvářet instancí dalších tříd, které potřebuje, tyto jsou jí dodány jiným způsobem z vnějšku, například IoC kontejnerem.[12][13]

[28] Inversion of Controll

3.5.4 Backend frameworky

Pod pojmem backend si lze představit modul aplikace, který je zodpovědný za zpracování vstupních požadavků od klienta, komunikaci s databází aplikace, řešení business logiky a vracení výsledků zpátky do klienta. Pro snadnější a rychlejší vývoj aplikací se používají různé frameworky. Hlavním cílem frameworků je převzetí typických problémů dané oblasti, čímž se usnadní vývoj tak, aby se návrháři a vývojáři mohli soustředit pouze na své zadání. Ve své práci jsem se hlavně soustředil na používání frameworku pro ovládání HTTP[29] requestu a response na webovém serveru, pro práci s databází a pro generování stránek pro front-end část. Následující podkapitola obsahuje popis frameworků, které jsem použil během vývoje backend částí.

3.5.4.1 Spring Boot

Pro implementaci webových modulů aplikace a řešení business logiky aplikace jsem použil framework Spring Boot. Spring Boot je nadstavba nad Spring Framework. Spring Framework je open-source aplikační rámec, který usnadňuje a urychluje vývoj JEE aplikací. Tento framework nabízí několik hotových modulů, které se nejčastěji využívají při vývoji aplikace:

- core container,
- data access/integration,
- web,
- AOP (Aspect Oriented Programming),
- instrumentation,
- test.

Hlavní výhodou Spring Bootu oproti Spring je snadná konfigurace, což umožňuje rychlý vývoj a start nové aplikace. Hlavní myšlenkou je to, že defaultní konfigurace umožňuje napsat plně funkční aplikaci, která bude využívat defaultní nastavení a zdroje. Na druhou stranu dle potřeby programátora jde nakonfigurovat každý detail aplikace. Spring Boot obsahuje vestavěný webový server Tomcat, na kterém běží vyvinutá aplikace. Důvodem výběru tohoto frameworku je také to, že tato technologie vznikla v roce 2002 a obsahuje spoustu užitečné dokumentace a vyřešených nedostatků, se kterými by se mohl setkat programátor.

[29] Hypertext Transfer Protocol

3.5.4.2 MyBatis

Jak jsem napsal v kapitole 2.2 ve své aplikaci pracuji s daty, která byla předem pomocí externí aplikace stáhnutá a načtená do databáze. Pro účely naší aplikace nepotřebujeme všechny informace, které jsou uložené v databázi ale jen jejich část. Proto jsem pro práci s daty z databáze použil Java framework MyBatis[30]. MyBatis je JPA[31] framework, který slouží jako vrstva abstrakce nad databází pro lehčí práci s ní. Tento framework umožňuje programátorovi snadno definovat metody pro čtení a zápis do databáze.

Na rozdíl od ORM[32] frameworků, které mapují Java objekty do tabulek databáze, MyBatis mapuje Java metody do SQL[33] příkazů. Toto je dosaženo pomocí definování SQL příkazů v XML souboru nebo pomocí anotací a definování rozhraní, které mapuje metody na SQL příkazy pomocí anotací.

Používání MyBatis je speciálně vhodné v momentě, kdy programátor používá již vytvořenou databázi a nepotřebuje využívat celé schéma, ale jen nějakou jeho část. MyBatis umožňuje nadefinovat složíte SQL příkazy v zvláštním XML souboru a tím ulehčit práci pro programátora při volání těchto příkazů.

3.5.5 Frontend

Na rozdíl od backendu, frontend je část aplikace, kterou vidí návštěvník stránek. Ve své práci jsem použil template engine Thymeleaf[34], který je zaměřen na generování stránek na straně serveru webové aplikace.

3.5.5.1 Thymeleaf

Thymeleaf je javovský XML/XHTML/HTML5 template engine, který může být použit pro webové a newebové aplikace. Hlavním záměrem je obsluha XHTML/HTML5 v prezentační vrstvě webových aplikací založených na MVC architektuře, ale může zpracovat libovolný soubor XML i v offline prostředí. Thymeleaf je open-source software šířený pod licenci Apache License 2.0. [14]

Hlavní výhody Thymeleaf jsou:

- modul pro integraci se Spring Boot,
- podpora XML, XHTML verze 1.0 a 1.1, HTML5,
- podpora offline režimu,
- plná dokumentace.

[30] http://www.mybatis.org/mybatis-3/
[31] Java Persistence API
[32] Object–relational mapping (Objektově relační mapování)
[33] Structured Query Language
[34] https://www.thymeleaf.org/

Základem práce v tomto frameworku je psání HTML5 kódu s přidáním speciálních atributů do HTML tagů pro generování dat ze serveru. HyperText Markup Language je značkovací jazyk používaný pro tvorbu webových stránek. V době psaní této bakalářské práce je HTML verze 5 hlavním z jazyků pro vytváření stránek pro webové aplikace. Template engine Thymeleaf podporuje offline režim, v případě nedostupnosti serveru stejně vygeneruje stránku, ale bez chybějících údajů.

3.5.6 Databáze

Pro ukládání a získávání dat je použit databázový systém PostgreSQL[35], který je open source projektem a je dostupný ke stažení zdarma. PostgreSQL je populární objektově-relační databázový systém, který se vyvíjí mnohá firmami, a proto existuje spousta nástrojů pro snadnou práci s ním a integrací do různých programovacích jazyků. Pro práci s databází v Javě je potřeba stáhnout PostgreSQL JDBC[36] ovladač a nakonfigurovat cestu ke staženému souboru.

3.5.7 Maven

Apache Maven[37] je nástroj pro správu, řízení a automatizaci sestavování aplikací. Maven se stará o správné sestavení aplikace, dokumentaci aplikace a dotažení potřebných knihoven. Základní princip fungování Mavenu je řízen souborem POM[38]. Tento soubor je ve formátu XML a lze v něm nadefinovat různé údaje o aplikaci jako:

- groupId - skupina, do které patří projekt,
- artifactId - jednoznačný identifikátor aplikace ve skupině,
- name - název projektu,
- kódování projektu,
- dependencies - závislosti na jiných knihovnách.

Díky rozsáhlostí své funkcionality a nahrazení nedostatků ostatních aplikací, jako Apache Ant[39], je Maven jedním z nejpoužívanějších nástrojů ve světě Java.

[35] https://www.postgresql.org/
[36] Java Database Connectivity
[37] https://maven.apache.org/
[38] Project Object Model
[39] https://ant.apache.org/

3.5.8 Verzovácí systém

Pro účely bakalářské práce byl použit verzovácí systém GitLab[40], který je volně dostupný pro studenty FITu[41].

3.6 Databázový model

Databázový model je typ modelu, který popisuje logickou strukturu databáze aplikace a základní princip uložení dat do databáze. Často databázový model hraje velkou roli při vývoji databáze a návrhu struktury aplikace. Jak jsem uvedl v kapitole 2.2 moje aplikace používá data z již hotové externí databáze a pro návrh dalších modulů aplikace je podstatné nejenom pochopit princip fungování databáze a mít i přehled o její struktuře. Pro modelování databázového modelu jsem použil modelovací jazyk UML a nástroj pro modelování Enterprise Architect.

Model je tvořen sadou entit, které jsou mezi sebou provázány pomocí vazeb s příslušnou kardinalitou. Každá entita reprezentuje skupinu objektů reálného světa a obsahuje množinu atributů charakteristických pro danou skupinu. Mezi hlavními entitami patří Record a Entity, které jsou znázorněny na obrázku 3.7.

Entita Record reprezentuje dokument, ke kterému chci poskytnout přístup pro běžného uživatele. Hlavními atributy jsou:

- amountCZK - částka, která se vztahuje k dokumentu,
- currency - měna, ve které byla částka zaplacena,
- date_created - datum vytvoření dokumentu,
- date_of_payment - datum zaplacení částky z dokumentu,
- master_id - unikátní identifikátor dokumentu,
- variable_sysmbol - variabilní symbol platby,
- record_type - typ dokumentu,
- subject - název dokumentu nebo krátký popis.

Entita se vztahuje k sadě dalších entit z našeho modelu. Tyto vztahy jsou realizovány pomocí cizích klíčů a znázorněny pomocí šípek, které vedou do a z jiných entit. Entita Record obsahuje atributy **authority** a **partner**, které slouží jako odkazy na tabulku Entita, což vyjadřuje vztah dokumentu k organizaci, která je plátcem a organizaci, která je dodavatelem. Tento vztah má kardinalitu 0..*:1 a znázorňuje, že jedna organizace může být plátcem a dodavatelem ve více dokumentech. Zajímavým faktem je, že entita Record drží

[40] https://gitlab.fit.cvut.cz/
[41] Fakulta Informačních Technologií

odkaz sama na sebe a tím je implementován vztah mezi dokumenty, které spolu souvisí.

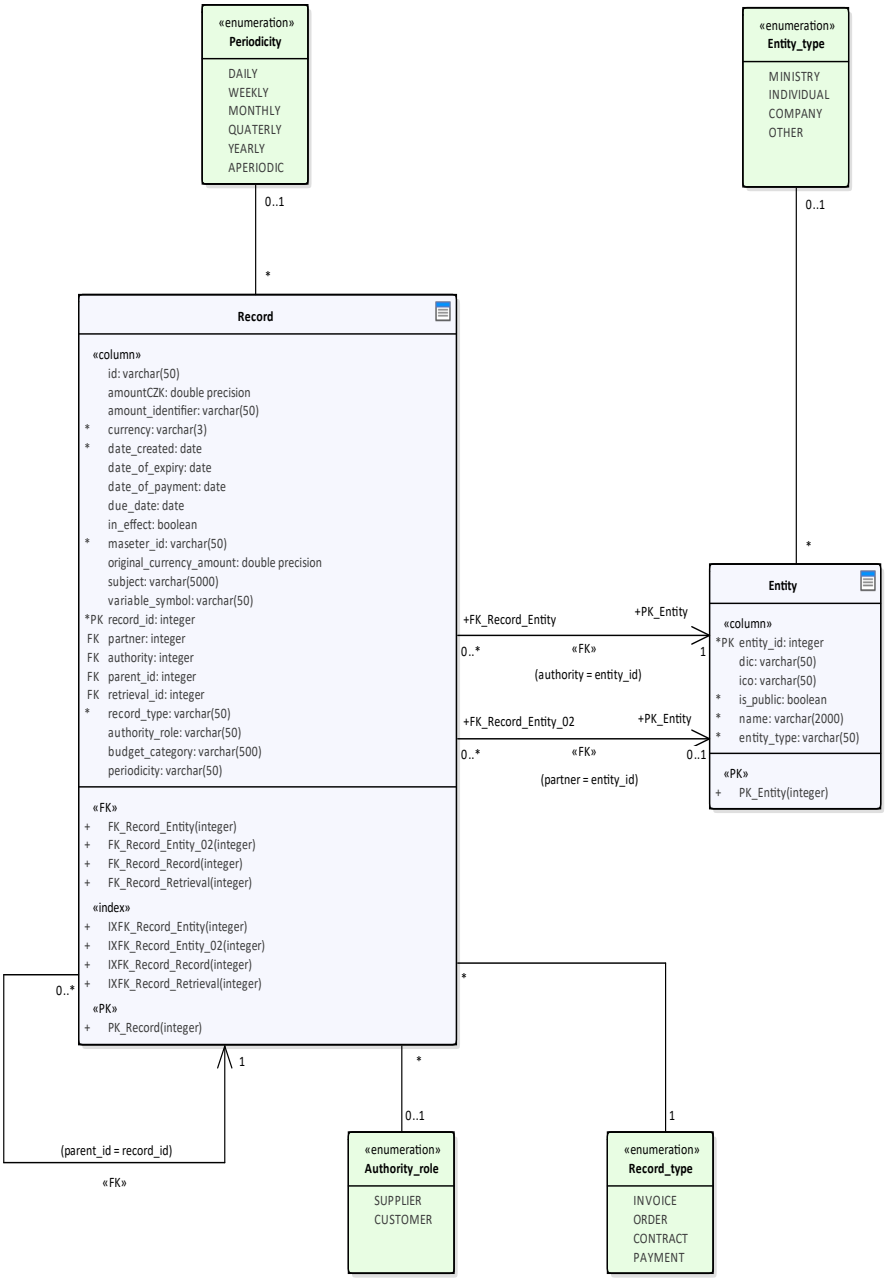
Entita Entity reprezentuje organizaci, která je plátcem anebo dodavatelem u nějakého dokumentu uloženého v databázi aplikace. Obsahuje tyto atributy:

- entity_id - vnitřní unikátní identifikátor organizace,
- dic - daňové identifikační číslo (DIČ), jednoznačná identifikace daňového subjektu,
- ico - identifikační číslo osoby(IČO), unikátní osmimístné identifikační číslo právnické osoby, podnikající fyzické osoby nebo organizační složky státu,
- is_public - jestli informace o organizaci je veřejná,
- name - název organizace,
- entity_type - typ organizace.

Jak jsem naznačil v předchozím odstavci entita Entity se hlavně vztahuje k entitě Record.

Model obsahuje přehled nejenom o tabulkách, které slouží pro ukládání dynamické informace a i o tabulkách, které obsahují výčet hodnot. Tabulka, která obsahuje výčet hodnot je implementována v databázi jako obyčejná tabulka s jediným rozdílem v tom, že data se tam uloží jednou při startu aplikace jako konstanty a pak se používají během dalších operací v databázi/aplikaci. Jednou z takových tabulek je Record_type, která drží všechny možné hodnoty pro typ dokumentu. Na modelu je znázorněno, že obsahuje čtyři hodnoty a to jsou **INVOICE** (faktura), **ORDER** (objednávka), **CONTRACT** (smlouva), **PAYMENT** (platba). Tabulky výčtového typu na diagramu modelu označuji zelenou barvou. Výčtové typy mají spoustu výhod, mezi které patří:

- zabranění redundance informace v databázi,
- šetření místa v databázi,
- přehlednost databáze,
- kontrola hodnot v atributech u jiných tabulek.



Obrázek 3.7: Databázový model: Entity a Record

Kapitola 4

Implementace

Tato kapitola popisuje postup implementace aplikace a použití jednotlivých technologií. Při vývoji aplikace jsem se řídil návrhem udělaným v předchozí kapitole. Kapitola obsahuje 5 podkapitol, ve kterých jsem popsal problémy, se kterými jsem se potkal během řešení jednotlivých vrstev aplikace.

4.1 Založení projektu

Jak jsem zmiňoval v kapitole 3.5.4.1 pro implementaci projektu byl použit framework Spring Boot a vývojové prostředí IntelliJ IDEA. Toto IDE nabízí různé možnosti vytvoření Java projektů. Jednou z takových možností je použití Spring Initializr[42]. Spring Initializr je nástroj pro rychlou inicializaci potřebných baličků a souborů pro start Spring Boot aplikace. Tento nástroj lze použít ve webové verzi na stránce `https://start.spring.io/` a také v IntelliJ IDEA. Pro založení projektu je potřebné vyplnit povinná pole Group (Skupina), Artifact (Artifakt), Type (Typ), Description (Popis) a nechat vychozí hodnoty v ostatních polích. Pak následuje výběr závislostí, kde jsem se řídil seznamem technologií z kapitoly 3.5. Zvolil jsem si Spring Boot verze 2.0.0, což je nejnovější verze frameworku v době psaní aplikace. Mezi závislosti aplikace patří:

- Web - artifactId „spring-boot-starter-web“,
- Template Engines - Thymeleaf, artifactId „spring-boot-starter-thymeleaf“,
- SQL - MyBatis, artifactId „mybatis-spring-boot-starter“,
- SQL - PostgreSQL, artifactId „postgresql“.

Po dokončení inicializačního průvodce se mi vytvořil nový projekt s již vyplněným POM.xml, o kterém jsem psal v kapitole 3.5.7. Projekt obsahuje

[42] https://start.spring.io/

jedinou třídu s názvem `OpendataApplication` s **main** metodou aplikace. Tato třída je anotovaná pomocí `@SpringBootApplication` anotace. Vysvětlení této anotace lze nalézt v oficiální dokumentaci frameworku[43]. Spring Boot přináší novou anotaci `@SpringBootAplication`, která nahrazuje funkčnost následujících anotací: `@Configuration`, `@EnableAutoConfiguration`, `@ComponentScan`.

Anotací `@Configuration` se označuje třída, která obsahuje deklaraci jedné či více java bean metod a může být použítá Spring IoC kontejnerem pro generování definic bean. Třída s touto anotací může být použitá pro konfiguraci jiných frameworků v rámci aplikace. Tato anotace je obdobou XML konfiguračního souboru používaného pro definici java bean v Spring frameworku.

Anotace `@EnableAutoConfiguration` se obvykle používá pro hlavní třídu aplikace a konfiguruje Spring aplikaci na základě nastavení cesty ke knihovnám, dalších bean a jiných konfiguračních souborů.

Anotace `@ComponentScan` říká Spring Boot frameworku, aby provedl auto detekci všech tříd anotovaných pomocí následujících anotací: `@Component`, `@Controller`, `@Service` a `@Repository`. Důsledkem procesu detekce je registrace těchto tříd do Spring IoC kontejnerů a možnost jejich vnoření pomocí anotace `@Autowired`.

4.2 Struktura aplikace

Na obrázku 4.1 je vidět struktura projektu. Základní kostra byla vytvořena pomocí inicializačního průvodce, viz předchozí kapitola. Díky vyběru „Maven project“ u typu aplikace se nám vygenerovala struktura, která je charakteristická pro všechny Maven aplikace. Sledování této konvence umožňují novému programátorovi se rychle a lehce vyznat v novém projektu. Specifikaci struktury složek v **Maven** projektu lze nalézt v oficiální dokumentaci nástroje[44].

Balíček `cz.cvut.fit.opendata.model` obsahuje třídy, které reprezentují POJO[45] aplikace sloužící pro manipulaci s daty z databáze. `RecordDataMapper` a `EntityDataMapper` se nachazejí v balíčku `cz.cvut.fit.opendata.mapper` a slouží jako rozhraní pro volání nadefinvaných databázových selectů. Balíček `cz.cvut.fit.opendata.rest` je modul aplikace, který slouží jako logická vrstva aplikace. Třídy `RecordController` a `EntityController` se starají o:

- komunikaci s uživatelem,
- zpracování HTTP požadavků jak na webový portál, tak i na vystavené API aplikace,
- generování stránek pro webový portál,

[43] `https://docs.spring.io/spring-boot/docs/current/reference/html/using-boot-using-springbootapplication-annotation.html`

[44] `https://maven.apache.org/guides/introduction/introduction-to-the-standard-directory-layout.html`

[45] Plain Old Java Objects

```
main.java.cz.cvut.fit.opendata
        |
        +---mapper.....................................................Databázové rozhraní
        |       EntityDataMapper.java
        |       RecordDataMapper.java
        |
        +---model..............................................Business objekty aplikace
        |   |   AuthorityRole.java
        |   |   DataInstance.java
        |   |   DataSource.java
        |   |   Entity.java
        |   |   EntityType.java
        |   |   PartnerListEntry.java
        |   |   Periodicity.java
        |   |   Record.java
        |   |   RecordType.java
        |   |   Retrieval.java
        |   |   UnresolvedRelationship.java
        |   |
        |   \---exception.............................................Výjimky aplikace
        |           ValidationException.java
        |
        +---rest.............................Kontroléry a správa požadavku od uživatele
        |       EntityController.java
        |       PagesController.java
        |       RecordController.java
        |
        \---service.............................................Rozhraní služeb aplikace
                |   LinkResolverService.java
                |   PaginationService.java
                |   TemplateService.java
                |   ValidationService.java
                |
                \---impl.......................................Implementace služeb aplikace
                                LinkResolverServiceImpl.java
                                PaginationServiceImpl.java
                                TemplateServiceImpl.java
                                ValidationServiceImpl.java
```

Obrázek 4.1: Struktura aplikace

- vnitřní logiku aplikace,
- rozdělení dílčích úkolů na služby a datové rozhraní aplikace.

`cz.cvut.fit.opendata.service` obsahuje seznam dostupných služeb aplikace. Služba (service) je seznam operací nabízených uživateli ve formě rozhraní, které není závislé na modelu aplikace, bez zapouzdřování stavu [15, str. 105]. Můžeme si službu představit jako rozhraní s definovanými metodami, které slouží pro řešení menších často opakujících úkolů. Výhody služeb jsou:

- lepší udržitelnost kódu do budoucna,
- znovupoužitelnost kódu,
- přehlednost kódu,
- zabránění „špagetového kódu“,

- možnost delegace požadavků na jiné moduly aplikace.

Podle návrhového vzoru „Strategy Pattern“ každá služba kromě svého rozhraní může obsahovat jedno nebo více implementací [16], které jsou umístěné v `cz.cvut.fit.opendata.service.impl`.

Složka `resources` obsahuje konfigurační soubory aplikace, CSS styly, JavaScipt soubory a šablony stránek pro webový portál. Soubory v balíčku `resources.mapper` jsou součástí frameworku MyBatis a slouží pro definici SQL selectů pro aplikaci. Balíček `resources.templates` obsahuje šablony pro template engine Thymeleaf.

Druhou částí aplikace jsou testy, které se nachází ve složce `test`. Popisu procesu testování se věnuji v kapitole 5.

4.3 Implementace business objectů

Podle [17] business object je objekt v business vrstvě objektně–orientováné aplikace, který představuje část podniku nebo položku v něm. Business object reprezentuje data v aplikaci a může být implementován jako entity bean, session bean nebo jiný javovský objekt. Business object může sloužit jako datová vrstva aplikace, ale ne samotná databáze. Business objecty jsou škálovatelné díky architektuře objektově orientovaných softwarových aplikací.

Při vývoji velkých aplikací se často stává, že data uložená v databázi aplikace musí být konvertována do business objektů aplikace, aby odpovídala business modelu a návrhu aplikace. Díky správnému návrhu databáze mojí aplikace není potřeba provádět velké úpravy nad daty dotaženými z databáze před tím, než se dostanou na prezentační vrstvu aplikace. Všechny drobné úpravy se provedou pomocí využití funkčností template engine.

Ve své aplikaci jsem implementoval business objekty jako POJO, které se nachází v balíčku `cz.cvut.fit.opendata.model`. Při implementaci jsem se hlavně řídil databázovým a doménovým modelem. Snažil jsem se o implementaci jenom těch objektů, které pak použiji v aplikaci a vynechaní nepotřebných tříd a atributů pro minimalizaci nepoužívaných dat. Příkladem implementace je třída `Entity`, která se skládá z deklarace jejích atributů, getterů a setterů pro tyto atributy, metod `equals`, `hashCode` a `toString`. Mezi atributy patří:

- `id`, typ `Long`,
- `dic`, typ `String`,
- `ico`, typ `String`,
- `name`, typ `String`,
- `isPublic`, typ `Boolean`,
- `entityType`, typ `EntityType`,

- `recordsAsAuthority`, typ `List<Record>`,
- `recordsAsPartner`, typ `List<Record>`.

`Equals(Object o)` a `hashCode()` jsou metody označeny `@Override` anotací, protože přepisují a poskytují svou vlastní implementaci metod základní třídy `Object` v Javě, ze které dědí všechny ostatní objekty. Tyto metody jsou nutné k implementaci u objektů mojí aplikace pro její korektní používání v kolekcích. Více o těchto metodách a jejich implementaci je popsáno v knize „Effective Java“. [18].

Ostatní business objekty aplikace jsou implementovány stejným způsobem.

4.4 Implementace databázových volání

Pro práci s databází a daty jsem použil framework MyBatis. Výhodou využívání tohoto framewrku je také to, že je integrován s frameworkem Spring Boot, který využívám pro vývoj svojí aplikace.

Pro instalaci MyBatis je potřeba přidat závislost s artifactid „mybatis-spring-boot-starter“ verze 1.3.1 do POM.xml souboru. Díky tomuto balíčku zajistíme:

- auto detekci `DataSource` aplikace,
- vytvoření a registraci instance `SqlSessionFactory`,
- vytvoření a registraci `SqlSessionTemplate`,
- skenování existujících mapperu v aplikaci a jejich registraci do kontextu Spring, aby je možná bylo vkládat a používat v jiných komponentách programu.

MyBatis umožňuje definovat SQL příkazy v anotaci nad metodou v mapperu nebo v XML souboru. Aby aplikace byla přehlednější a z důvodu složitosti některých dotazů, jsem si zvolil způsob napsaní SQL příkazů v XML souboru. Registrace XML souboru se provádí v `applications.properties` pomocí příkazu `mybatis.mapper-locations=classpath:mapper/*.xml`, kde uvádím cestu k XML mapperům. Stejným způsobem je možné nadefinovat i ostatní konfigurace pro tento framework. Pro učely mojí práce vystačím s výchozími nastaveními.

XML mapper je základem práce frameworku MyBatis, který obsahuje seznam tvrzení pro konfiguraci různých SQL příkazů jako **SELECT**, **UPDATE**, **INSERT** a **DELETE**. Tato tvrzení se nazývají Mapované Tvrzení nebo Mapované SQL Tvrzení.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper
        PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="cz.cvut.fit.opendata.mapper.RecordDataMapper">

    <select id="getRecordsCountByTypes" parameterType="map" resultType="Long">
        SELECT COUNT(*)
        FROM RECORD r
        <if test="organizationId != null">
            JOIN entity e
            ON r.authority = e.entity_id
            OR r.partner = e.entity_id
        </if>
        WHERE
        (r.record_type = ''
        <foreach collection="recordTypes" item="element" index="index">
            OR r.record_type = #{element}
        </foreach>
        )
        <if test="organizationId != null">
            AND e.entity_id = #{organizationId}
        </if>
        ;
    </select>
```

Obrázek 4.2: XML mapper: ukázka

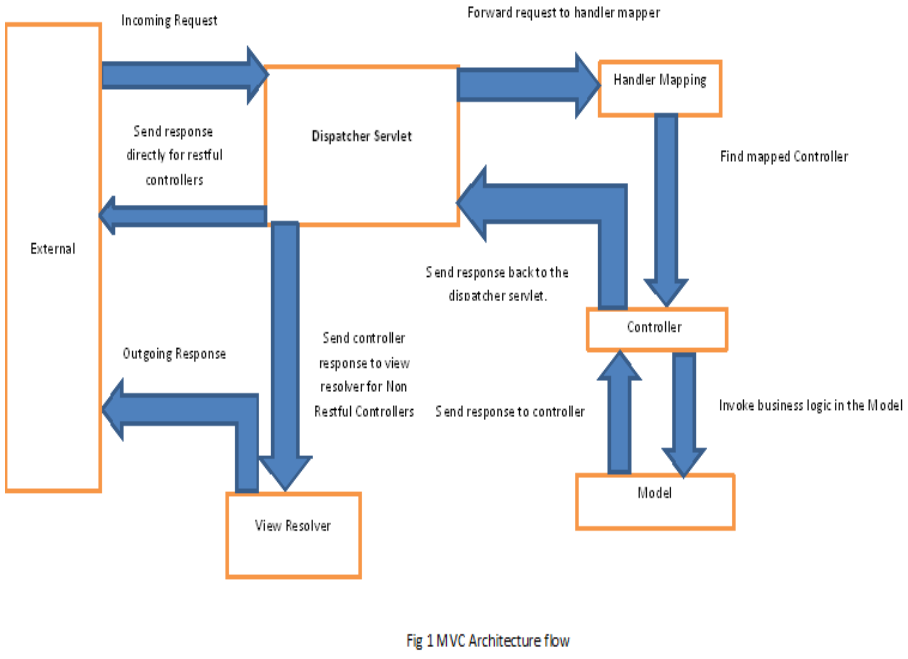
- Každé tvrzení má unikátní id, které se používá pro provedení daného tvrzení.

- Všechna tvrzení se nachází uvnitř elementu **mapper**. Tento element má atribut **namespace**, který odkazuje na Java rozhraní pro volání těchto tvrzení.

MyBatis podporuje používání dynamického SQL v XML Mapperu. Dynamické SQL je technika programování, která umožňuje vytvářet SQL příkazy za běhu programu. Díky této vlastnosti je možné SQL příkaz měnit, doplňovat v závislosti na vstupních parametrech příkazu. MyBatis podporuje používání podmínek typu „if“, „when“, „switch“. Na obrázku 4.2 je vidět část kódu XML mapperu.

Aby nadefinované SQL příkazy bylo možné používat, je potřeba vytvořit rozhraní pro jejich volání. Rozhraní je anotováno pomocí anotace @Mapper, která umožňuje Springu zaregistrovat tento Mapper do Spring Contextu. Část rozhraní je vidět v následujícím kódu.

```
@Mapper
public interface RecordDataMapper {
Long getRecordsCountByTypes(@Param("recordTypes") List
    <String> recordTypes, @Param("organizationId") Long
     organizationId);
}
```



Obrázek 4.3: Architektura toku dat v rámci MVC [1]

4.5 Implementace controllerů

V následujících dvou podkapitolách se soustředím na popis hlavní komponenty Spring MVC - Controlleru.

Na obrázku 4.3 je vidět architekturu toku dat v Spring MVC. Controller hraje roli jednotky, která se zabývá zpracováním požadavku od uživatele po tom, co budou zpracovány Dispatcher Servletem a namapovány Handler Mappingem na jednotlivé metody. Díky používání Spring Boot frameworku se nestarám o přijetí HTTP požadavku a routing těchto požadavku na metody.

Třída, která hraje roli controllera v mojí aplikaci, je označena anotací `@RestController`. Stereotyp `@RestController` nebo `@Controller` umožňují automatickou detekci a registraci těchto komponent do Spring kontejneru

při startu aplikace. Tato anotace také naznačuje roli těchto bean jako web komponent aplikace.

Mapování HTTP požadavku na metody v controlleru se provádí na základě anotace `@RequestMapping`[46]. Anotace `@RequestMapping` podporuje parametry uvedené v dokumentaci. V mojí aplikaci jsem použil jenom nejdůležitější:

- `value` - základní mapování vyjádřené touto anotací,
- `produces` - typ dat, který se vytváří tímto mapovaným požadavkem („application/json“, „application/xml“),
- `method` - typ požadavku, který se zpracuje mapovanou metodou (PUT, GET, POST),
- `consumes` - typ dat, který konzumuje mapována metoda.

4.5.1 RecordController

Třída `RecordController` je zodpovědná za poskytování informace o dokumentech, které jsou v databázi mojí aplikace. Tento controller obsahuje 4 metody anotovány pomocí `@RequestMapping`, dvě ze kterých jsou určeny pro správu web portálu a další dvě pro API aplikace. Každá z těchto metod a způsob jejího používání je detailně popsán v dokumentaci k aplikaci. `RecordController` využívá rozhrání `RecordDataMapper` pro komunikaci s databází aplikace, služby `PaginationService` a `TemplateService` pro řešení problémů stránkování a základního plnění šablon pro stránky webového portálu. Jak jsem naznačil v kapitole 4.4 mapper pro práci s databází a služby jsou označeny pomocí anotací `@Mapper` a `@Service`, což umožňuje jejich registraci do Spring kontejneru při startu aplikace. Pro jejich použití v kontroléru stačí využít návrhový vzor „Depndency Injection“, který je implementován v Spring Boot frameworku. Vkládání závislostí v Spring Boot je implementováno ve dvou základních variantách: constructor-based dependency injection a setter-based dependency injection. Popis každé varianty lze nalézt v oficiální dokumentaci frameworku Spring [19]. Pro účely své aplikace používám první variantu. Třída využívá službu `ValidationService` pro validaci a konverzaci dat, které přijdou v HTTP požadavku, speciálně tedy parsování a validace typu dokumentu, který požádá uživatel webového portálu nebo API.

4.5.2 EntityController

`EntityController` a `RecordController` jsou implementovány stejným způsobem, ale s rozdílem v poskytované informaci. `EntityController` se stará o požadavky na seznam organizací evidovaných v databázi a další informace

[46] https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/web/bind/annotation/RequestMapping.html

s nimi spojené. Třída obsahuje 6 metod anotovaných pomocí `@RequestMapping`, ze kterých tři jsou určeny pro správu API a další tři pro webový portál. Rozdíl `EntityController` a `RecordController` je v mapperu, který se používá pro dotažení dat. `EntityController` využívá metody `EntityDataMapper`.

4.6 Implementace webových stránek

V následujících dvou podkapitolách se zaměřím na popis třetí části Spring MVC - View a její implementaci ve své aplikaci.

Pro vytvoření prezentační vrstvy aplikace jsem si zvolil template engine Thymeleaf. Pro použití tohoto nástroje je potřeba přidat závislost s artifactId `spring-boot-starter-thymeleaf`.

4.6.1 Implementace fragmentů

Stránky mého webového portálu sdílejí společné komponenty, jako je záhlaví, zápatí nebo stránkování. Template engine Thymeleaf umožňuje znovupoužití stejných částí HTML kódu. Pomocí atributů `th:insert` a `th:replace` lze zařadit část jiné stránky jako fragment. Výhodou je možnost zařazení jenom části stránky, zatímco jiné technologie jako JSP[47] zařazují jenom celé stránky.

Ve balíčku `resources.templates.fragments` se nacházejí fragmenty mojí aplikace. Mezi hlavními fragmenty jsou:

- `header.html` - záhlaví stránky,
- `assets.html` - odkazování na CSS styly a JavaScripty,
- `pagination.html` - stránkování.

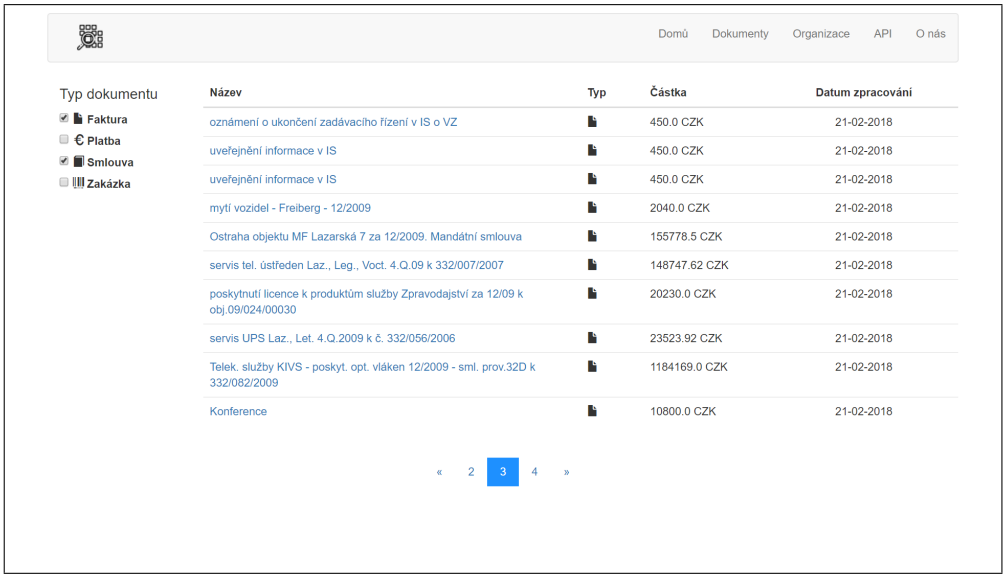
4.6.2 Implementace šablon

Pro implementaci šablon stránek jsem použil jazyk HTML5 a pro jejich stylování CSS. Šablony se nacházejí ve složce `resources.templates`. Při implementaci šablon webových stránek jsem se řídil navrženými wireframe v kapitole 3.4.

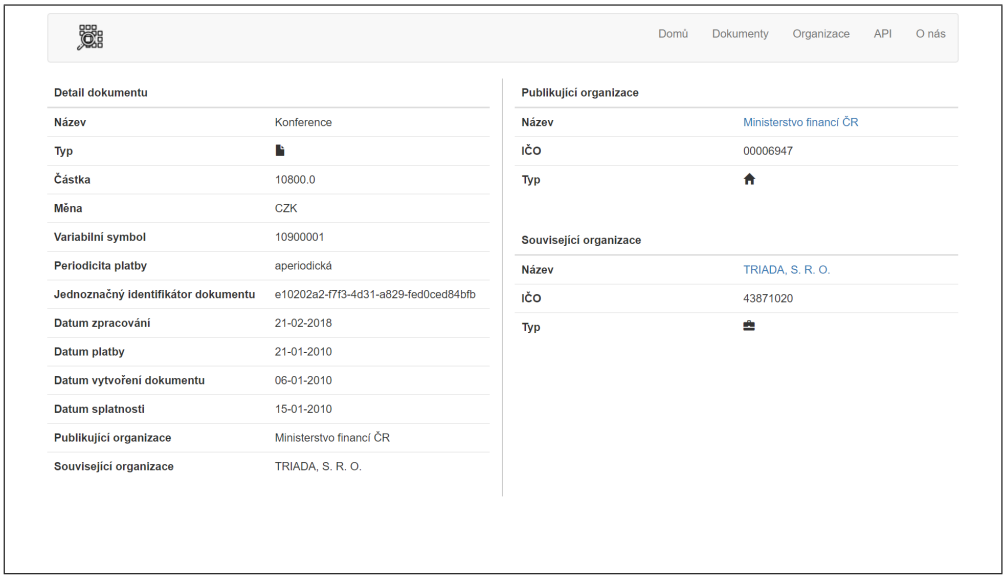
Při implementaci šablon jsem se nesnažil o implementaci jedné hlavní šablony, do které by pak zařazoval potřebné fragmenty podle typu stránky. Naopak jsem naimplementoval zvlášť šablonu pro přehled dokumentů, detail dokumentu, přehled organizací a detail organizace. Výhodou tohoto způsobu je možnost navržení nového stylu některých stránek bez toho, aby se měnily všechny ostatní.

Část hotových stránek webového portálu je vidět na následujících obrázcích.

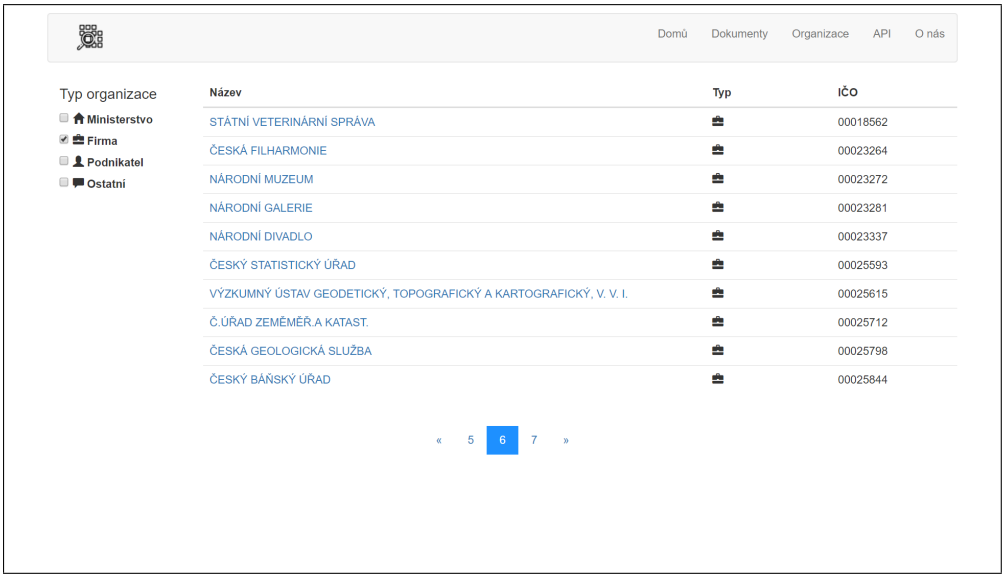
[47] Java Servlet Pages



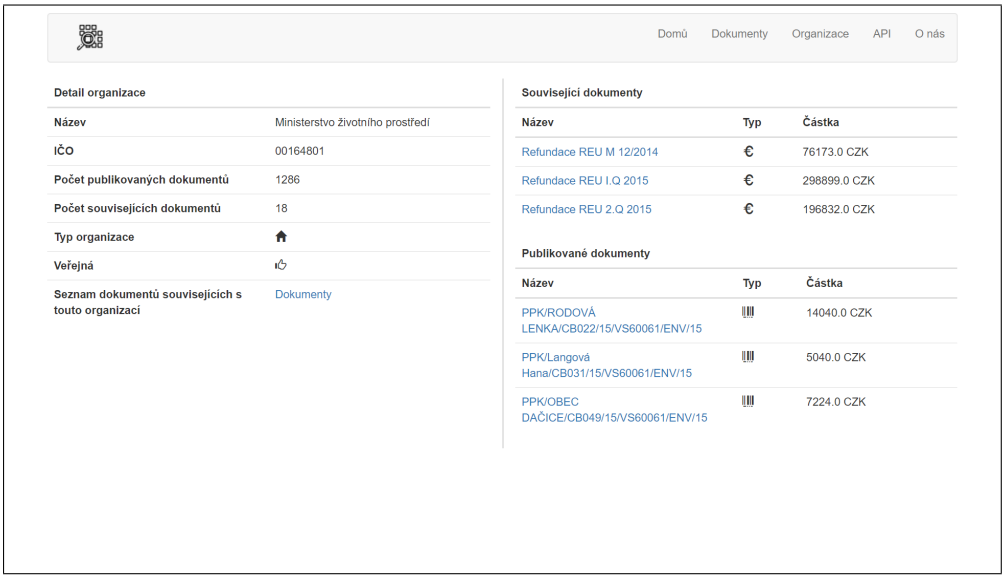
Obrázek 4.4: Obrazovka „**Dokumenty**“



Obrázek 4.5: Obrazovka „**Detail dokumentu**“



Obrázek 4.6: Obrazovka „**Organizace**“



Obrázek 4.7: Obrazovka „**Detail organizace**“

Kapitola 5

Testování

5.1 UNIT testování databázového rozhraní

S ohledem na způsob implementace datové vrstvy a existence jednoduché logiky na úrovni selectu v `RecordDataMapper.xml` a `EnityDataMapper.xml` je potřeba provést unit testování této vrstvy. Unit testování je metoda testování softwaru, pomocí které se testují jednotlivé jednotky zdrojového kódu, sady jednoho nebo více modulů počítačového programu spolu se souvisejícími kontrolními daty, použití procedur a konající procedury, aby se zjistilo, zda jsou vhodné pro použití. [20, str. 75]

Pro práci s testy používám framework JUnit verze 4.12[48]. Framework JUnit přináší spoustu pro testování vhodných technologií, užitečných knihoven a anotací, které ulehčuji proces napsání a spouštění unit testů. Mezi ně patří statické metody typu `assert()`, které slouží pro ověřování výsledků testovaných funkčností, anotace `@Test`, `@RunWith`, `@Before` pro konfiguraci testů.

Testovací třídy se nacházejí ve složce `src.test` a struktura složek odpovídá struktuře projektu. Třída `RecordDataMapperTest` obsahuje 6 testů, které kontrolují funkčnost metod v `RecordDataMapperu`. Třída `EntityDataMapperTest` se stará o testování metod a funkčností v `EntityDataMapper`.

Pro ověření správnosti výsledků se testy data mapperů provádí na základě testovací databáze, která obsahuje testovací data. Pro tyto účely byla vytvořena testovací databáze s názvem `opendata_test`. Struktura testovací databáze je kopií struktury databáze používané na produkčním serveru. Scripty pro vytvoření databáze a její plnění daty jsou součástí instalační příručky k aplikaci.

[48] `https://junit.org/junit4/`

5.2 UNIT testování služeb

Při implementaci aplikace jsem se snažil rozdělovat požadavky od klientů na menší požadavky a delegovat je na zpracování vytvořenými služby. Proto je velká část funkčností a logiky aplikace založena na korektním chování služeb: `ValidationService`, `LinkResolverService` a `PaginationService`.

V testech `LinkResolverServiceTest` jsem otestoval správné chování metod služby pro generování potřebných odkazů pro webovou aplikaci. Protože funkčnost služby je založena na použití aktuálního `HttpServletRequest` od klienta, v testech jsem použil třídu `MockHttpServletRequest` ze Spring balíčku pro mockování.

Služba `PaginationService` je zaměřená na generování správných čísel stránek v závislosti na vstupních parametrech pro stránkování webového portálu. Mezi metody služby patří validace aktuální stránky, výpočet následující a předchozí stránky, výpočet počtu záznamů, který by měl vrátit databázový dotaz. V testech jsem kontroloval korektní výstupy z metod pro výpočet oproti předpokládaným hodnotám.

Mezi funkčnosti služby `ValidationService` patří validace hodnot, které přijdou v požadavku od klienta, a jejich převedení do požadováného pro další práci aplikace stavu. Ověřil jsem správné chování metod na parsování typů dokumentů a typů organizací, jejich validaci oproti povoleným hodnotám ve výčtových typech. Kontroloval jsem i případy, kdy musí být vyhozena výjimka. Anotace `@Test(expected = ValidationException.class)` umožňuje nejenom odchytit výjimku, ale i provést kontrolu typu odchycené výjimky.

Kapitola 6

Možnosti rozšíření aplikace

Implementována aplikace je hotová a splnila všechny cíle uvedené na začátku. Vytvořený webový portál je začátkem nového projektu v oblasti otevřených dat, který by šlo rozšiřovat o nové funkčnosti. Mezi hlavní nápady na rozšíření funkcionalit patří:

- registrace uživatele,
- registrace publikující organizace,
- nahrání nových záznamů do databáze pro ověřené organizace,
- vizualizace dat a užitečné statistiky.

Všechny uvedené nápady lze realizovat jako nové komponenty s minimálním dopadem do stávajících částí aplikace.

Pro mě je hlavní myšlenkou projektu Otevřená data vybudování otevřené infrastruktury v České republice a zabránění korupce v jakémkoliv její projevu. Data, se kterými pracuje moje aplikace, je možné analyzovat pomocí moderních algoritmů a strojů, což by vedlo na objevování závislostí, které není vidět na první pohled. V první etapě lze realizovat kontrolu publikovaných dokumentů, které byly uzavřeny mezi dvěma různými ministerstvy. Jeden a ten samý dokument měl by být publikován dvěma různými ministerstvy, což by umožnilo zkontrolovat případné chyby v publikaci u publikující strany a upozornit ji na to. V dalších etapách by se dalo mluvit o kontrole dat v dokumentech mezi ministerstvy a firmami pro objevování případných nesprávností.

Záměrem vytvořené aplikace je poskytování dat pro veřejnost. V době psání bakalářské práce databáze aplikace obsahovala přes 300 tisíc záznamů o dokumentech různých typů dostupných pro veřejnost. Objevování nových datových zdrojů a nahraní datových sad z těchto zdrojů do databáze webového

portálu je pro realizaci jedním z nejlehčích nápadů na rozšíření. Díky správnému návrhu a implementaci projektu stačí nahrát nové datové sady do databáze aplikace a korektní přehled a chování aplikace zajistí sama.

V případě dalšího rozvoje aplikace by bylo vhodné použít moderní technologie a principy ze softwarového procesu. Pokrytí unit testy všech modulů aplikace by odhalilo další implementační chyby a umožnilo snadnější udržitelnost do budoucna. Nasazení aplikace do produkčního prostředí a otestování reálnými uživateli by umožnilo obdržet zpětnou vazbu na aplikaci a nápady na nové funkčnosti aplikace. Zavedení *Continuous Integration*[49] a *Continuous Delivery*[50] pomocí nástroje Jenkins[51] by zajistilo automatické sestavování aplikace, provedení automatizovaných testů, pravidelnou kontrolu kódu a vydávání nových verzí aplikace.

[49] Kontinuální integrace
[50] Průběžné vydávání nových verzí
[51] https://jenkins.io/

Závěr

Cílem práce bylo navrhnout a vytvořit webovou aplikaci pro podporu projektu Otevřena data v České republice. Zároveň teoretická část práce by měla sloužit jako úvod k orientaci v daném tématu pro někoho, kdo se s pojmem „otevřená data“ v životě nikdy nesetkal.

Na začátku práce jsem provedl analýzu stávajících řešení, kde jsem identifikoval jejich hlavní výhody a nevýhody. Na základě této analýzy a popisu dat, se kterými bude pracovat aplikace, jsem vymezil funkční a nefunkční požadavky kladené na nový systém.

V části návrhu aplikace jsem se zaměřil na podrobnější popis funkčních požadavků pomocí definování případů užití. Další podkapitolou této části byl návrh obrazovek, kde byl kladen důraz na jednoduchý a minimalistický design. Návrhová část také obsahuje rozbor technologií použitých pro implementaci nové aplikace a popis databázového modelu.

Předposlední částí byla implementace navržené aplikace, kde byl popsán proces založení nové aplikace a konfigurace použitých frameworku. Byly popsány jednotlivé komponenty aplikace a jejich propojení mezi sebou. Při implementaci byl kladen důraz na čitelnost kódu a použití návrhových vzorů, aby aplikace byla snadno rozšiřitelná do budoucna.

Poslední kapitola práce se zabývá unit testováním funkčních modulů aplikace. Testy byly zaměřeny na odhalení implementačních chyb a udržitelnost aplikace do budoucna.

Cíle kladené na začátku práce byly úspěšně dosaženy.

Literatura

[1] Dutta, P.: Quick Guide to Spring Controllers. *baeldung.com* [online], Sep 2016, [cit. 2018-04-19]. Dostupné z: `http://www.baeldung.com/spring-controllers`

[2] Comprehensive Knowledge Archive Network (CKAN). *ckan.org* [online], [cit. 2018-04-19]. Dostupné z: `https://ckan.org/`

[3] Franklin, C.: How Internet Search Engines Work. *computer.howstuffworks.com* [online], Sep 2000, [cit. 2018-04-19]. Dostupné z: `https://computer.howstuffworks.com/internet/basics/search-engine1.htm`

[4] Jha, A.: Web Crawling: Data Scraping vs. Data Crawling. *promptcloud.com* [online], May 2012, [cit. 2018-04-19]. Dostupné z: `https://www.promptcloud.com/data-scraping-vs-data-crawling/`

[5] Laplante, P. A.: *What every engineer should know about software engineering*. S.l.: CRC Press, 2007, ISBN 978-0849372285.

[6] Leonard, J.: *Systems engineering fundamentals: Supplementary text*. DIANE Publishing, 1999.

[7] Stellman, A.; Greene, J.: *Applied software project management*. Sebastopol, CA: O'Reilly Media, Inc., 2005, ISBN 978-0596009489.

[8] Cockburn, A.: *Writing effective use cases*. Boston: Addison-Wesley, October 2000, ISBN 978-0201702255.

[9] Fowler, M.: *UML distilled: a brief guide to the standard object modeling language*. Boston: Addison-Wesley Professional, třetí vydání, 2004, ISBN 978-0321193681.

[10] Gosling, J.; Joy, B.; Steele, G. L.; aj.: *The Java language specification*. Upper Saddle River, NJ: Addison-Wesley, 2014, ISBN 978-0133900699.

[11] Lindholm, T.; Yellin, F.; Bracha, G.; aj.: *The Java virtual machine specification: Java SE 8 edition*. Upper Saddle River, NJ: Addison-Wesley, 2014, ISBN 978-0133905908.

[12] Fowler, M.: Inversion of Control Containers and the Dependency Injection pattern. *martinfowler.com* [online], 2004, [cit. 2018-04-19]. Dostupné z: `https://www.martinfowler.com/articles/injection.html`

[13] sqdw: Inversion of Control, Dependency Injection. *signaly.cz* [online], Apr 2008, [cit. 2018-04-19]. Dostupné z: `https://sqdw.signaly.cz/0804/inversion-of-control-dependency`

[14] Thymeleaf. *thymeleaf.org* [online], [cit. 2018-04-19]. Dostupné z: `https://www.thymeleaf.org/`

[15] Evans, E.: *Domain-driven design: tackling complexity in the heart of software*. Boston: Addison-Wesley, první vydání, 2004, ISBN 978-0321125217.

[16] Freeman, E.; Robson, E.; Bates, B.; aj.: *Head First Design Patterns: A Brain-Friendly Guide*. Sebastopol, CA: O'Reilly Media, Inc., první vydání, 2004, ISBN 978-0596007126.

[17] What is a Business Object (BO) in IT? - Definition from Techopedia. *techopedia.com* [online], [cit. 2018-04-19]. Dostupné z: `https://www.techopedia.com/definition/25982/business-object-bo`

[18] Bloch, J.: *Effective Java*. Addison-Wesley, druhé vydání, 2008, ISBN 978-0134685991.

[19] 4. The IoC container. *docs.spring.io* [online], [cit. 2018-04-19]. Dostupné z: `https://docs.spring.io/spring/docs/3.1.x/spring-framework-reference/html/beans.html`

[20] Huizinga, D.; Kolawa, A.: *Automated defect prevention: best practices in software management*. Hoboken, NJ: Wiley-Interscience, 2007, ISBN 978-0470042120.

Příloha **A**

Seznam použitých zkratek

SHP Shapefile

GML Geography Markup Language

GeoJSON Geographical JavaScript Object Notation

CSV Comma-separated values

XML Extensible Markup Language

HTML HyperText Markup Language

CKAN Comprehensive Knowledge Archive Network

EET Elektronická evidence tržeb

PDF Portable Document Format

URL Uniform Resource Locator

JSON JavaScript Object Notation

API Application Programming Interface

MFČR Ministerstvo financí České republiky

IČO Identifikační číslo osoby

DPH Daň z přidané hodnoty

MSČR Ministerstvo spravedlnosti České republiky

MKČR Ministerstvo kultury České republiky

DIČ Daňové identifikační číslo

XHTML Extensible HyperText Markup Language

CSS Cascading Style Sheets

UML Unified Modeling Language

IDE Integrated Development Environment

GUI Graphical User Interface

JEE Java Enterprise Edition

JVM Java Virtual Machine

MVC Model–View–Controller

IoC Inversion of Control

HTTP Hypertext Transfer Protocol

AOP Aspect Oriented Programming

JPA Java Persistence API

ORM Object–Relational Mapping

SQL Structured Query Language

JDBC Java Database Connectivity

POM Project Object Model

FIT Fakulta informačních technologií

POJO Plain Old Java Object

JSP Java Servlet Pages

Příloha B

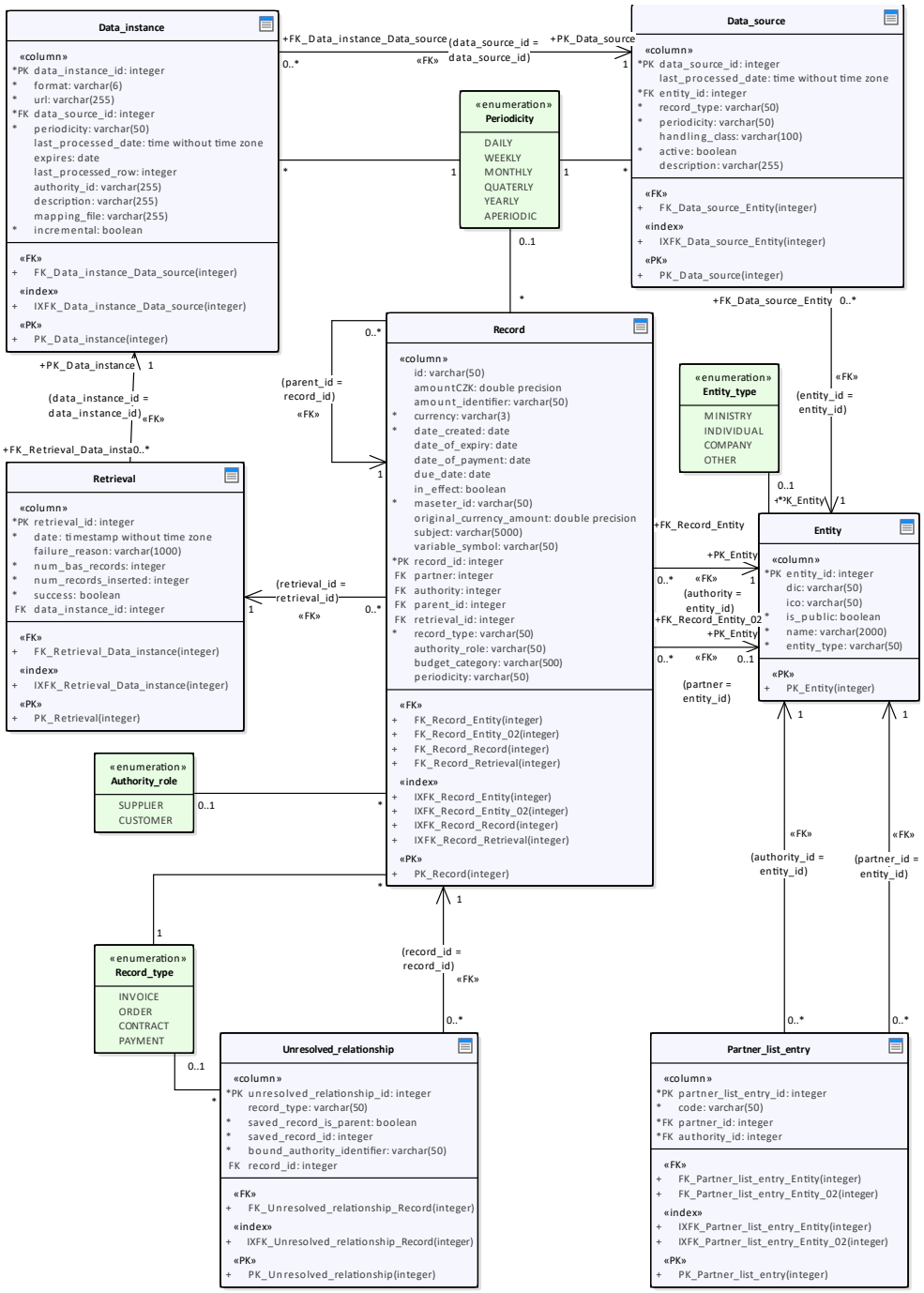
Přílohy

Na obrázku B.1 je vidět databázový model mojí aplikace. Na diagramu jsou znázorněny entity a jejich vazby s příslušnou kardinalitou. Mezi hlavními entitami patří:

- Record,
- Entity,
- Retrieval,
- Data_instance,
- Data_source,
- Unresolved_relationship,
- Partner_list_entry.

Zelenou barvou označeny následující výčtové typy:

- Periodicity,
- Entity_type,
- Authority_role,
- Record_type.



Obrázek B.1: Databázový model

Příloha C

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `exe` ..... adresář se spustitelnou formou implementace
- `Opendata`. ..... adresář projektu se zdrovými kódy
- `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF