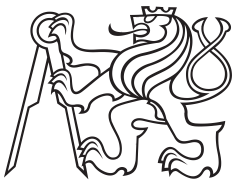
Diplomová práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Manažer sémantických formulářů

Marek Kozlovský

Vedoucí: Mgr. Miroslav Blaško, Ph.D.
Obor: Otevřená informatika
Studijní program: Softwarové inženýrství
Srpen 2021

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Kozlovský** Jméno: **Marek** Osobní číslo: **423308**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Specializace: **Softwarové inženýrství**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Manažer sémantických formulářů

Název diplomové práce anglicky:

Semantic forms manager

Pokyny pro vypracování:

SForms je javascriptová knihovna pro zobrazovaní interaktivních formulářů ve frameworku ReactJS. Na rozdíl od obdobných knihoven je SForm založena na techologiích Sémantického webu, což umožňuje otázky a odpovědi formuláře asociovat s konkretní znalostí uložené v ontologiích. Cílem práce je vytvoření nástroje pro správu a analýzu SForms formulářů včetně integrace s vybraným ticketovacím systémem (např. Trello), který umožní úkoly správy formulářů řídit. Nástroj zprostředkuje komunikaci pro účely návrhu formuláře pro sběr dat i analýzu dat v průběhu sběru dat. Umožní pro různé verze stejného formuláře a jeho uložení základní vizualizace statistik, filtrace pomocí otázek a odpovědí, dotazování na uživatelem definované vzory práce s formulářem a komentování konkrétních otázek formulářů.
Instrukce:
1) seznamte se s technologiemi Sémantického webu pro reprezentaci (OWL, RDF, JSON-LD) and dotazování (SPARQL) znalostí
2) přezkoumejte požadavky pro správu a analýzu SForms formulářů
3) přezkoumente a popište existující nástroje pro správu formulářů a ticketovací systémy vhodné pro integraci
4) definujte uživatelské scénáře pro práci s nástrojem
5) navrhněte a implementujte prototyp aplikace
6) otestujte použitelnost prototypu na definovaných scénářích minimálně na 3 uživatelích
7) porovnejte implementované řešení s existujícími nástroji

Seznam doporučené literatury:

1. Ledvinka M., Blaško M., SForms (https://kbss.felk.cvut.cz/web/kbss/s-forms)
2. Abdalimov, Zakir. Pokročilý Firemní Issue Tracker. BS thesis. České vysoké učení technické v Praze. Vypočetní a informační centrum., 2019.
3. Wood, David. "What's New in RDF 1.1." W3C Working Group Note (2014).
4. Klíma, Tomáš. Sémantický manažer prospektivní klinické studie. BS thesis. České vysoké učení technické v Praze. Vypočetní a informační centrum., 2018.

Jméno a pracoviště vedoucí(ho) diplomové práce:

Mgr. Miroslav Blaško, Ph.D., skupina znalostních softwarových systémů FEL

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

Datum zadání diplomové práce: **15.02.2021** Termín odevzdání diplomové práce: **13.08.2021**

Platnost zadání diplomové práce: **19.02.2023**

Mgr. Miroslav Blaško, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

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

.	
Datum převzetí zadání	Podpis studenta

Poděkování

Poděkování patří především vedoucímu práce, Mgr. Miroslavu Blaškovi, Ph.D., za lidský přístup a podporu, ale také odborné vedení a podnětné přípomínky při zpracování této práce.

Prohlášení

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

V Praze, 13. srpna 2021

Abstrakt

Cílem této práce je navrhnout a implementovat prototyp nástroje pro správu a analýzu formulářů založených na ontologiích. Knihovna SForms, založená na technologiích Sémantického webu, je nástroj pro zobrazování dynamických formulářů, které mohou čítat stovky, až tisíce otázek. Pro takové formuláře, zvlášť při velkém počtu respondentů, je kritické umět analyzovat strukturu formuláře, odpovědi uživatelů a vzory chování při jejich vyplňování.

Nástroj implementovaný v této práci umožňuje připojit se k projektu používajícímu SForms pro sběr dat a importovat formulářová a uživatelská data do interní databáze. Ve shromážděných datech systém sám identifikuje některé zásadní informace o struktuře formulářů, ale především poskytuje možnost tato data analyzovat pomocí dotazovacího jazyka SPARQL. Navíc je do systému integrovaný tiketovací nástroj Trello, pomocí něhož lze spolupracovat nad designem formuláře a nasbíranými daty.

Klíčová slova: SForms, Sémantický web, Sémantický formulář, Trello, Tiketovací systém, RDF, SPARQL, Spring Framework, React, JOPA

Vedoucí: Mgr. Miroslav Blaško, Ph.D.

Abstract

This thesis aims to design and implement a prototype of an analytical tool and a manager for ontology-based smart forms. SForms is a Semantic web-based library for displaying dynamical forms with hundreds to thousands of questions. It is critical be to able to analyze the structure of such forms, their collected data, and patterns of behaviour during their completion, especially for a large number of submissions.

The tool implemented in this thesis enables conneting and importing user and form data from a project that is using SForms for data collection. During the import of data to an internal database, the system identifies some of the key information about the structure of forms and their answers. However, most importantly it allows the users to create analytical SPARQL queries for data examination. Moreover, the Trello ticketing tool is integrated into the system to support collaboration over the design of the forms and their data.

Keywords: SForms, Semantic web, Semantic form, Trello, Ticketing system, RDF, SPARQL, Spring Framework, React, JOPA

Title translation: Semantic forms manager

Obsah

Kapitola 1

Úvod

Digitalizovaná forma shromažďování dat se stala nepostradatelnou součástí moderních institucí a klíčovým prvkem pro zaznamenání jakýchkoliv důležitých informací. Ačkoliv je dnes velmi populární shromažďovat digitalizované informace automaticky, např. na základě nejrůznějších měření a pozorování, stále nejběžnějším způsobem zapisování informací do strojové podoby je sběr dat pomocí digitálních formulářů, nejčastější webových. Realizací webových formulářů existuje nepřeberné množství. Pouze zlomek existujících nástrojů však dokáže nasbíraná data mapovat na systémy znalostní, známé jako ontologie.

Formuláře založené na ontologiích, resp. technologiích Sémantického webu, mají oproti ostatním formulářům nespornou výhodu v tom, že jejich otázky i odpovědi lze mapovat na existující systémy znalostí. Takovým spojením vznikají data, která mají širší kontext a vyšší informační hodnotu.

Jedním z nástrojů, které pracují se sémantickými formuláři, je knihovna SForms[1]. Tato knihovna standardně zobrazuje dynamické formuláře, které čítají stovky až tisíce otázek. U formulářů takového rozsahu je zejména důležité, aby otázky i odpovědi byly zasazeny v širším kontextu informací a snadno se s nimi pracovalo, proto jsou pro ně technologie Sémantického webu vhodné.

Takto rozsáhlé formuláře se používají např. při vedení klinických studií, pro které se z hlediska sběru dat rozlišují tři fáze. Součástí první fáze (tzv. design formuláře) je vytváření formulářové struktury a ověřování funkčnosti jednotlivých dynamických sekcí. Druhou fází je sběr uživatelských dat a přizpůsobování formuláře nárokům zadavatele studie. Poslední fází je zhodnocení a analýza výsledků.

Limitací existujících nástrojů, které s SForms formuláři pracují, je nedosta-

[1] `https://github.com/kbss-cvut/s-forms`

tečná možnost pracovat s historií odpovědí uživatelů a samotnou strukturou formuláře. Kromě toho neexistuje nástroj, pomocí něhož lze efektivně analyzovat formulářová data. Vytváření a ověřování funkčnosti rozsáhlých formulářů je poměrně těžkopádné, protože pro reportování chybových stavů neexistuje žádný příliš intuitivní způsob pro identifikaci problémových částí.

Tato práce si klade za cíl navrhnout a vytvořit nástroj, pomocí něhož bude možné spravovat a analyzovat jak formuláře, tak formulářová data z knihovny SForms. Součástí řešení bude automatická identifikace uživatelských záznamů a formulářové struktury jejich vyplněných formulářů. Pomocí tohoto nástroje bude možné vytvářet analytické dotazy nad omezenou množinou formulářů, a to za účelem nejen rozpoznání uživatelských vzorů chování při vyplňování formuláře. Dále bude v systému integrovaný tiketovací systém, pomocí něhož bude možné spolupracovat při navrhování formulářových změn.

Práce se věnuje rešerši technologií Sémantického webu v kapitole 2. Následně zkoumá tiketovací systémy a existující nástroje pro správu formulářů v kapitole 3. Knihovna SForms a s ní spojené technologie jsou rozebrány v kapitole 4.Návrhu řešení a shromáždění požadavků na implementovanou aplikaci se věnuje kapitola 5. V kapitole 6 je popsána architektura aplikace. Kapitola 7 poskytuje přehled implementovaných funkcí a míru naplnění požadavků. Porovnáním implementovaného nástroje s jedním z již existujících řešení a uživatelským testováním se zabývá kapitola 8.

Kapitola 2

Úvod do tématu

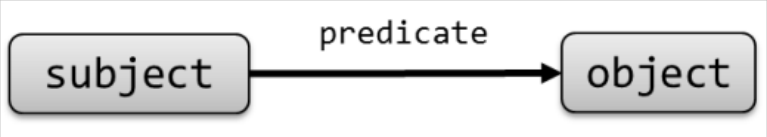
Cílem této kapitoly je poskytnout vhled a určitý technický základ v oblasti sémantického webu a technologií s ním spjatých. Tyto technologie jsou základem jednak pro formuláře SForms, ale také pro implementaci práce samotné. Kromě toho se čtenář seznámí s později používanou terminologií a k dispozici dostane řadu referencí, které je možné použít pro další studium materiálů, relevantních pro tuto práci.

2.1 Sémantický web

Obecně známým systémem pro procházení a vzájemné propojení internetu je WWW (World Wide Web) [1]. Tento systém pomocí předepsaných standardů propojuje dokumenty (webové stránky), které jsou součástí internetu. Soustředí se zejména na propojení dokumentů, které jsou snadno čitelné pro člověka, ale z hlediska automatického zpracování dat pomocí počítačových programů jsou těžko čitelné.

Sémantický web [2] je rozšířením systému WWW, jehož přidanou hodnotou je možnost popisovat a strukturovat data, která se nacházejí v dokumentech na internetu, resp. na webových stránkách. Důsledkem této přidané funkce je, že data obsažená ve zmíněných dokumentech nejsou čitelná pouze pro člověka, ale jsou také strojově zpracovatelná. Toto rozšíření je tedy především řada standardů, které musí jednotlivé dokumenty respektovat, aby mohly být součástí sémantického webu. Data, která vyhovují předepsanému standardu, tak mohou být sdílená a využívaná napříč mnoha informačními systémy.

Stavebním kamenem pro definici vztahů a vlastností dat v rámci Sémantic-



Obrázek 2.1: Grafické znázornění RDF trojice.
Zdroj: https://book.validatingrdf.com/RDFTriple1.png

kého webu jsou zejména technologie RDF (Resource Description Framework) (kapitola 2.2) a OWL (Web Ontology Language) [49]. Pro vytváření dotazů nad daty ve formátu RDF pak existuje jazyk SPARQL (kapitola 2.4).

Je samozřejmě nutné si uvědomit, že technologie sémantického webu neexistují pouze pro sdílení dat napříč webovými stránkami. Tyto technologie v první řadě určují formát, ve kterém lze definovat znalostní systémy, a až v druhé řadě jak s těmito znalostmi pracovat. Použití těchto principů však nikoho nezavazuje k tomu, aby tyto znalostní systémy a informace sdílel.

Velmi častým příkladem užití těchto technologií je specifikování nějaké domény pomocí ontologií. Specifikace této domény je často otevřená a volně dostupná na internetu. Avšak data, která vznikají v tomto formátu, většinou slouží pouze pro interní účely, a tedy mohou, ale nemusí být veřejná.

2.2 RDF

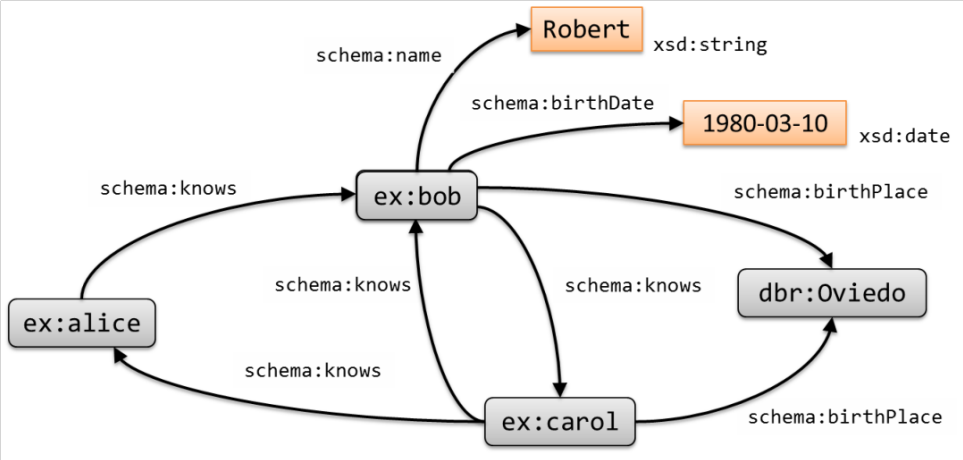
Resource Description Framework (RDF) [3] je jazyk používaný pro popisování zdrojů a modelování informací o nich. Formát RDF je čitelný jak pro člověka, tak pro stroj. Je to abstraktní formát, který definuje pouze způsob, kterým lze informace zapisovat, ale který nedefinuje jednotnou syntaxi těchto informací.

Zdroji, které mohou být popisovány pomocí RDF, mohou být jakékoliv objekty, které jsou jednoznačně identifikovatelné pomocí tzv. URI (anglicky: Uniform Resource Identifier).

Podle standardu RDF se informace o těchto zdrojích zapisují pomocí trojice výrazů (anglicky: *triple*) *subjekt* - *predikát* - *objekt* (ilustraci je možné vidět na obrázku 2.1). Subjekt (podmět) má v tomto případě identifikovat zdroj, o kterém informace vzniká. Predikát zde určuje vlastnost, která je v rámci informace přisuzovaná subjektu. Jinými slovy, predikát určuje druh vazby mezi subjektem a objektem (předmět).

Předmětem, tedy zdroji přisouzenou vlastností, může být buď literál [1], prázdný uzel (blank node) [4] nebo jiný zdroj. Literály se v kontextu RDF používají ke znázornění textových řetězců, datumů a jiných jednoduchých datových typů. Blank nody se používají pro vyjádření existence určitého zdroje bez bližší specifikace jeho vlastností.

[1] https://www.w3.org/TR/rdf11-concepts/section-Graph-Literal



Obrázek 2.2: Příklad reprezentace RDF informací pomocí grafu.
Zdroj: https://book.validatingrdf.com/RDFGraph.png

RDF trojici je také možné představit si jako spojení dvou uzlů pomocí vazby. Není tedy překvapením, že znalosti zapsané v RDF lze reprezentovat pomocí grafu.

Pro ilustraci jednoduchého příkladu viz obrázek 2.2. Obrázek znázorňuje sadu informací vyjádřených pomocí RDF jako orientovaný graf. Z obrázku lze snadno vyčíst vazby mezi jednotlivými zdroji a jejich RDF reprezentace, např.:

- Alice zná Boba: *<ex:alice> <schema:knows> <ex:bob>*
- Bob zná Carol: *<ex:bob> <schema:knows> <ex:carol>*
- Bob má jméno Robert: *<ex:bob> <schema:name> "Robert"*
- Bob byl narozen 10. března 1980: *<ex:bob> <schema:birthDate> "1980-03-10"*

Na obrázku 2.2 jsou mimo jiné vidět znázorněné tzv. prefixy *schema* a *ex*, které zpřehledňují zápis informací tím, že dočasně nahrazují dlouhá tzv. IRI [2] nějakou zkratkou. Tato URL odpovídají ontologiím, ze kterých formát informace vychází.

[2] https://www.w3.org/TR/rdf11-concepts/section-IRIs

2.2.1 RDF formáty

Možností serializace RDF trojic je celá řada, nejznámějšími formáty jsou:

- RDF/XML - XML reprezentace (zdrojový kód 2.1)
- JSON-LD - JSON reprezentace (kapitola 2.5)
- N-Triples - jednoduché trojice, snadno dávkově zpracovatelné
- Turtle - kompaktní, dobře čitelné, podporuje zkrácené zápisy

```
1 @prefix rdf: .
2 @prefix my: .
3 @prefix xsd: .
4
5 my:George my:loves my:Peggy .
6 my:Peggy my:hasHusband my:John .
7 my:John rdf:type ;
8 my:hates my:George ;
9 my:hasAge "27"^^xsd:integer.
```

Zdrojový kód 2.1: Příklad Turtle serializace, včetně zkráceného zápisu

2.3 RDFS

RDF Schema [5] je ontologický jazyk [6], který poskytuje mechanismus pro rozlišení zdrojů - jejich tříd a vlastností. Pomocí tohoto standardu se definují vlastní ontologie a slovníky, díky čemuž je potom možné vytvářet propracované a komplexní specifikace pro libovolnou doménu. Základní třídy v RDFS standardu jsou:

- **rdfs:Resource** – Třída společná pro všechny zdroje v rámci RDF.
- **rdfs:Class** – Třída pro definici samotných zdrojů.
- **rdf:Property** – Třída pro definici vlastností.
- **rdfs:Datatype** – Třída pro definici datových typů.

- **rdfs:Literal** – Třída pro definici literálů.

Zejména důležité vlastnosti v RDFS standardu jsou:

- **rdf:type** – Určuje instance RDF třídy.
- **rdfs:subClassOf** – Indikuje, že třída je podtypem jiné existující třídy.
- **rdfs:label** – Textový, pro člověka srozumitelný popis zdroje.
- **rdfs:comment** – Komentář spojený s třídou.

2.4 SPARQL

SPARQL [7] je SQL [8] jazyk pro dotazování nad RDF (kapitola 2.2) databázemi, který umožňuje efektivně vybírat data z databáze a manipulovat s nimi. Z hlediska struktury dotazů SPARQL do jisté míry respektuje koncept relačních databází a klíčových slov *SELECT, FROM, WHERE*. Avšak identifikace dat v RDF databázi, tedy výběr konkrétních uzlů v RDF grafu, má od relačních databází daleko - připomíná především samotnou deklaraci dat pomocí RDF trojic (obrázek 2.1). Příklad jednoduchého výběru dat je možné vidět ve zdrojovém kódu 4.1.

```
1  PREFIX :
2  SELECT ?sn, (?projname AS ?pn)
3  WHERE {
4      ?e a :Employee .
5      ?e :surname ?sn .
6      ?e :gender 'male'.
7      ?p a :Project .
8      ?p :name ?pn .
9      ?p :administrator ?e .
10     FILTER (strstarts(?sn,'N'))
11 }
```

Zdrojový kód 2.2: Příklad výběru v jazyce SPARQL

Jazyk SPARQL podporuje 4 základní typy dotazů pro výběr dat. Klíčová slova pro uvození těchto dotazů jsou:

- **SELECT** - standardní výběr dat ve formě tabulky pouze s daty, která vyhovují dotazu
- **ASK** - dotaz s výsledkem pravda/nepravda, indikující zda databáze vyhovuje dotazu
- **CONSTRUCT** - standardní výběr dat s výsledkem ve formátu RDF grafu
- **DESCRIBE** - výsledek ve formátu RDF, který popisuje dotázaná data

Kromě známých konstruktů GROUP BY, ORDER BY apod. SPARQL také podporuje celou řadu dalších operací, které jsou specifické pro dotazování nad grafovými databázemi, viz [9].

2.5 JSON-LD

JSON-LD je formát pro serializaci dat, která používá JSON. Oproti základnímu JSON však obsahuje informace o mapování položek ve výsledné struktuře na nějaký RDF model, potažmo ontologii. Mapování modelu je realizováno pomocí tzv. kontextu, který je standardně součástí výsledného JSON, ale také může být k výsledku přidán dodatečně v jiném dokumentu. JSON-LD je mimo jiné formátem pro serializace RDF. Příklad tohoto formátu je možné nahlédnout ve zdrojovém kódu 2.3, kde je zmíněný kontext uvozen klíčovým slovem *@context*.

```
{
  "@context": {
    "firstName": "http://xmlns.com/foaf/0.1/firstName",
    "lastName": "http://xmlns.com/foaf/0.1/lastName",
    "accountName": "http://xmlns.com/foaf/0.1/accountName",
    "isAdmin":
    ↪ "http://krizik/../ontologies/jb4jsonld/isAdmin",
    "role":
    ↪ "http://krizik.felk.cvut.cz/ontologies/jb4jsonld/role"
  },
  "@id": "http://krizik/../jb4jsonld#Catherine+Halsey",
  "@type": [
    "http://onto.fel.cvut.cz/ontologies/ufo/Person",
    "http://krizik.felk.cvut.cz/ontologies/jb4jsonld/User",
    "http://onto.fel.cvut.cz/ontologies/ufo/Agent"
  ],
  "isAdmin": true,
  "accountName": "halsey@unsc.org",
  "firstName": "Catherine",
  "lastName": "Halsey",
  "role": "USER"
}
```

Zdrojový kód 2.3: Příklad výběru dat v jazyce SPARQL

Kapitola 3

Existující řešení

Cílem této práce je implementace nástroje pro správu formulářů SForms. Tento nástroj by měl jako součást uživatelského rozhraní integrovat nějaký tiketovací systém, který přispěje k porozumění odevzdaných formulářům a jejich vyplněným údajům. Tato kapitola poskytuje vhled do řešení a analýzy existujích nástrojů, které se zabývají stejnými nebo podobnými problémy, jaké řeší také tato práce.

Z hlediska tiketovacích systémů je součástí této práce pouze integrace vybraného nástroje (kapitola 3.1). Z tohoto důvodu je součástí rešerše rozlišení několika kategorií, do kterých se mohou řadit tiketovací systémy. Dále je zde rozbor dvou konkrétních nástrojů, které by mohly být vhodné k integraci do nástroje implementovaného v této práci.

Pro implementaci manažeru formulářů je jednoznačně vhodné se inspirovat řešeními, která se tímto problémem již zabývají. Tato kapitola tedy poskytne také vhled do dvou konkrétních nástrojů pro správu běžných webových formulářů. Předmětem zkoumání však není jenom správa formuláře, ale také pohled do možností analýzy nasbíraných dat.

Nelze opomenout, že kromě běžných formulářů je SForms nástroj vhodný také pro sběr dat během klinických studií. Proto je v této kapitole předmětem zkoumání také existující nástroj pro vedení klinických studií (kapitola 3.5.2).

3.1 Tiketovací systémy

Tiketovací systém [10] je označení pro aplikace, které pomocí tzv. tiketů evidují a organizují úkoly při dosahování nějakých komplexních cílů. Tiketovacím systémům se také říká Issue Tracking systémy [34]. Své využití velmi často nacházejí v korporátním prostředí, ale i v menších firmách, kde se dbá na transparentnost a správnou organizaci úkolů a jejich řešení.

Základní funkce, které jsou charakteristické pro všechny tiketovací systémy, jsou:

- vytváření a správa úkolů, reps. tiketů
- přiřazení odpovědnosti za úkoly konkrétním lidem
- řazení úkolů do různých kategorií

Většina těchto systémů však v dnešní době nezůstává u základních funkcí a svá řešení obohacuje o mnoho dalších možností. Mezi obvyklé, avšak pokročilé funkce potom patří:

- vyhledávání tiketů podle pokročilých kritérií
- možnost shromáždění rozsáhlé dokumentace a materiálů, které souvisí s úkoly
- zprostředkování komunikace mezi zainteresovanými osobami (např. pomocí komentářů)
- monitoring a sledování času stráveného úkoly, vytváření metrik
- management jednotné workflow
- integrace s dalšími firemními systémy

Tyto nástroje dnes velmi často nahrazují starší neelektronické způsoby řízení procesů a organizace úkolů. V některých případech dokonce zavádí a pomáhají definovat pravidla v místech, kde předtím žádná nebyla. Je tedy evidentní, že výše zmíněné funkce mohou pro organizace mít při správném použití velmi pozitivní účinky. Některé výhody však nemusí být na první pohled jasné. Shrňme tedy ty obecné benefity, které tiketovací systémy přináší:

- časové zefektivnění plnění úkolů
- elektronická dostupnost systému
- transparentní řízení a organizace procesů v týmu
- vytváření metrik při plnění úkolů a vhled do časových plánů
- zefektivnění dodání výsledků skrz prioritizaci úkolů
- nepřímé snížení nákladů díky zefektivnění procesů

Dalších funkcí a výhod, které tiketovací systémy přináší, je obrovské množství. Neexistuje však nástroj, který by spojoval všechny funkce do jednoho systému. Z tohoto důvodu vznikly kategorie, do kterých se tyto systémy dělí podle zaměření. O těchto kategoriích se lze dočíst v kapitole 3.2.

Pro bližší vhled do principů a realizace tiketovacích systémů je součástí této kapitoly také bližší rozbor nástrojů Trello (kapitola 3.3) a Jira (kapitola 3.4).

3.2 Kategorie tiketovacích systémů

Tiketovacích systémů je v dnešní době k dispozici celá řada, a každý má nějaké vlastní funkce a zaměření. Rozdělení těchto systémů do konkrétních kategorií není vždy jednoznačné. Některé nástroje totiž mohou spadat do více kategorií zároveň, či naopak některé systémy odpovídají charakteristice tiketovacího systému, ale do žádné z těchto kategorií prakticky nespadají. Poměrně rozšířené je následující rozdělení:

- Bug Tracking nástroje pro vývoj (kapitola 3.2.1)
- Help Desk Software (kapitola 3.2.2)
- Service Desk Software (kapitola 3.2.3)
- Project Management Software (kapitola 3.2.4)

Další reference:

- seznam nástrojů Issue Tracking a jejich srovnání [1]

[1] `https://en.wikipedia.org/wiki/Comparison_of_project_management_software`

3.2.1 Bug Tracking nástroje pro vývoj

Jedním z typů tiketovacích nástrojů jsou Bug Tracking nástroje, resp. nástroje pro sledování chyb. Tyto nástroje nejčastěji slouží týmům, které se věnují vývoji nějakých produktů, velmi často softwarových. Vývojové týmy pomocí těchto nástrojů evidují a dokumentují defekty či bugy v produktu, a následně plánují jejich řešení.

Nejčastějším příkladem užití nástroje pro sledování chyb je případ, kdy vývojář nebo tester během vývoje produktu nalezne nějaký nečekaný defekt nebo chybu v aplikaci. Tiketovací systém mu následně poslouží pro zdokumentování celého nálezu. Tento bug, resp. tiket vidí celý tým, a je tedy možné se jím zabývat na současně v celém týmu.

Zároveň ovšem účel těchto nástrojů postupem času přerostl svůj původní význam pouhého sledování chyb. Velmi běžné dnes je, že takovéto nástroje podporují kromě evidence bugů také jiné typy úkolů a dále pomáhají organizovat týmovou práci - nebo jsou dokonce používané jako plnohodnotné nástroje pro projektové řízení. V mnoha případech je zde tedy překryv funkcí s nástroji z kategorie Project Management Software (kapitola 3.2.4).

Reprezentativními příklady systémů, které se zaměřují pouze na sledování chyb bez podpory projektového řízení jsou:

- Bugzilla [2]
- Mantis Bug Tracker [3]

Nástrojem, který navýšil svou použitelnost nad pouhé sledování bugů a do jisté míry pomáhá také s organizací úkolů, je Jira Work Management. Tento nástroj má daleko do toho, aby se dal označovat jako nástroj pro Project Management. Na druhou stranu už funkce, které poskytuje, jsou poměrně užitečné a pro základní organizaci projektu stačí. Tento nástroj je předmětem bližšího zkoumání v kapitole 3.4.

3.2.2 Help Desk Software

Help Desk je základní technická podpora pro zákazníky nebo koncové uživatele nějakého produktu. Právě koncept tiketovacího systému je velmi často použitý pro realizaci takovéto služby. Obecně vzato je cílem Helpdesku nejčastěji

[2] https://cs.wikipedia.org/wiki/Bugzilla
[3] https://cs.wikipedia.org/wiki/MantisBT

okamžité odstraňování problémů zákazníků nebo poskytování informací o produktech, jako jsou počítače a další elektronická zařízení nebo software.

Cílem helpdesku je rychle a efektivně řešit problémy a incidenty, které zákazníkům mohou vyvstat při používání produktu. Help Desk může být buď součástí nějaké Service desk služby (kapitola 3.2.3), která poskytuje širší spektrum pomoci s obsluhou nějakého produktu, nebo může být oddělený.

Známými příklady služeb tohoto typu jsou:

- Salesforce [4]
- Zendesk [5]

3.2.3 Service Desk

Service Desk je druh technické podpory, resp. jde o kontaktní místo pro vyřizování incidentů, které souvisí s nějakým provozovaným produktem, ale jde také o kontakt pro vyřizování běžných požadavků uživatelů. Service Desk tak typicky uspokojuje potřeby jednak, ale také např. vlastních zaměstnanců v rámci firmy. Help Desk služby jsou podmnožinou služeb Service Desk, a je běžné že jsou také realizovány pomocí konceptu tiketovacích systémů.

Mezi známé a populární nástroje Service Desk patří:

- ServiceNow [6]
- Cherwell [7]
- EasyVista [8]

[4] https://www.salesforce.com/
[5] https://www.zendesk.com/
[6] https://www.servicenow.com/
[7] https://www.cherwell.com/
[8] https://www.easyvista.com/

3.2.4 Project Management software

Obecně vzato jsou Project Management nástroje takové nástroje, které v rámci projektového řízení [11] umožňují plánovat, organizovat a celkově spravovat týmovou práci. Tiketovací nástroje jsou pak jedním z možných způsobů, jak takové nástroje realizovat - a nutno říci že to je poměrně populární volba.

Uplatnění project managementu lze najít v široké řadě odvětví. Tyto nástroje většinou podporují nějakou ze známých metodik pro projektové řízení. Každá metodika se opírá o nějaké paradigma a od ostatních se liší přístupem, který aplikuje při řešení dané problematiky - resp., poskytuje jiný způsob a pohled na danou doménu. Uplatnit tyto metodiky je vhodné v jakýchkoliv situacích, které vyžadují netriviální řízení lidí a jejich úkolů.

Některé styly řízení mohou být vhodné pro vývoj softwarových produktů, kde je priorita jednotlivých úkolů většinou jednoznačně určena. Jiné styly řízení zase mohou být vhodné pro kontinuální řízení výroby produktů v nějaké továrně. Známá paradigmata pro agilní řízení projektů [12] jsou např. Kanban [13] a Scrum [14]. Naproti tomu stojí naprosto odlišný vývojový proces - model waterfall [15].

Kategorie nástrojů pro project management se částečně liší od ostatních kategorií tiketovacích systémů. Liší se především tím, že systémy tohoto typu lze spojit s některou z dalších ze zmíněných kategorií. Např. nástroje pro sledování chyb je velmi vhodné obohatit o podporu paradigmat projektového řízení. Tímto spojením vzniká multifunkční nástroj, ve kterém lze řídit a sledovat vývoj rozsáhlých projektů agilně. Výsledný nástroj si lze představit jako bug tracker (kapitola 3.2.1), jehož úkoly však mohou nabývat dalších typů (nejenom bugů) a hierarchií, které lze řídit pomocí zmíněných agilních metodik.

Tiketovací systémy zaměřené především na projektové řízení jsou např.:

- Trello [9] - dále popsáno v kapitole 3.3
- Basecamp [10]

Mezi nástroje, které spojují nástroje pro sledování chyb a koncepty projektového řízení, patří:

- Jira Software [11] - dále popsáno v kapitole 3.4

[9] `https://cs.wikipedia.org/wiki/Trello`
[10] `https://basecamp.com/how-it-works`
[11] `https://cs.wikipedia.org/wiki/JIRA`

- Redmine [12]

Další reference:

- seznam nástrojů pro projektové řízení a jejich srovnání [13]
- rozdíl mezi Jira Work Management a Jira Software [14]

3.3 Trello

Trello [16] je webová aplikace, jež umožňuje řídit projekty pomocí paradigmatu Kanban [13]. Pro projekty řízené pomocí tohoto paradigmatu existují v aplikaci tzv. nástěnky. Každá nástěnka je rozdělena na několik sloupců, ve kterých se mohou vyskytovat karty (úkoly). Stav, ve kterém se úkoly nacházejí indikuje právě sloupec, ve kterém jsou umístěny. Sloupce je možné vytvořit a pojmenovat podle vlastní potřeby, díky čemuž je možné snadno navrhovat vlastní systém stavů, kterých mohou úkoly nabývat. Mezi základní, velmi často používané stavy patří:

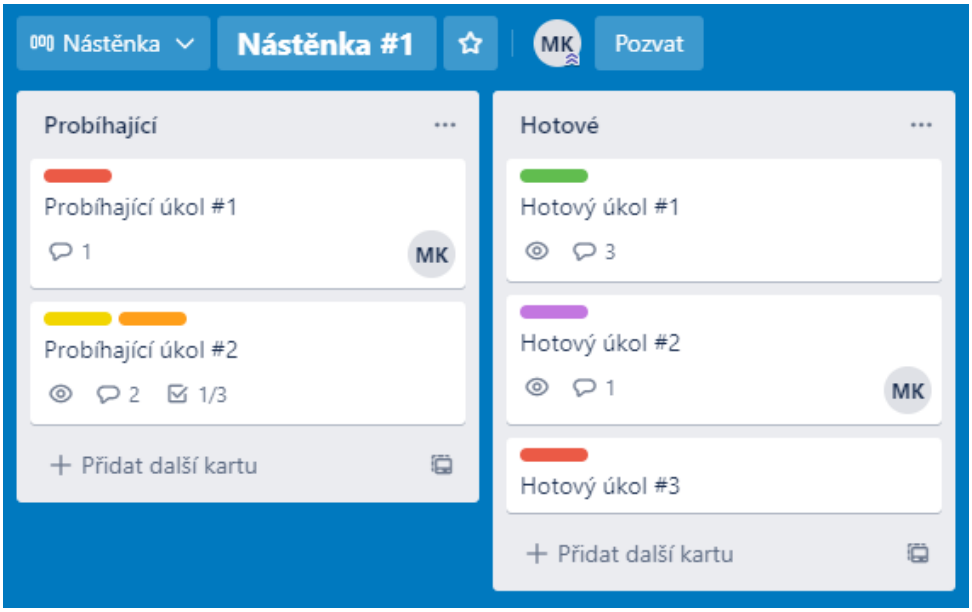
- **ToDo** - naplánované úkoly
- **In Progress** - právě probíhající úkoly
- **Done** - hotové úkoly

Pro zaznamenání změny stavu úkolu je tedy nutné pouze přesunout kartu do jiného sloupečku pomocí tzv. drag and drop. Obrázek 3.1 poskytuje náhled do projektu s 5 vytvořenými kartami a 2 možnými stavy (sloupci).

Pro vytvoření detailní specifikace úkolu existuje v Trello celá řada možností. Základními vlastnostmi úkolu jsou zde název, slovní charakteristika, termín dodání, osoba odpovědná za úkol, přiložené dokumenty a tzv. barevné štítky (indikující další interní kategorie). Kromě jiného je také možné jako součást karty uvést seznam podúkolů, diskutovat s ostatními účastníky projektu (přímo ve vlákně karty), či nahlížet do historie aktivit spojených s úkolem.

[12] https://cs.wikipedia.org/wiki/Redmine
[13] https://en.wikipedia.org/wiki/Comparison_of_project_management_software
[14] https://thejiraguy.com/2021/03/10/so-what-even-is-jira-work-management/

Uživatelské rozhraní pro správu a editaci karty je možné vidět na obrázku 3.1.



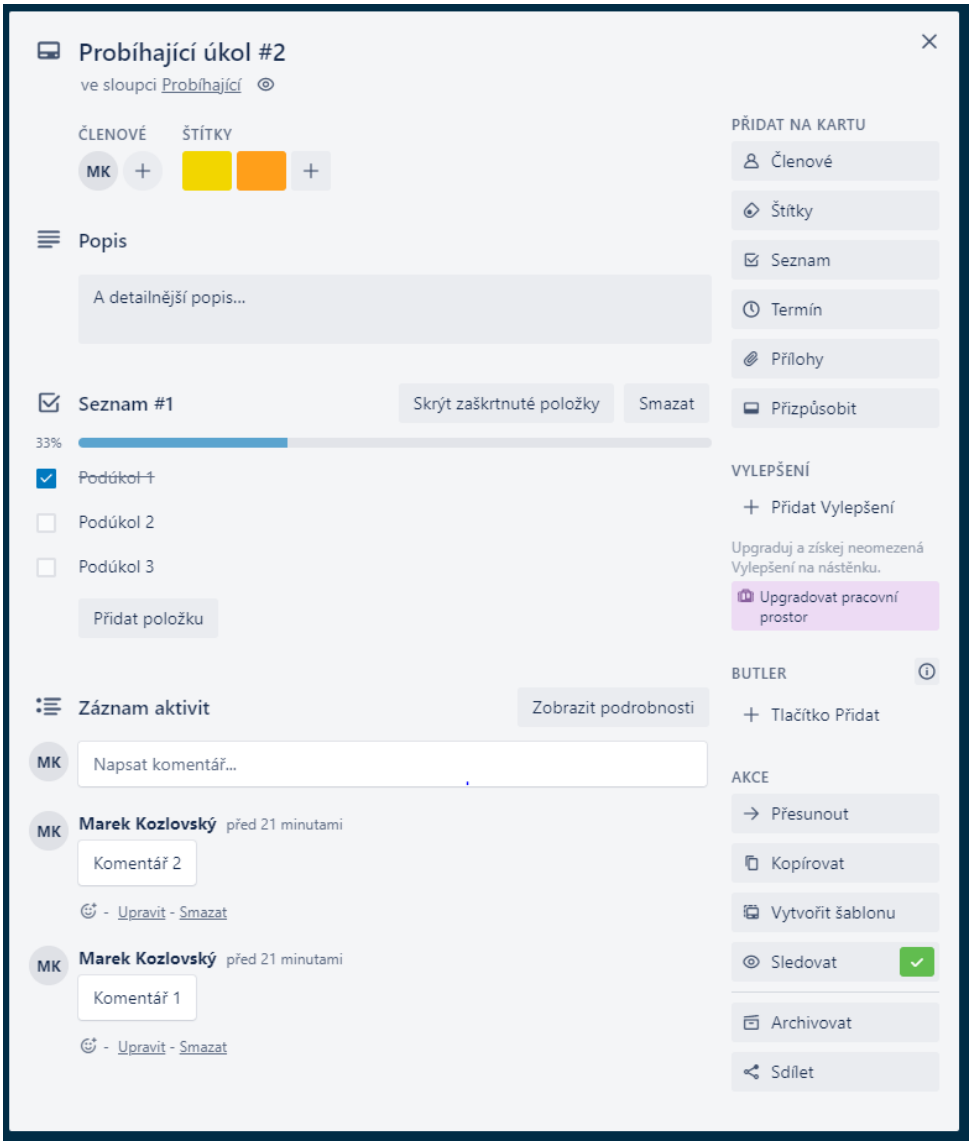
Obrázek 3.1: Přehled projektu v Trello.

Trello se v projektovém řízení těší vysoké popularitě, a to zejména díky jednoduchosti a intuitivnosti celé aplikace. Založení projektu a spravování úkolů je velmi snadné a přehledné. Vysoký potenciál a využití však nemá pouze díky jednoduchosti základních use-casů, nýbrž proto, že pro expertního uživatele představuje velmi silný nástroj. Silným nástrojem může být např. i díky celé řadě pluginů, které je možné do projektů přidávat.

Služba nabízí 3 cenové balíčky pro používání aplikace [15]. Základní balíček je bezplatnou variantou, která nabízí všechny výše zmíněné utility, jakými je vytváření nástěnek, úkolů apod. Součástí balíčku *business class* je oproti bezplatné verzi např. neomezený počet nástěnek, pokročilé vytváření seznamů, přiložení map a neomezený počet nainstalovaných pluginů. *Edice enterprise* poskytuje všechny možnosti jako *business class*, ale kromě toho přidává možnost spravovat nejen projekty a nástěnky, nýbrž celou organizaci.

Lze namítat, že Trello je možné od roku 2016 používat také jako již zmíněný Bug Tracking nástroj (kapitola 3.2.1), tedy ne pouze pro řízení projektů. Do veřejného povědomí však Trello vstoupilo hlavně díky možnosti intuitivně řídit projekty s paradigmatem Kanban, a takto vnímané i částečně zůstalo.

[15] `https://trello.com/cs/pricing`



Obrázek 3.2: Uživatelské rozhraní pro správu úkolů v Trello.

Trello Rest API

Pro správu projektu v aplikaci Trello je kromě standardního webového rozhraní možné použít také programové rozhraní REST API [17]. Trello REST API [50] umožňuje řídit projekt pomocí prakticky všech funkcí, které jsou dostupné ve webovém klientovi. Ke komunikaci je nutné používat identifikační klíč a ověřovací token. Oba tyto údaje je možné vygenerovat ve webovém rozhraní a musí být součástí každého požadavku směřujícího do programového rozhraní.

Příklad vytvoření požadavku na Trello REST API v jazyce Java [18] je možné vidět ve zdrojovém kódu 3.1. Součástí obrázku je požadavek na adresu začínající klíčovým slovem api (identifikace REST API) a končící klíčovým slovem cards (indikující požadavek na správu karet/úkolů v projektu). Dále je možné vidět použité parametry *key*, *token* a *idList*, kde *key* je již zmíněný identifikační klíč, *token* je ověřovací údaj a *idList* je identifikátor listu, ve kterém se má karta vytvořit. Důležitým prvkem je také zvolená HTTP metoda [19] požadavku - v tomto případě post, která v konvencích REST API indikuje vytvoření nového objektu.

Stejným způsobem je možné pracovat s dalšími prvky nástěnky v Trello, k čemuž je dostupná rozsáhlá oficiální dokumentace.

```
1  // Tento kód používá knihovnu 'Unirest':
2  // http://unirest.io/java.html
3  HttpResponse<String> response =
4      Unirest.post("https://api.trello.com/1/cards")
5    .queryString("key", "0471642aefef5fa1fa76530ce1ba4c85")
6    .queryString("token", "9eb76d9a9d02b8dd40c2f3e5df18556c8")
7    .queryString("idList", "5abbe4b7ddc1b351ef961414")
8    .asString();
9
10 System.out.println(response.getBody());
```

Zdrojový kód 3.1: Příklad vytvoření požadavku do Trello REST API v jazyce Java

3.4 Jira

Pojem Jira [20] u většiny lidí z oboru asociuje tiketovací nástroj, který je používaný především v korporátním prostředí pro řízení projektů a vývoj. Nástroj Jira je produktem firmy Atlassian [16], zaměřující se na vývoj aplikací vhodných pro vývojáře software, projektové manažery a obecně se zabývající podpůrnými systémy pro práci vývojových týmů.

Produkt Jira sám o sobě nabízí tři základní konfigurace, resp. balíčky, ve kterých ho lze použít. Těmito balíčky jsou Jira Work Management, Jira Software a Jira Service Management. Pro tuto diplomovou práci je však relevantní pouze první z uvedených nástrojů. Je však důležité říci, že Jira WM poskytuje jakýsi základ pro práci s úkoly a projekty, na němž dále staví oba dva další produkty. Jira Software a Service Management jsou tedy nástavbou

[16] https://www.atlassian.com/

Jira Work Management a pokud používáme pojem Jira přímo bez dalších přívlastků, vztahuje se popisovaná charakteristika ke všem těmto balíčkům. Jednotlivým zaměřením se krátce věnuje kapitole 3.4.2.

Jira Work Management spadá svou charakteristikou především do kategorie Bug Tracking nástrojů pro vývoj z kapitoly 3.2.1. Je to tiketovací systém, jehož základními entitami pro správu úkolů jsou *projekt* a *issue*. Jira obecně disponuje velkým množstvím konfigurovatelných prvků - jak v ohledu uživatelského rozhraní, tak z pohledu správy úkolů a projektu.

Pojem issue v kontextu systému Jira představuje úkol. Úkol vždy spadá do nějakého projektu, kterých může být v systému definováno libovolné množství. Jira projekty oddělují úkoly podle cílů, jimiž může být jakákoliv zvolená činnost, kterou lze nějak popsat. Jira projekt typicky definuje a zaznamenává vývoj nějakého produktu, přičemž úkoly jsou zde dílčí kroky, které vedou k dokončení tohoto produktu.

Důležitou součástí tohoto nástroje jsou tzv. boardy, které zobrazují aktuální stav projektu alespoň jeho části. Je to náhled na množinu úkolů, které jsou aktuálně přiřazeny danému boardu, jehož rozložení určuje nějaké aktivní workflow. Koncept workflow určuje jednak množinu stavů, kterých mohou úkoly nabývat, a jednak také jejich uspořádání neboli pořadí, ve kterém by měly v ideálním případě úkoly do stavů přicházet. Jira nabízí řadu předdefinovaných workflow, z nichž lze vybírat. Snadnou možností je však také vytvořit workflow vlastní. Základní workflow, které Jira nabízí se skládá ze tří stavů, jež jsou stejné jako u nástroje Trello - *ToDo*, *In Progress*, *Done*. Zajímavějším je však např. už netriviální workflow se stavy:

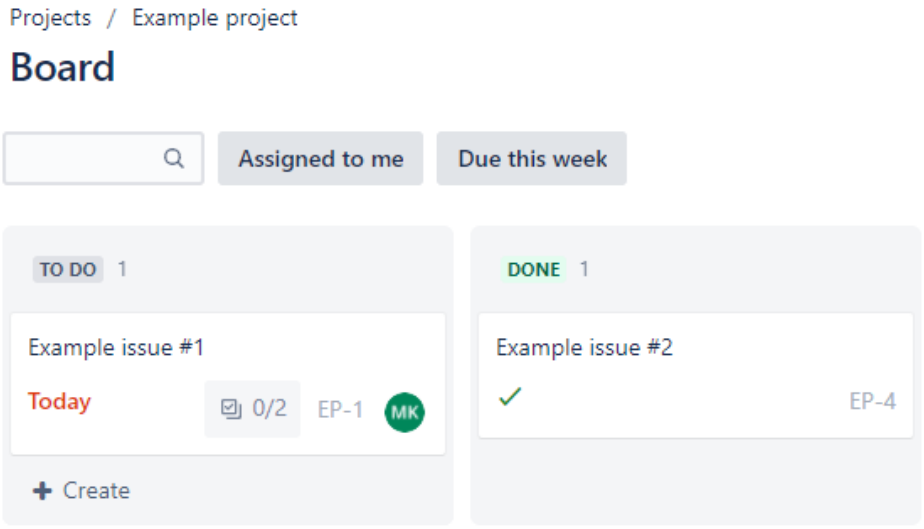
- **Open** - úkoly pro budoucí zpracování
- **In progress** - právě probíhající úkoly
- **Resolved** - vyřešené úkoly
- **Reopened** - znovu otevřené úkoly
- **Closed** - uzavřené úkoly

Stejně jako nástěnky v nástroji Trello i zde je board rozdělen na sloupečky, ve kterých se mohou nacházet úkoly. Tyto sloupečky jsou seřazené tak, aby odpovídaly zvolenému workflow. Pořadí stavů ve workflow lze snadno ilustrovat stavovým diagramem, se kterým je možné pracovat přímo v aplikaci. Na obrázku 3.3 je vidět diagram s dvěma definovanými stavy, který popisuje jména stavů a jednu z přechodovým funkcí mezi těmito stavy.

Pro podrobnou definici úkolů Jira nabízí celou řadu funkcí, mezi něž samozřejmě patří i zmíněný stav issue. Jira v základní konfiguraci rozlišuje tři typy issue:

- **Task** - menší, naprosto běžný úkol
- **Subtask** - menší úkol, který je součástí většího úkolu
- **Epic** - velký úkol, určený k dalšímu rozdělení na dílčí části

Obrázek 3.3: Stavový diagram reprezentující workflow o dvou možných stavech.

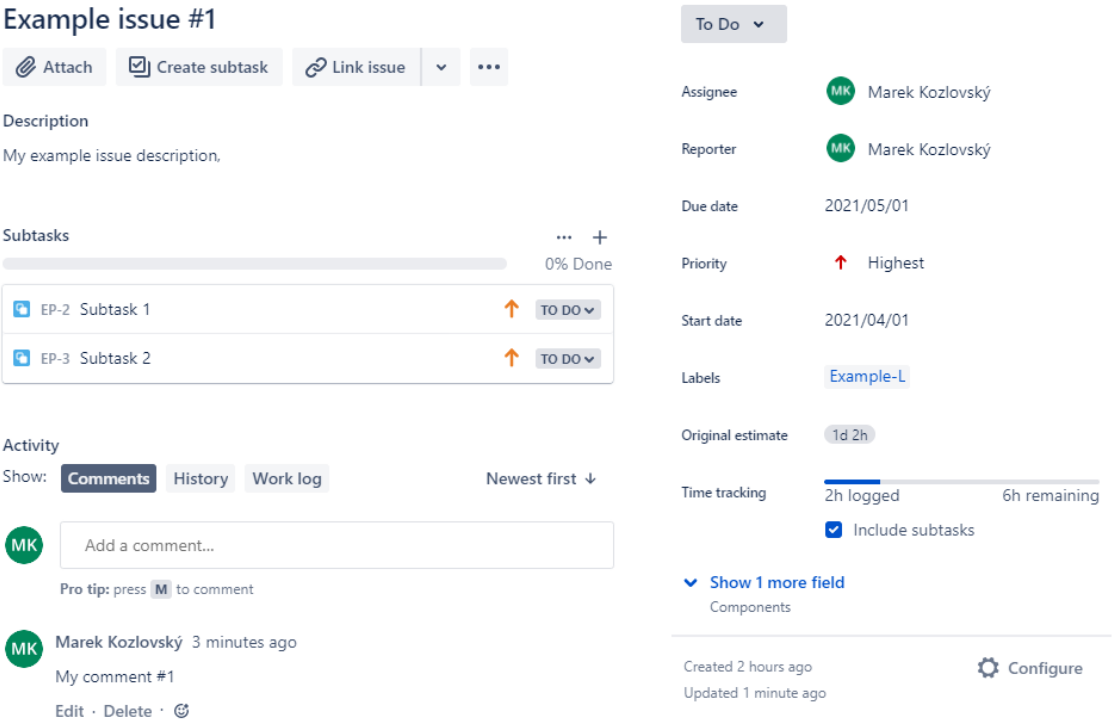


Obrázek 3.4: Náhled Jira boardu s dvěma definovanými stavy, dvěma issue a ovládacími prvky pro filtraci issue na boardu.

Typické položky, pomocí nichž lze úkol specifikovat:

- jméno issue a popis
- priorita, časový odhad a termín odevzdání
- odpovědná osoba a další zainteresované osoby
- podúkoly
- přílohy

Kromě těchto funkcí je možné dále sledovat čas strávený úkolem, diskutovat přímo ve vlákně úkolu či procházet historii jeho úprav. Detail issue je možné vidět na obrázku 3.5.



Obrázek 3.5: Náhled detailu Jira issue.

Za zvláštní zmínku stojí tzv. dashboardy, které uživateli prezentují statistiky a další užitečné informace o probíhajících projektech a jejich issues. Typickým příkladem použití je např. graf zobrazující poměr otevřených issue k probíhajícím a dokončeným. Dalším přikladem může být přehled úkolů přiřazených jednotlivým lidem v konkretním týmu nebo diagram, zobrazující počet vznikajících issue v čase.

3.4.1 Konfigurace Jira

Jira je dobře známým a populárním nástrojem zejména u velkých firem, pro které je důležité řídit a sledovat práci na úrovni celé organizace. V Jira si mnoho firem našlo zalíbení hlavně díky opravdu široké řadě možností konfiguracovat celý nástroj k vlastní potřebě. Jednu stranu pohledu na konfiguraci představuje oblast administrace a ta druhá představuje možnost přidávat do nástroje různá rozšíření.

Administrátorská sekce umožňuje měnit uživatelské rozhraní, a tedy pro všechny uživatele přizpůsobovat např. hlavní stranu aplikace, dashboardy apod. Také ale poskytuje pokročilejší možnosti správy uživatelů, např. z hlediska práv atd.

Rozšíření, které je možné přidávat do nástroje, je celá řada. Jsou to buď oficiální rozšíření přímo od Atlassian, nebo pluginy pro integraci ostatních produktů Atlassian, a nebo rozšíření třetích stran. Všechna rozšíření lze procházet na Atlassian Marketplace [17].

3.4.2 Rozdělení balíčků Jira

Jak je již zmíněno výše, pod pojmem Jira si většina lidí představí nástroj pro správu projektů a úkolů. Samotné označení Jira však už dnes neoznačuje přímo jeden produkt, ale spíš spojuje několik produktů, které fungují na stejné bázi, avšak zaměření mají jiné. Těmito produkty jsou:

- Jira Work Management [18]
  - základní balíček pro práci na projektu
  - obsahuje všechny funkce výše zmíněné v této kapitole
  - zdarma až pro 10 uživatelů
- Jira Software [19]
  - balíček pro pokročilé řízení projektů
  - obsahuje všechny funkce Jira Work Management
  - navíc podporuje také agilní metodiky Scrum [14] a Kanban [13], plánování projektů pomocí konceptu Roadmap [21] a další

[17] `https://marketplace.atlassian.com/`
[18] `https://www.atlassian.com/software/jira/work-management`
[19] `https://www.atlassian.com/software/jira`

- Jira Service Management [20]
  - balíček pro pokročilou podporu IT služeb
  - obsahuje všechny funkce Jira Work Management
  - navíc obsahuje rozhraní pro zadávání požadavků klienty a automatizované procesy pro správu těchto požadavků a další možnosti

3.4.3 Jira REST API

Jak je tomu i u jiných služeb tohoto typu, Jira vystavuje komunikační rozhraní podle standardu REST API [17], pomocí něhož je také možné pracovat s tímto nástrojem - Jira REST API [21].

S Jira REST API je možné vykonávat základní funkce, jako je např. vytváření, mazání a editace issues a projektů. Dále je možné se dotazovat na množiny issues, které vyhovují i složitějším kritériím.

Pro komunikaci s nástrojem je nutné využít jednu ze dvou možností autentifikace, kterými jsou:

- **OAuth 1.0a** [52] - metoda delegování přístupu pomocí jiné služby
- **Basic authentication** [51] - metoda pomocí dvojice ověřovacích informací (např. jméno a heslo)

Vzorová URI pro běžnou komunikaci, např. pro práci s tiketem vypadá takto:

- $http://host:port/context/rest/api/projectId/project$

Vzorová URI pro účely ověření vypadá takto:

- $http://host:port/context/rest/auth/latest/session$

[20] `https://www.atlassian.com/software/jira/service-management`
[21] `https://developer.atlassian.com/server/jira/platform/rest-apis/`

3.4.4 Jira Java API

Není tak časté, že by pro webovou aplikaci existovala oficiální knihovna, kterou lze použít pro komunikaci s běžícími instancemi služby. Jira však takovou možnost nabízí, a to konkrétně pro jazyk Java.

Jira Java API [22] je oficální podporovanou knihovnou pro programovou komunikaci Java aplikace a nástroje Jira. Tato knihovna nabízí velmi podobné možnosti jako Jira REST API, přičemž její výhodou je, že zajišťuje vytváření i složitých požadavků, jejichž podoba je ve formě URI poměrně nepřehledná.

3.5 Nástroje pro správu formulářů

Cílem této práce je vytvořit nástroj, pomocí něhož bude možné spravovat formuláře SForms. Takový nástroj musí umět rozlišovat jednotlivé formuláře, jejich vyplnění, a případně jejich verze. Tato kapitola se věnuje analýze existujících nástrojů, které se tímto problémem zabývají. Obsahem kapitoly je tedy představení vybraných nástrojů, které jsou za tímto účelem používány širokou veřejností a jejichž design může být vzorem i pro správce formulářů.

Formuláře SForms (kapitola 4) se dají použít jako běžné formuláře pro sběr dat v každodenní činnosti a jsou vhodné pro integraci do jakékoliv aplikace, která používá React [22]. Kromě použití SForms jako běžných formulářů pro každodenní činnosti jsou však SForms využívané také v poměrně specifické oblasti sběru dat, jakou jsou klinické studie.

Nástroj, jehož implementací se tato práce zabývá, by tedy měl přihlížet k designu široce používaných nástrojů pro správu běžných formulářů, ovšem také respektovat podobu formulářů pro klinické studie - které např. mohou obsahovat i stovky otázek. Jako vhodný vzorek nástrojů pro běžné formuláře poslouží nástroje Google Forms (kapitola 3.5.1) a JotForm (kapitola 3.5.1). Pro vhled do klinických studií se rozebere nástroj OpenClinica (kapitola 3.5.2).

3.5.1 Standardní nástroje pro správu formulářů

Tato kapitola představí nástroje Google Forms a JotForm. Oba tyto nástroje je možné použít nejen k procházení a správě vyplněných formulářů, nýbrž také k jejich designování. Tato kapitola jevšak prozkoumá pouze ty části

[22] `https://developer.atlassian.com/server/jira/platform/rest-apis/`

nástrojů, které se věnují správě formulářů. Analýza částí, které se věnují designu formulářů, není součástí tohoto textu. Google Forms i JotForm jsou nástroje volně dostupné k osobnímu použití a jejich rozbor se opírá i o osobní zkušenost autora této práce.

Google Forms

Formuláře Google [23] neboli Google Forms je nástrojem pro vytváření webových formulářů. Google Forms jsou jedním ze známých nástrojů Google, které může používat kterýkoliv uživatel služby Gmail. Po této službě sáhne velmi mnoho lidí právě proto, že se k ní každý dostane přímo ze svého emailového účtu. Jako je zvykem u většiny služeb Google, i Google Forms jsou velmi snadno použitelné a lidé tedy většinou nemají důvod hledat další nástroje obdobného typu.

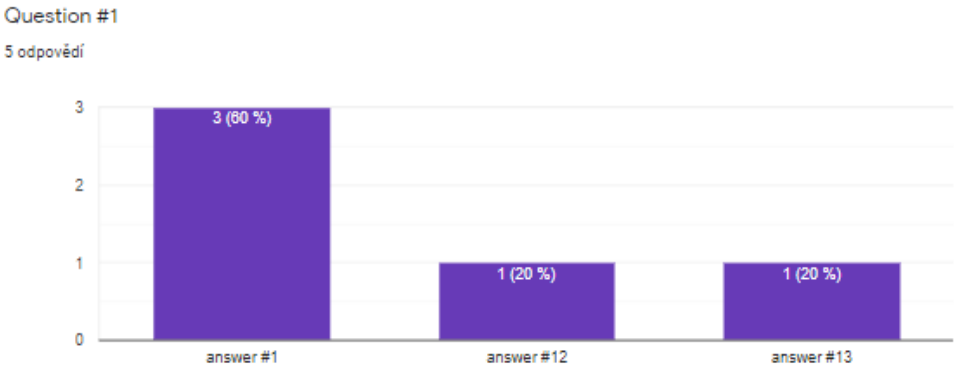
Uživatelské rozhraní tohoto nástroje má sekci pro autora formuláře, kde je možné upravovat formulář a procházet jeho odevzdání. Další sekcí je samotný formulář, který lze vyplnit a odevzdat.

V administrátorské sekci pro design formulářů služba nabízí 10 typů vstupních polí a 17 základních šablon. Mezi otázky je možné vkládat obrázky a videa.

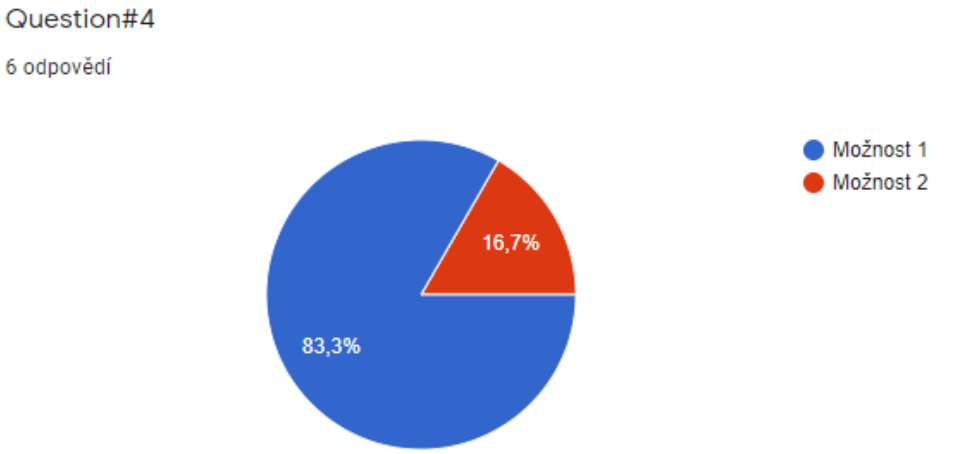
Autor formuláře může procházet odpovědi respondentů ve třech možných variantách:

- **Souhrn** - souhrnný přehled odpovědí, jehož součástí jsou nejčastěji grafy zobrazující výskyt jednotlivých odpovědí
- **Otázky** - zobrazení vždy jedné otázky a všech jejích odevzdaných odpovědí
- **Individuální** - procházení jednotlivě odevzdaných formulářů - tedy zobrazení všech odpovědí jednoho uživatele

Nejzajímavější variantou zobrazení odpovědí je *souhrn*. Souhrnné zobrazení používá řadu grafů pro otázky různých typů. Pro textová pole používá sloupcový graf [3.6] nebo tabulku s odpověďmi. Pro políčka, která mají definovaný rámec odpovědí, používá koláčové grafy [3.7]. Tyto grafy je možné snadno přenést jako obrázky do ostatních nástrojů Google (Dokumenty, Tabulky atd.) a v některých případech i do dalších služeb.



Obrázek 3.6: Sloupcový graf znázorňující výskyt odpovědí pro textové pole (Google Forms).



Obrázek 3.7: Koláčový graf, který znázorňuje výskyt odpovědí pro otázku s definovaým rámcem možných odpovědí (Google Forms).

Formuláře v této aplikaci je možné upravovat, avšak bohužel zde není možné sledovat historii jeho úprav. Dále aplikace neudržuje odpovědi otázek, které byly jednou zodpovězeny, ale jejich otázka byla později z formuláře odstraněna. Pro nově vytvořené otázky je v již vyplněných formulářích prázdná odpověď. U nových otázek tedy není jasné, zda byly součástí formuláře, ale nebyly vyplněny, nebo zda byly otázky přidány až po vyplnění formuláře. Podpora různých verzí formuláře je zde tedy nedostatečná.

Služba Google Forms je bezplatná a její formuláře lze snadno sdílet, a to i na úrovni přístupu do administrátorské sekce. Je dobře integrovaná s účtem Google. Odpovědi respondentů lze snadno exportovat do do formátu csv. Důležité však je, že je lze propojit také s Tabulkami Google, kam

se automaticky doplňují další odpovědi. Následně je tedy možné odpovědi zpracovat jako tabulku a lze tak využít všech možnosti tabulkového editoru včetně jeho funkcí apod.

Celkově je služba velmi intuitivní a snadná k použití. Pro technické uživale nabízí také editaci skriptů, využitelnou také k doprogramování vlastních funkcí. Je možné službu propojit i s množstvím doplňků třetích stran. Na druhou na stranu je zjevné, že tato služba je designovaná především pro jednoduché use-casy a spíš krátkodobé použití. Oproti ostatním nástrojům nedisponuje tak velkým množstvím typů vstupních polí a přehledy jsou zde poměrně stručné. Neeviduje historii úprav formuláře. Pro náročnější případy užití a dlouhodobou práci je tedy hůř použitelná než její konkurenti.

JotForm

JotForm [23] je webová aplikace pro vytváření webových formulářů, jejich sdílení a následné shromažďování vyplněných odpovědí. Je to tedy program, kde je možné jednak formuláře designovat, ale také spravovat a analyzovat jejich odpovědi. Mezi význačné aplikace svého typu se řadí proto, že byla jedním z prvních nástrojů, který tyto služby nabízel. Funguje od roku 2006 a počet jejích uživatelů se za tu dobu vyšplhal až na 10 milionů. Za svoju popularitu vděčí především své jednoduchosti, které dosahuje přesto, že je zároveň poměrně mocným nástrojem.

Klíčové funkce aplikace jsou vytváření formulářů, jejich sdílení, systém notifikací při jejich vyplnění a samozřejmě správa odpovědí a možnost analyzovat výsledky.

Webové rozhraní pro tvorbu formulářů disponuje desítkami typů vstupních polí a validačních pravidel, jako jsou např. textová pole, pole pro datum, čísla apod. Uživatel má k dispozici až 10000 předpřipravených šablon. Pouze část z nich je však zadarmo. Sdílet formuláře lze pomocí odkazu nebo je lze vložit na vlastní stránky jako HTML. Pro správu odpovědí existuje v aplikaci samostatná sekce, kde je ve formě tabulky možné prohlížet odevzdané odpovědi. Tyto odpovědi lze chytře filtrovat:

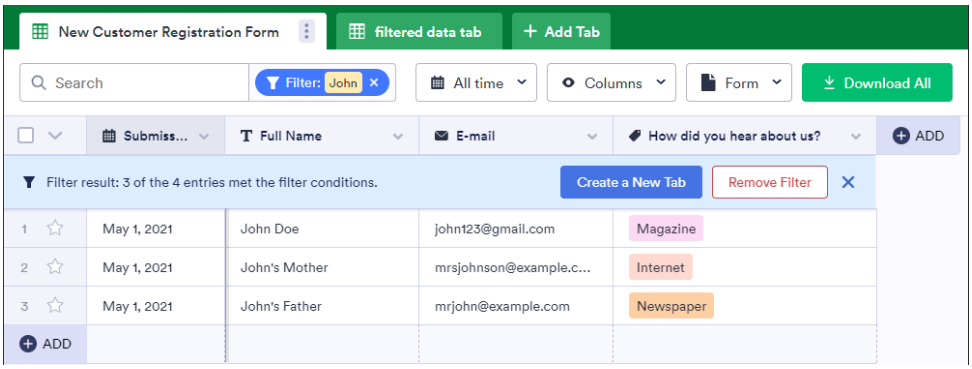
- podle zobrazené, nezobrazené, oblíbené a neoblíbené
- podle hodnot odpovědí
  - pro textová podle a např. také pro IP adresy podle operace *obsahuje*, *rovná se*, *nerovná se*
  - pro čísla a datumy podle operace *menší*, *větší* apod.

[23] https://www.jotform.com/

Filtračních kritérií na hodnoty pole je možné použít libovolné množství zároveň. Neexistuje zde však operátor *NEBO*, nýbrž pouze operátor *A*. Použitý filtr je možné použít k vytvoření nové záložky, kde je vybraný filtr předem nastavený. Data na této nové záložce potom odpovídají údajům na té původní, a to i ve chvíli, kdy se data změní. Tato nová záložka tedy slouží jako náhled na nějakou podmnožinu nasbíraných odpovědí.

V sekci pro správu odpovědí je možné také odpovědi editovat. Na editaci ovšem musí mít uživatel práva přidělená vlastníkem formuláře. Každý zásah do odpovědí podléhá revizi a každá taková úprava je svázaná s jejich autorem. Vlastník formuláře a uživatelé, kteří jsou k tomu zmocnění pomocí práv, mohou do historie úprav nahlížet a úpravy případně vracet zpět.

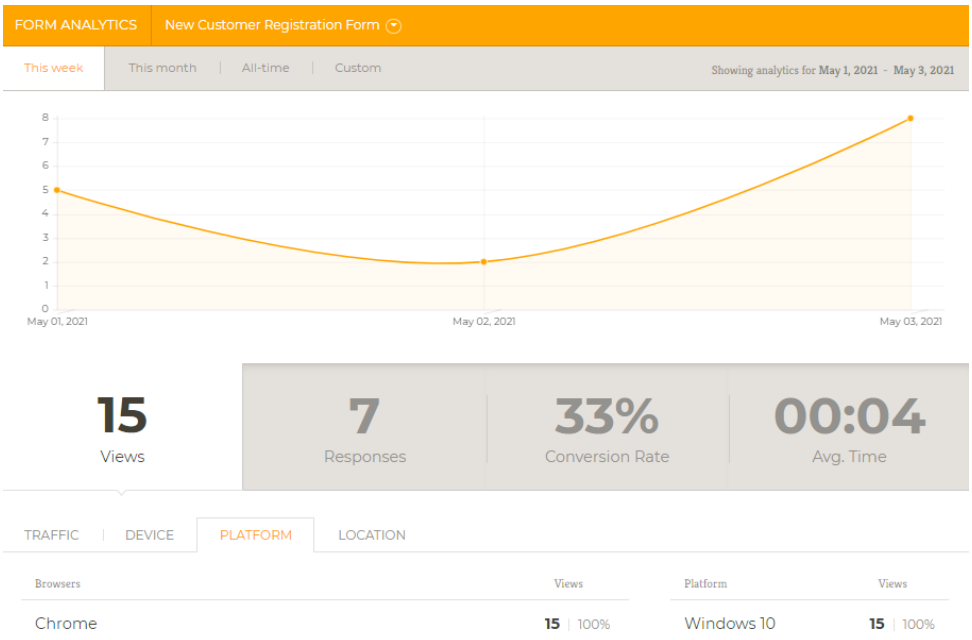
Další sférou možností, která aplikaci přidává velký potenciál, je možnost přidat libovolné množství interních sloupců do nasbíraných dat. Interní jsou proto, že nejsou součástí standardního zobrazení formuláře při jeho vyplňování respondenty. Tento typ sloupců slouží jako prostor pro interní data, kterými mohou být manuálně vložená data nebo automaticky odvozené hodnoty. Základními typy sloupečků pro manuální vložení jsou např. *text*, *email*, *telefonní číslo*, *datum* apod. Automatické hodnoty mohou být daty z jiných sloupců stejné tabulky, mohou ale nabývat hodnot i ze sloupců úplně jiných formulářů. Hodnoty mohou být odvozené také pomocí pokročilých formulí, resp. pomocí sady operací nad libovolnými sloupečky. Těmito operacemi jsou např. *concatenate*, *search*, *replace*, *average* a mnoho dalších. Většinou se jedná o operace dostupné i v jiných tabulkových nástrojích typu Tabulky Google [24] nebo Ms Excel.



Obrázek 3.8: Náhled nasbíraných odpovědí v JotForm.

Analyzovat formuláře, resp. jejich odpovědi, lze ze dvou pohledů. Jedním pohledem je vizualizace odevzdaných dat pomocí grafů. Jednoduché grafy se snadno a rychle vytvářejí přímo ze sloupečků, ty složitější je možné je vytvořit v samostatné sekci reportů. Druhým pohledem je disciplína zvaná Form Analytics [25]. Tato disciplína se běžně zabývá obecnou interakcí uživatelů s webovými formuláři. JotForm v tomto ohledu poskytuje stručný přehled uživatelů, kteří formulář alespoň zobrazili. Součástí tohoto přehledu jsou

IP adresy návštěvníků, jejich rozlišení monitoru, lokace, čas návštěvy atd. Mimo to je zde k dispozici tzv. *conversion rate* představující podíl zobrazení formuláře vůči počtu skutečně odevzdaných formulářů.



Obrázek 3.9: Sekce Form Analytics nástroje JotForm.

V oblasti spolupárce je JotForm také poměrně intuitivním a dobře použitelným nástrojem. Jak bylo zmíněno výše, nástroj disponuje systémem práv a přístupů, pomocí nichž je možné snadno řídit kompetence jednotlivých spolupracovníků. Nasbíraná data je možné v případě potřeby exportovat do formátu *CSV*, *Excel*, či *PDF*. Nástroj je např. díky historii odpovědí i celých formulářů vhodný používat i dlouhodobě.

3.5.2 Nástroje pro správu klinických studií

Předmětem zkoumání v této kapitole je nástroj OpenClinica [26]. Tento nástroj spadá do kategorie systémů, sloužících k vedení a správě klinických studií. Systémy pro klinické studie jsou označovány i jako eCRF systémy (Electronic Case Report Form system) [27], což mimo jiné spadá do oblasti systémů známých jako EDC systémy (Electronic Data Capture system) [28]. Společnými rysy těchto nástrojů jsou většinou následující možnosti:

1. vytvářet/designovat formuláře pro pacienty

2. uchovávat a spravovat výsledky studií - resp. vyplněné formuláře
3. výsledky validovat a případně opravovat
4. výsledky exportovat, případně umožnit jejich analýzu

Cíle této práce a funkce zmíněných EDC systémů se prolínají zejména v oblasti správy výsledků studií a jejich analýzy - to znamená rozboru za účelem posouzení výsledků studií, ale také za účelem posouzení funkčnosti formuláře z hlediska designu. Přínos posouzení výsledků studií je zřejmý, vyhodnocuje se s jejich pomocí, zda studie naplnila očekávání, nebo ne. Avšak přínos posouzení funkčnosti už tak zřejmý není.

Ve fázi vytváření formulářů pro pacienty některé nástroje umožňují rovnou vyplňovat testovací data a validovat funkčnost celého formuláře. V některých případech však nástroje tak propracované nejsou a jejich validitu lze posoudit pouze tak, že jejich autor formulář vytvoří, vyplní a odevzdá. Až tehdy může vyplněný formulář podrobit analýze některých funkčních kritérií. To znamená, že analytický nástroj nepřispívá pouze k přímému posouzení výsledků, pro které celá studie vznikla, ale také napomáhá vylepšení formuláře na začátku a v průběhu studie.

Obsahem následujících odstavců je tady rozbor nástroje OpenClinica z hlediska správy formulářů a možností analyzovat jejich výsledky.

Pro kontext je dobré zmínit, že sběr dat pro klinické studie dnes již většinou probíhá elektronickou formou (viz eCRF [27]). Není však výjimkou, že ke sběru dat je použit také papírový formulář (viz pCRF [29]). Je běžnou praxí, že výsledky, které jsou papírovou formou odevzdány jsou ve finále zadány počítače tak, aby data byla elektronicky zpracovatelná a celkově analyzovatelná. Databáze, která následně obsahuje všechny elektronicky zpracované údaje, je pak vhodným objektem pro připojení nějakého dalšího nástroje, který s daty pracuje. Takovým nástrojem je i Manažer sémantických formulářů, jehož implementace je součástí této práce.

OpenClinica

OpenClinica [26] je webová aplikace pro vytváření, spravování a analyzování klinických studií. TechnickýmI parametry spadá do kategorie EDC [28] a CDM [30] systémů. Je to tedy program typický pro systémové ověřování a testování zdravotnických léčiv, vakcín, léčebných postupů apod.

OpenClinica je první open-source EDC nástroj, který přišel na trh už v roce 2005. Jde o široce používaný nástroj, pokrývající prakticky všechny funkční požadavky, které v této oblasti nastoluje vědecké prostředí a konkurenční nástroje. OpenClinica je rozšiřitelným a modulárním systémem, jenž je aktuálně

dostupný ve třech různých verzích [24]:

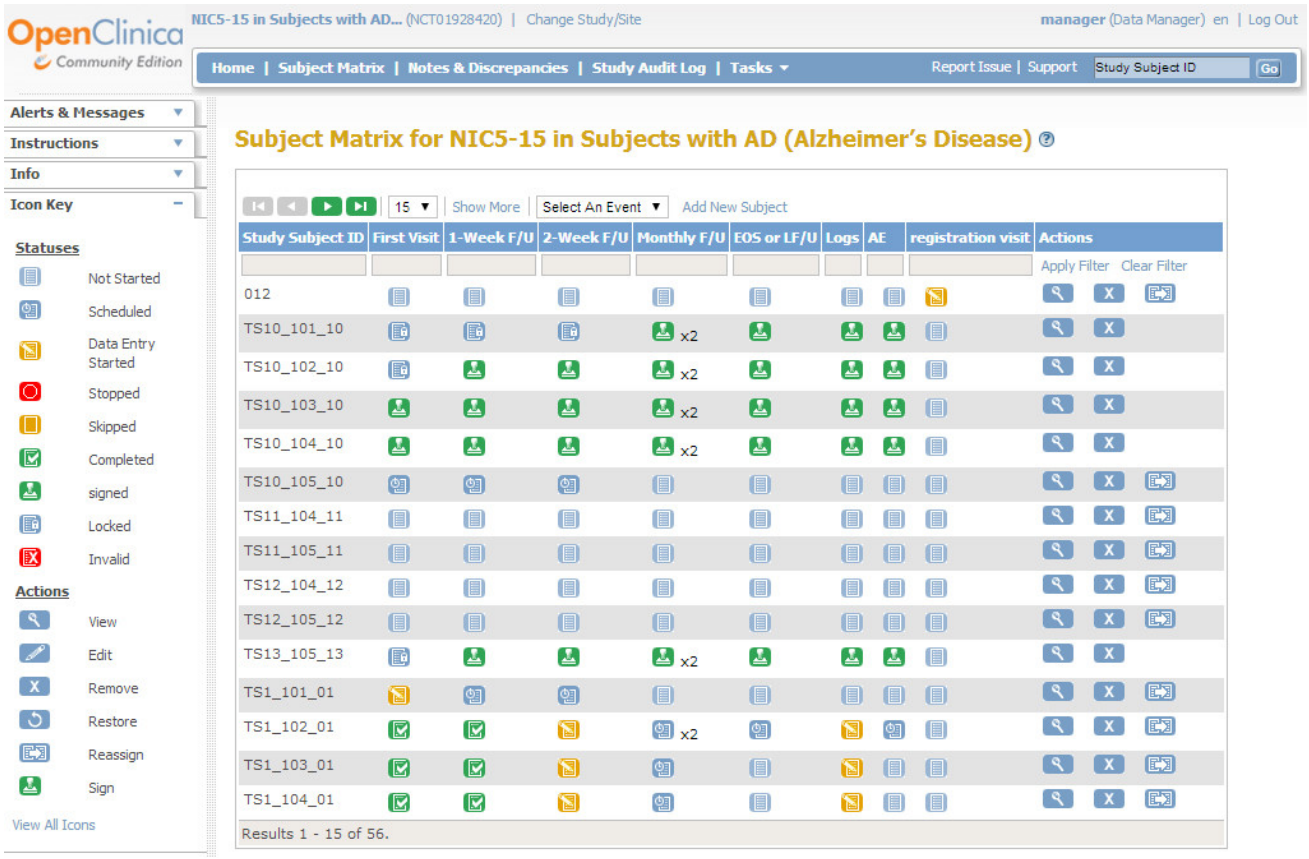
- Community Edition (bezplatná verze)
- Enterprise - OpenClinica verze 3
- Enterprise - OpenClinica verze 4

Klíčovými funkcemi nástroje je evidování studií, vytváření eCRF formulářů a monitoring a správa nasbíraných dat. Kromě toho je v aplikaci možné plánovat schůzky s pacienty, importovat/exportovat, či extrahovat data. Počet funkcí, především těch pokročilých, je možné rozšířit přidáním OpenClinica modulů. Z nich je z hlediska této práce zajímavý pouze modul OpenClinica Insight [25], který v aplikaci otevírá další možnosti pro vytváření reportů z výsledných dat.

Z pohledu správy studií je v nástroji možné procházet jednotlivé případy, zobrazovat přiřazené formuláře a jejich respondenty. Každý respondent může být součástí libovolného počtu studií a mohou mu být přiřazené různé formuláře k vyplnění. Průběh studie je mapován sadou událostí, se kterými jsou formuláře spojené. Pacient má v rámci studie naplánovaný počet schůzek, během nichž vyplňuje různé formuláře. Studie je pro konkrétní subjekt (většinou) dokončená, jestliže proběhly všechny její události. Události se mohou nacházet ve stavu *nenaplánovaný*, *naplánovaný*, *aktuálně vyplňovaný*, *hotový*, *přeskočený*, *zastavený* a *přerušený*.

V aplikaci je možné vyhledávat formuláře podle omezeného množství informací v matici subjektů, kterými jsou *jméno*, *informační štítky*, *přístupové štítky* a *členové studie*. OpenClinica verze 4 umožňuje do této matice přidávat i další sloupečky, jejichž obsah tvoří vybraná metadata subjektů, ale ta jsou omezená. V rámci studie lze využít matici subjektů pro vyhledávání účastníků (subjektů) studie podle *ID*, *stavů* a *datumů jejich událostí*, viz obrázek 3.10.

[24] https://www.openclinica.com/compare/
[25] `https://docs.openclinica.com/oc4/insight/`



Obrázek 3.10: Matice subjektů v OpenClinica 3.
Zdroj: `https://github.com/OpenClinica/OpenClinica`

Modul OpenClinica Insight. Modul OpenClinica Insight je zásuvný modul pro dotazování a vizualizaci dat ve vedených studiích. Pro účely vytváření reportů je v modulu zavedený koncept otázek, což jsou přepoužitelné dotazy do používané databáze (PostgreSQL [31]). Model databáze je zde navržený tak, že pro každou vedenou studii zde existuje vlastní schéma, jejímiž tabulkami jsou eCRF [27], sloupečky tvoří otázky a hodnotami těchto tabulek jsou odpovědi subjektů. Uživatelské rozhraní modulu používá software Metabase [26], jehož součástí je podpora pro vytváření dotazů, ale také podpora pro prezentaci výsledků.

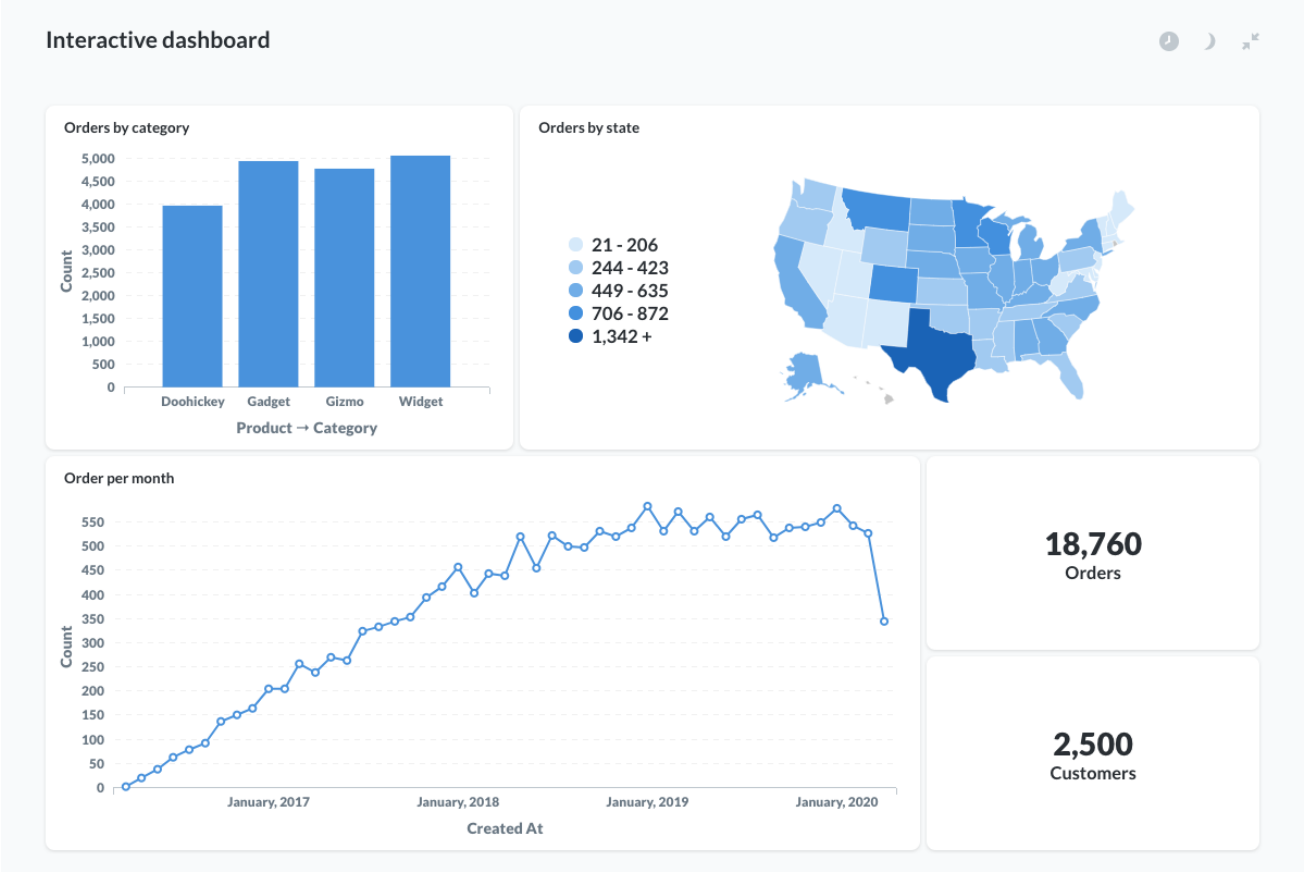
Vytvářet dotazy, resp. otázky, lze pomocí přímo SQL nebo vestavěného *builderu*. Tento tzv. builder není tak mocný jako samotné SQL, ale je výrazně snazší ho používat pro netechnické uživatele. Jeho základními funkcemi jsou filtrování, agregace a groupování dat. I pomocí tohoto builderu je tedy možné cílit dotazy na omezenou množinu otázek či respondentů a dělat na nich analýzy.

Výsledky z otázek, které jsou předkládány databázi, je možné interpretovat

[26] https://www.metabase.com/docs/latest/users-guide/start.html

buď textově nebo graficky. Software Metabase za tímto účelem disponuje velkým množstvím přizpůsobitelných grafických prvků, které lze použít jak ke grafické, tak textové reprezentaci výsledků. Těmito prvky jsou např. interaktivní sloupcové a koláčové grafy a mapy.

Pro sledování změn v datech je v tomto softwaru zavedený koncept tzv. dashboardů, viz 3.11. Tyto přizpůsobitelné dashboardy slouží jako živý přehled výsledků z provedených analýz. Jde o přehled výsledků několika dotazů a výsledky, které jsou aktualizovány v čase (historie hodnot).



Obrázek 3.11: Příklad dashboardu vytvořeného pomocí Metabase, resp. OpenClinica Insight
Zdroj: `https://www.metabase.com/docs/latest/users-guide/07-dashboards.html`

Kapitola 4

SForms

V rámci oficiální dokumentace se knihovna SForms sama označuje za procesor a generátor sémantických formulářů, které jsou založené na ontologiích. Jinými slovy je to nástroj, který pracuje se sémantickými formuláři, které dokáže za určitých podmínek zpracovat a také produkovat, resp. vizualizovat. Autory této knihovny jsou pracovníci Skupiny znalostních softwarových systémů KBSS [1] na ČVUT FEL, přičemž její základy byly položeny v roce 2016.

Tato kapitola poskytne širší vhled do problematiky, kterou knihovna SForms řeší. V první řadě ukáže funkce, kterými knihovna disponuje. V další řadě popíše formát, který musí respektovat vstupní data, aby byla z hlediska SForms validní a zpracovatelná. Následně popíše také technologický stack knihovny a způsoby, jakými vznikají její vstupní data.

Knihovnu SForms je možné snadno integrovat do webových služeb. Nutností pro tyto služby je pouze podpora několika vybraných technologií. Kromě knihovny samotné však existuje řada projektů, které možnosti a potenciál tohoto nástroje uvádí do úplně jiné roviny a které je pro hlubší porozumění SForms a jeho příkladů užití nezbytné znát. Těmito projekty jsou např. knihovny S-Pipes (kapitola 4.4), Study-Manager (kapitola 4.5) a Semantic Form Editor (kapitola 4.3.1).

[1] `http://cs.fel.cvut.cz/en/kbss`

4.1 Funkce a možnosti SForms

SForms je knihovna, která ve webovém prostředí vizualizuje data založená na ontologiích jako formuláře. Je to knihovna, jejímž vstupem je JSON-LD v předepsaném formátu a jejímž výstupem je chytrý webový formulář. Následující odstavce představí podobu a možnosti uživatelského rozhraní, ale také formát dat, se kterými SForms pracuje.

4.1.1 Uživatelské rozhraní

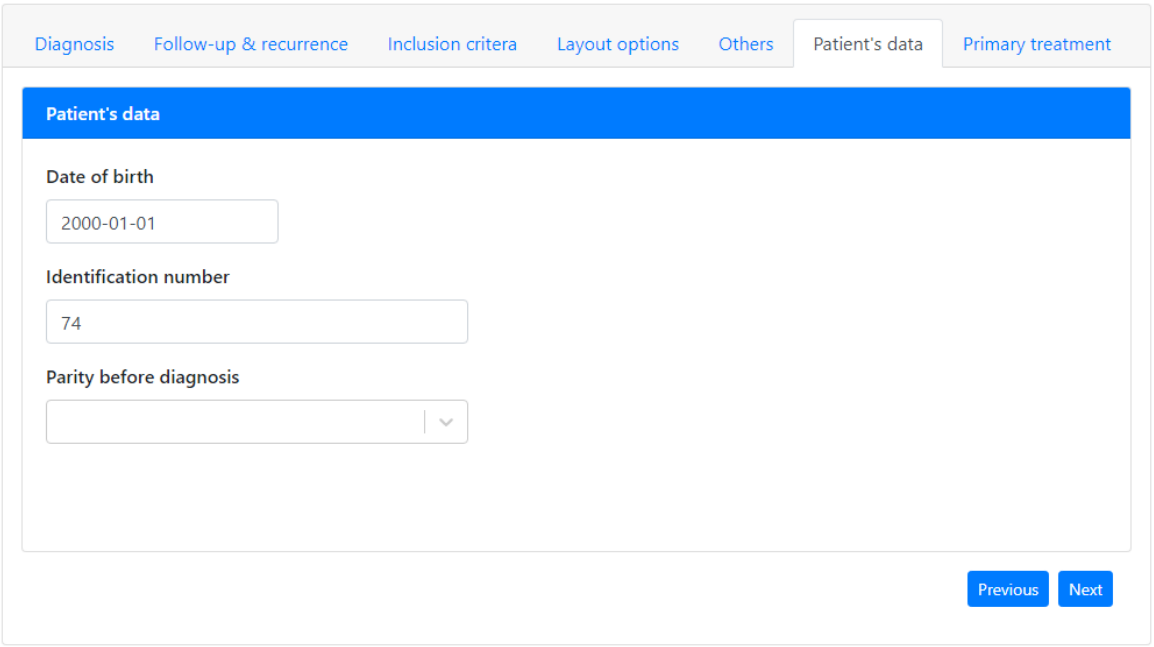
Z hlediska koncového uživatele se SForms příliš neliší od standardních formulářů, které vznikají např. pomocí Google Forms a JotForm. V rámci uživatelského rozhraní SForms podporuje 8 typů formulářových polí s celou řadou validací a ve vícestránkových formulářích 2 typy průvodců (horizontální a vertikální navigace). Kromě toho však disponuje možností dynamicky měnit podobu dalších částí formuláře na základě vyplněných dat. Taková funkce, kromě jiných, pomáhá zpřehlednit i velmi dlouhé a komplexní formuláře, na které se SForms také orientuje.

Tabulka 4.1 znázorňuje formulářová pole a ovládací prvky, které SForms podporuje. Tabulka je zobrazuje také jména parametrů, které je třeba použít pro jejich identifikaci v rámci vstupních dat - tedy aby se pro otázky vykreslil správný ovládací prvek. Tyto parametry musí být součástí vlastnosti vstupních uzlů *form-lt:has-form-layout-class*, viz kapitola 4.1.2.

Podoba výsledného formuláře je tedy celkem standardní a na první pohled není zjevné, že se jedná o něčím význačný nástroj, viz obrázek 4.1. To samé však nelze říci při pohledu na vstupní formát dat a proces tvorby formulářů, kde je na první pohled jasné to, že se SForms mezi typické nástroje neřadí.

Vstupní pole	Mapování parametrů
Textové pole	text
Delší text (textarea)	textarea
Text s maskou	masked-text
SPARQL syntax text (with Yasgui editor)	sparql
TURTLE syntax text (with Yasgui editor)	ttl
Checkbox	checkbox
Datum	date
Datum a čas	datetime
Dropdown s nápovědou	type-ahead
Media (video / obrázek)	section (+ vlastnost form:has-media-content)
Sekce	section
Krok v průvodci	wizard-step
Sekundární vlastnosti: disabled, answerable, collapsed, emphasized, category-x (x: 1-5)	

Tabulka 4.1: Tabulka ovládacích prvků v SForms formulářích a jejich mapování na identifikátory.



Obrázek 4.1: Příklad uživatelského rozhraní formuláře SForms.

4.1.2 Vstupní data

Vstupní data pro knihovnu SForms musí být serializovaná ve formátu JSON-LD. Vlastnosti uzlů a tzv. kontext, na který se mapují tato data jsou součástí ontologií vyvíjených skupinou KBSS na ČVUT FEL. Konkrétně ontologie SForms je definována separátně v repozitáři KBSS [2]. Na základě pořadí uzlů ve vstupním JSON-LD a jejich vlastností v této struktuře se určuje pořadí a grafická podoba výsledného formuláře.

Rozeznávané typy uzlů, které je možné ve vstupních datech definovat jsou *otázka*, *odpověď*, *podmínka* a *možnost odpovědi*, viz tabulka 4.2. Data ve formátu JSON-LD musí mít také pro zpracování pomocí SForms plochou strukturu - a to i přesto, že zdrojová data jsou většinou reprezentována v grafových databázích. Řešení problému převedení grafu do ploché struktury (jako je např. sekvence) je však známé a používá se pro něj typicky Flattening algoritmus [32].

Typ uzlu	**URI**	**Popis**
otázka	doc:question	uzel s informacemi o vstupním poli formuláře
odpověď	doc:answer	uzel s hodnotou vyplněné odpovědi
podmínka	form:condition	uzel s definicí podmínky (např. pro zobrazení dalších otázek)
možnost odpovědi	*nespecifikováno*	uzel s hodnotou, kterou lze vybrat v otázkách s definovaným rámcem odpovědí
použité prefixy: doc - http://onto.fel.cvut.cz/ontologies/documentation/ form - http://onto.fel.cvut.cz/ontologies/form/condition/		

Tabulka 4.2: Typy uzlů, které lze definovat ve vstupních datech pro SForms.

Některé vlastnosti uzlů jsou společné pro všechny typy (tabulka 4.3), ale většina jich je specifická pro každý typ - pro otázky (tabulka 4.4) a pro odpovědi (tabulka 4.5). Uzly pro možnosti odpovědí se většinou skládají pouze z vlastností *@id* a *rdfs:label*.

Za zvláštní pozornost u otázek stojí vlastnost *doc:has_related_questions*, která u otázek udržuje list podotázek. Díky této vlastnosti je možné vytvořit stromovou strukturu formuláře. Dynamicitu formuláře z hlediska skrývání otázek potom zajišťuje vlastnost form:is-relevant-if, jejímiž parametry jsou zmíněné podmínkové uzly.

[2] `http://cs.fel.cvut.cz/en/kbss`

URI	Datový typ	Popis
@id	String	Identifikátor uzlu (většinou jedinečný pouze v rámci samotného dokumentu)
@type	doc:question doc:answer form:condition	Jeden ze tří podporovaných typů uzlů
dc:description	String	Nápověda k uzlu
rdfs:comment	String	Komentář. Není zobrazován v SForms, ale je i tak často přítomný
použité prefixy: dc - http://purl.org/dc/elements/1.1/ rdfs - http://www.w3.org/2000/01/rdf-schema/		

Tabulka 4.3: Vlastnosti, které lze definovat pro všechny typy uzlů.

URI	Datový typ	Popis
rdfs:label	String	Textový popis vstupního pole
form:has-question-origin	String	Jednoznačný identifikátor otázky napříč formuláři
form:has-question-origin-path	String	Jednoznačný identifikátor otázky a její pozice ve stromu otázek
doc:has_related_questions	List referencí na otázkové uzly	List referencí na podotázky
doc:has_answer	Reference	Reference na odpověď otázky
form-lt:has-form-layout-class	List stringů	Určuje výslednou podobu vstupního pole otázky
form:is-relevant-if	List referencí na podmínkové uzly	Určuje za jakých podmínek se otázka zobrazí
form:has-possible-value	List referencí na možnosti odpovědí	Určuje možné odpovědi, které lze vybrat v otázce (vylučuje se s form:has-possible-values-query)
form:has-possible-values-query	String	Určuje link, jehož součástí je query pro výběr možných odpovědí (vylučuje se s form:has-possible-values)
použité prefixy: rdfs - http://www.w3.org/2000/01/rdf-schema/ doc - http://onto.fel.cvut.cz/ontologies/documentation/ form - http://onto.fel.cvut.cz/ontologies/form/ form-lt - http://onto.fel.cvut.cz/ontologies/form-layout/		

Tabulka 4.4: Vlastnosti, které lze definovat pro otázky.

Dále za pozornost stojí vlastnost *form-lt:has-form-layout-class*, pomocí níž se specifikuje podoba vstupního pole otázky (ovládacího prvku). Parametry pro tuto vlastnost se dělí na primární a sekundární, přičemž primární může být pouze jeden, ale sekundárních libovolné množství. První skupina parametrů určuje typ vstupního pole, jako je např. textové pole, datum apod. Sekundární parametry určují speciální styly, jako je např. zvýraznění, kategorie, apod. Ovládací prvky je možné vidět v tabulce 4.1, která mapuje ovládací prvky na jména jejich parametrů.

Specifikovat uzly pro odpovědi je možné s pomocí jen několika vlastností, viz tabulka 4.5. Uzly s odpovědí se dělí prakticky na dva typy. Odpověď má buď formu řetězce (text odpovědi), a nebo je odpovědí nějaký další uzel. Tyto typy se rozlišují na základě přítomnosti vlastnosti buď *doc:has_data_value* nebo *doc:has_object_value*.

URI	**Datový typ**	**Popis**
doc:has_data_value	String	Text odpovědi (vylučuje se s doc:has_object_value)
doc:has_object_value	Reference na další uzel	Odpověď ve formě reference na další uzel (vylučuje se s doc:has_data_value)
form:has-answer-origin	String	Jednoznačný identifikátor odpovědi napříč formuláři
použité prefixy: doc - http://onto.fel.cvut.cz/ontologies/documentation/ form - http://onto.fel.cvut.cz/ontologies/form/		

Tabulka 4.5: Vlastnosti, které lze definovat pro odpovědi.

Formát podmínkových uzlů je jednoduchý - eviduje pouze otázky, které jsou na něm závislé a podmínku pro splnění, viz tabulka 4.6.

URI	**Datový typ**	**Popis**
form:has-tested-question	List referencí na otázky	Eviduje otázkové uzly, které jsou na podmínkovém uzlu závislé.
form:accepts-answer-value	List stringů nebo referencí na uzel možné odpovědi	Validační pravidlo podle řetězce nebo uzlu možné odpovědi
použitý prefix: form - http://onto.fel.cvut.cz/ontologies/form/		

Tabulka 4.6: Vlastnosti, které lze definovat pro podmínkové uzly.

4.2 Technické řešení a integrace SForms

SForms je open-source knihovna, jejíž kód je dostupný na GitHub. Závislosti použité v SForms 0.2.1 jsou:

- React 16.9
- React Bootstrap 1.0.1 (kompatibilní s Bootstrap 3)
- jQuery 2.2.4
- Babel 6.26
- ECMAScript 6

Tuto knihovnu je doporučené používat v prostředí, které podporuje výše zmíněné závislosti. Pro integraci SForms do webové aplikace je nutné importovat styly *s-forms/css/s-forms.min.css* a renderovat React komponentu *s-forms/SForms*, které lze předložit JSON-LD formulář jako vstup.

4.3 Nástroje pro tvorbu vstupních dat

Způsoby, kterými je možné vytvářet vstupní data pro formuláře SForms jsou dva. Nástrojem s grafickým rozhraním, který se specializuje především na tvorbu formulářů SForms je Semantic Form Editor (kapitola 4.3.1). Další možností je využít službu FormGen (kapitola 4.3.2), která v principu transformuje existující databázi otázek a odpovědí do JSON-LD formátu, který je pro SForms čitelný.

Při sběru dat je samozřejmě žádoucí mít nějaký kontext, v němž respondent vyplňuje formulář. Metadata o respondentech (identifikace uživatele, datum vyplnění apod.) typicky nejsou součástí otázek ve formulářích. Tuto problematiku lze řešit více způsoby, ale např. pro vedení klinických studií existuje nástroj Study Manager (kapitola 4.5). Study Manager totiž kromě výsledků z vyplněných formulářů eviduje také pacienty, instituce a další zainteresované subjekty (vše definované ve veřejných ontologiích pracoviště KBSS). Vstupními daty jistě mohou být i jiné formuláře. Klinické studie jsou však poměrně dobrý příklad, na kterém lze tuto problematiku ukázat.

Tyto studie, jak je popsáno v kapitole 3.5.2, jsou většinou poměrně komplexní proces získávání a analyzování dat. Jejich formuláře jsou často velmi

rozsáhlé a otázky propracované - a to jak z odborného, tak technického hlediska. Je tedy jasné, že otázky musí vytvářet specialista na danou doménu. Z tohoto důvodu vznikl jednoduchý formát pro specifikaci formuláře netechnickými uživateli, který je vidět ve zdrojovém kódu 4.1. Tento formát je poměrně jednoduchý na zápis pro netechnického člověka, ale také snadno strojově zpracovatelný. Jeho součástí je definice několika málo pravidel pro vytváření sekcí a různých typů otázek. Příklad tohoto formátu, jehož oddělovačem je tabulátor je také vidět ve zdrojovém kódu 4.1. K tomu je veřejně dostupný také na webu v plném rozsahu [3].

```
ABRAX study
  Inclusion criteria
    Histologically confirmed invasive cancer [No,Yes]
      Stage pT1a - pT2b [No,Yes]
      Patient referred for primary surgical treatment [No,Yes]
      Intraoperative detection of LN [No,Yes]
      Follow-up data available for more than 2 years [No,Yes]
      Surgery performed after January 2005 [No,Yes]
  Patient's identification and history
    Date of birth (month/year)
    Second primary cancer (previous or simultaneous)
      Breast cancer [No,Yes]
        Date of diagnosis (year)
        Treatment
          Chemotherapy [No,Yes]
```

Zdrojový kód 4.1: Příklad netechnického formátu pro stavbu formulářů.

4.3.1 Semantic Form Editor

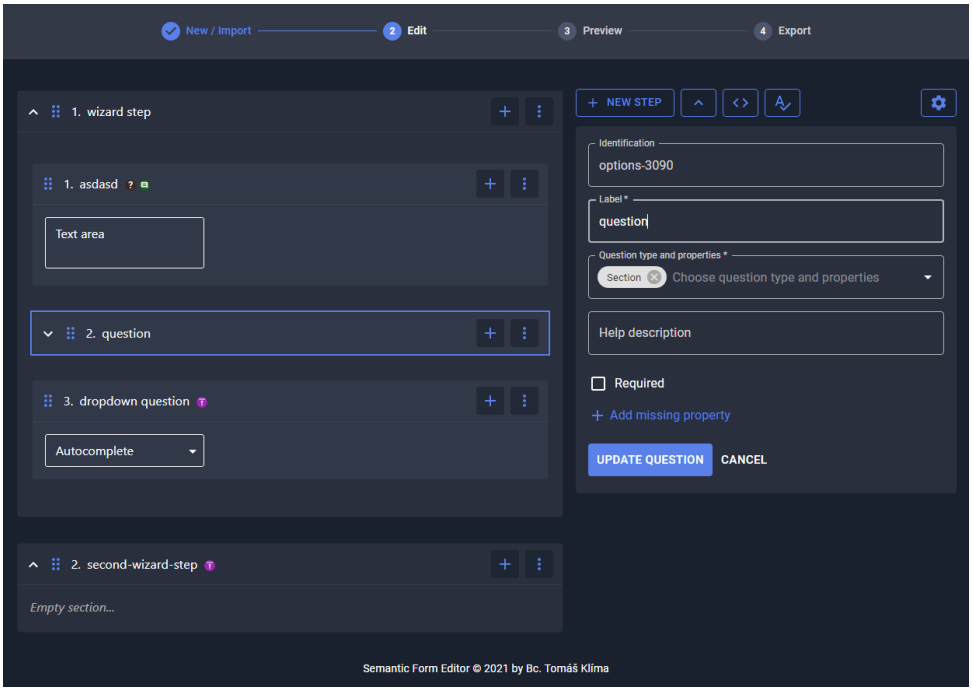
Semantic Form Editor [40], nebo-li editor sémantických webových formulářů je open-source projekt, který slouží k tvorbě a editaci formulářů SForms. Projekt vznikl v roce 2021 a spadá pod stejné pracoviště jako SForms (ČVUT FEL KBSS).

Klíčové funkce tohoto nástroje jsou:

1. import formuláře ve formátu JSON-LD

[3] https://trial4you.eu/abrax-study/form/abrax-form.pdf

2. vytvoření nového formuláře
3. interaktivní grafické rozhraní pro přidávání ovládacích prvků SForms
4. náhled, validace a export formuláře v JSON-LD formátu



Obrázek 4.2: Příklad použití SForms editoru.

Tento způsob tvorby formulářů SForms je velmi vhodný pro účely návrhu rozsáhlých studií. Vhodný je zejména proto, že poskytuje okamžitý náhled grafické verze formuláře, ale zároveň umožňuje technickým uživatelům upravovat jeho zdrojová data ručně.

Grafické rozhraní poskytuje možnost vytvářet různé typy otázek, či dokonce podotázek a umožňuje jim rovnou přiřazovat sekundární atributy, které SForms podporuje. Pomocí tzv. drag-and-drop je možné měnit pořadí otázek. Formulář je také možné rozdělit na několik sekcí, mezi kterými se lze pohybovat pomocí tzv. wizardu, resp. průvodce (horizontální a vertikální).

4.3.2 FormGen

FormGen je služba, kterou lze použít k vytvoření JSON-LD reprezentace SForms formulářových dat. Její implementace používá projekt SPipes, popsaný

v kapitoly 4.4, k obohacení a transformaci databázových dat, ze kterých vzniká SForms formulář.

Službu je možné volat pomocí HTTP dotazu, jehož odpovědí je právě formulář ve formátu JSON-LD. Parametry dotazu určují výslednou podobu formuláře, a to včetně otázek a odpovědí. Těmito parametry však nejsou data samotná - jsou jimi pouze adresy příslušných repozitářů a identifikátor zpracovávaných dat:

- parametr *repositoryUrl* - URL App repozitáře (kapitola 4.3.2)
- parametr *formGenRepositoryUrl* - URL FormGen repozitáře (kapitola 4.3.2)
- parametr *recordGraphId* - identifikátor RDF4J kontextu s formulářovými daty (součást FormGen repozitáře)

Už ze zmíněných parametrů je patrné, že daný nástroj pracuje s dvěma datovými úložišti. Repozitáře musí být součástí RDF4J [35] databáze a říká se jim *app repository* a *formgen repository*. Jak je zmíněno výše, služba používá nástroj SPipes k transformaci otázek a odpovědí do formulářové podoby, a obě tato úložiště k tomu používá.

FormGen repository ve zmíněné transformaci slouží jako zdroj otázek a odpovědí, které se mají pomocí SPipes přetvořit. Důležitým faktem však je to, že otázky zde nejsou reprezentovány přímo textem, nýbrž pouze jejich identifikátory. To znamená, že udržovaná je zde jenom jejich struktura (pomocí *form:has_related_question*) a ne textové znění. Pokud jde o odpovědi, jsou součástí repozitáře všechna data - včetně textu.

Chybějící část formgen repozitáře (tedy skutečný text otázek a další) existuje v *app repository*. Pro každou otázku, která může být součástí formuláře, je zde vedený tzv. question template, neboli vzorová otázka. Vlastností této vzorové otázky je několik, jednou z vlastností např. je, že její text je vedený pod tzv. labelem (rdfs:label).

SPipes je tedy ve FormGen službě nakonfigurována tak, aby přečetlo strukturu otázek z formgen repozitáře a konkrétní data otázek (text apod.) z app repozitáře. Po postupném přečtení, zpracování a kombinování obou těchto zdrojů následuje zploštění grafu pomocí tzv. Flattening alrogitmu. Takto zploštělý graf, převedený do JSON-LD formátu lze zobrazit jako formulář SForms.

App repozitář

App repozitář je RDF4J repozitář, který obsahuje statická data formulářů, jakými jsou např. text otázek a aktuální vzorová formulářová struktura. V případě použití Study Manageru, popsaného v kapitole 4.5), jsou v tomto repozitáři také záznamy pacientů, institucí a dalších subjektů, které Study Manager eviduje.

FormGen repozitář

FormGen repozitář je RDF4J repozitář, který uchovává informace o struktuře formulářových otázek a jejich odpovědi. Skutečné znění každé otázky je uchováváno v app repozitáři (kapitola 4.3.2), kde jej lze pro každou otázku dohledat pomocí tzv. question originu (*form:has-question-origin*).

Podoba formuláře je zde uložena do samostatného kontextu pokaždé, kdy je formulář zobrazen nebo upraven. Každé uložení je tedy otisk aktuálního stavu formuláře při jeho zobrazení. To znamená, že mnoho těchto záznamů je prakticky stejných, a pokud se liší, tak především v odpovědích (nebo při změně formuláře ve struktuře otázek).

Způsob ukládání dat je však závislý na nástroji, který tyto formuláře vytváří a který tedy s těmito repozitáři pracuje - viz Study Manager (kapitola 4.5).

4.4 SPipes

SPipes [36] je nástroj, pomocí něhož je možné transformovat data v sémantických proudech. Nástroj je inspirovaný technologií SPARQLMotion [37], jejíž funkce v mnoha ohledech napodobuje či kopíruje. Koncept této technologie je založený na paradigmatu prodouvě orientovaného zpracování a může tak konceptem připomínat třeba Java Stream API [38]. Zdrojový kód SPipes je open-source a je vyvíjený skupinou KBSS ČVUT FEL [4], stejně jako SForms (kapitola 4).

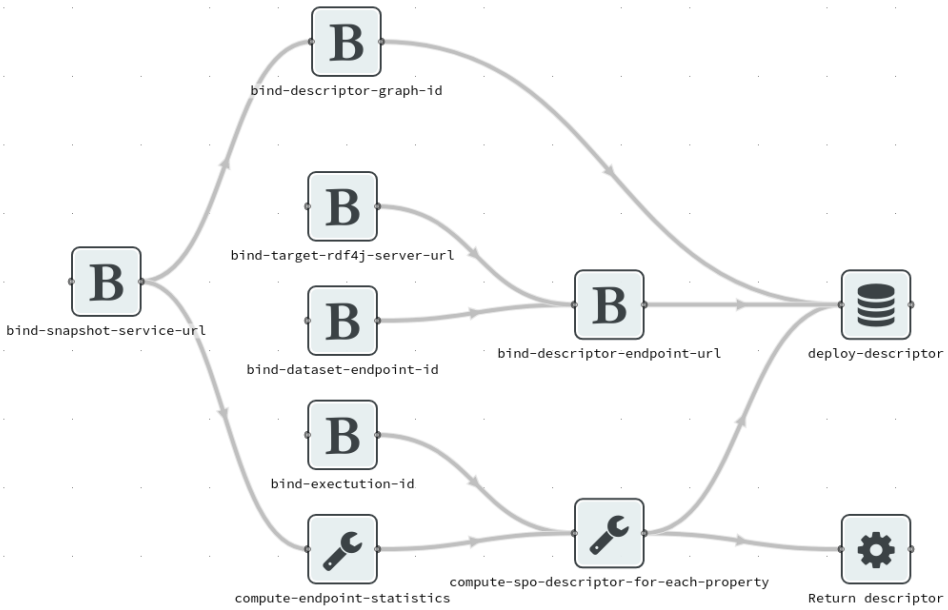
Pro transformaci sémantických proudů je možné v SPipes definovat řadu transformačních modulů. Každý modul zde definuje pravidla pro nějakou transformaci buď pomocí jazyka SPARQL nebo pomocí Java tříd [39]. Pořadí těchto modulů je jednoznačně definováno pomocí skriptu.

Formátem dat, se kterými SPipes pracuje, je typicky RDF, XML nebo

[4] `http://cs.fel.cvut.cz/en/kbss`

JSON-LD. SPipes disponuje také grafickým rozhraním, v němž lze editovat transformační moduly samotné, ale také jejich pořadí v rámci jednotlivých pipeline.

SPipes hraje velmi důležitou roli v transformaci formulářových dat. Služba FormGen, popsaná v kapitole 4.3.2, využívá SPipes při stavbě JSON-LD reprezentace formuláře.



Obrázek 4.3: Příklad grafického rozhraní SPipes.
Zdroj: `https://dspace.cvut.cz/handle/10467/76534`

4.5 Study Manager

Study Manager [33] je aplikace, kterou lze použít ke sběru dat během klinických studií pomocí SForms formulářů popsaných v kapitole 4. Kromě zprostředkování příslušných formulářů je její přidanou hodnotou uživatelský systém pro pacienty a instituce, které jsou do studie zapojeny. Systém je založený na ontologiích, a stejně jako SForms je vyvíjen pracovištěm KBSS ČVUT FEL [5].

Study Manager je pro tuto práci důležitý proto, že studie v něm vedené mohou být zdrojem dat, se kterými následně pracuje nástroj implementovaný v této práci - tedy Manažer sémantických formulářů.

Entity, které rozeznává nástroj Study Manager, jsou pacient, doktor, instituce, administrátor a formulář (otázka + odpověď). Typickým příkladem užití nástroje pro netechnické uživatele je:

[5] `http://cs.fel.cvut.cz/en/kbss`

1. instituce projeví zájem o zapojení do klinické studie
2. zadavatel studie v systému vytvoří záznam o instituci a uživatelské účty lékařů
3. lékaři vytvoří uživatelské účty pro své pacienty
4. pacient vyplní formulář klinické studie
5. lékař reviduje pacientův formulářový záznam

Study Manager je aktivně používaná aplikace, pomocí níž je aktuálně vedeno několik klinických studií. V současné době je také jedna naplánovaná a jedna již uzavřená. Přehled těchto studií, a to včetně specifikace formulářových otázek, je veřejně dostupný na [6].

4.5.1 Technické řešení

Součástí projektu Study Manager je serverová i klientská část. Pro implementaci serverové části je použitý jazyk Java 8 [41] a Spring Framework 5. Klientská část aplikace používá javascriptovou knihovnu React 16 a kaskádové styly Bootstrap.

Aplikace používá dvě RDF databáze, jejichž účel je rozepsaný v samostatných podkapitolách FormGen služby - app repozitář a formgen repozitář.

Jak již však bylo zmíněno, zápis dat do těchto databází se může lišit podle projektu, který s daty pracuje. V případě Study Manageru nejsou součástí app repozitáře pouze data, která rozšiřují odpovědi ve formulářích (jak je tomu popsáno v kapitole 4.3.2), nýbrž jde také o uložiště pro ostatní aplikační data, jako jsou informace o pacientech, institucích apod. Z hlediska formgen repozitáře platí, že každý záznam / context v databázi je otiskem formulářových dat v nějakém čase. Study Manager však do těchto dat přidává také záznam o pacientovi a jeho instituci v daném čase, jde tedy nejenom o formulářová data.

Study Manager každou změnu v aplikačních datech ukládá do databáze při každé jejich změně, jak by se dalo očekávat. Ne úplně typické chování je však implementováno pro úpravu SForms formulářů - tedy pro klinická data. V tomto ohledu totiž Study Manager ukládá otisk aktuálních dat (klinická data + informace o pacientovi atd.) pokaždé, když se formulář i zobrazí, ne jen při jejich editaci.

[6] `https://github.com/kbss-cvut/s-forms`

4.5.2 Statická vs dynamická data

V kontextu SForms formulářů a Study Manageru jsou rozlišovány dva typy dat - dynamická a statická.

Dynamickými daty v kontextu SForms formulářů se rozumí otázky ve formulářích. Tyto otázky respektují zavedenou ontologii SForms formulářů [7], která se prakticky nemění. Struktura otázek, jejich znění a jejich identifikátory jsou však pro každou studii naprosto rozdílné - proto jsou tato data označována za *dynamická*. Je důležité zmínit také to, že i tato struktura se může v čase měnit i v rámci jedné studie. I při změně struktury však identifikátor každé otázky zůstává stejný.

Jako statická data jsou označována ta data, která s formuláři asociuje Study Manager nad rámec ontologie SForms formulářů. Jsou jimi informace o pacientech, institucích a dalších evidovaných subjektech. Formát, resp. struktura těchto dat je statická, protože je pro každou studii stejná.

Study Manager nicméně není jediný nástroj, který s SForms formuláři pracuje a který formgen repozitáře obohacuje o vlastní informace. I tyto další nástroje definují vlastní podobu, resp. strukturu statických dat a proto i statická data se pro různé nástroje liší.

[7] `https://github.com/kbss-cvut/s-forms`

Kapitola 5

Návrh řešení

Předcházející kapitoly poskytly v první řadě vhled do technologií Sémantického webu (kapitola 2.1). V druhé řadě ukázaly a popsaly existující řešení nástrojů pro správu formulářů a tiketovací systémy, pomocí nichž je možné spolupracovat na projektech (kapitola 3.1). V kapitole 4 práce nabídla informace o principech, na kterých staví knihovna SForms.

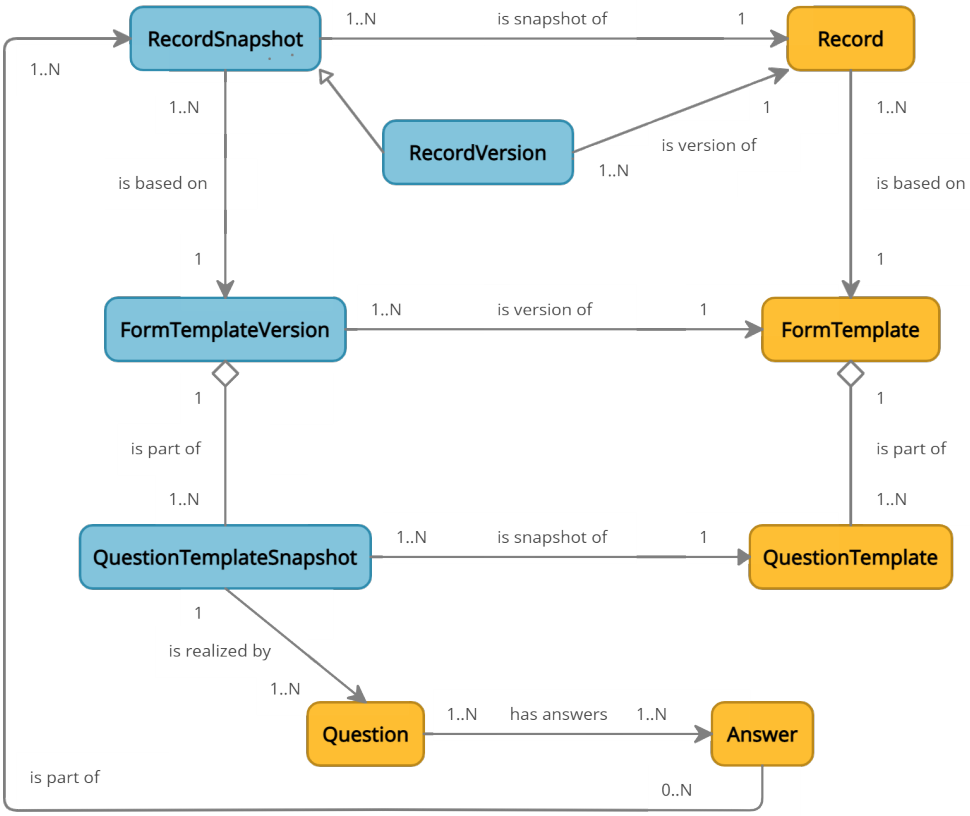
Díky analýze těchto nástrojů a technologií je možné sestavit konkrétní návrh řešení pro implementaci Manažeru sémantických formulářů. První část této kapitoly se věnuje představení konceptuálního modelu dat, se kterými implementovaný nástroj pracuje a který vychází z infrastruktury postavené okolo frameworku SForms. Druhá část této kapitoly představí konkrétní požadavky a use-case diagram aplikace. Třetí část kapitoly obsahuje uživatelské scénáře, které popisují možnou interakci uživatele s webovým rozhraním nástroje.

5.1 Konceptuální model

Rešerse provedená na frameworku SForms poskytla hrubý přehled toho, jak vznikají a jak se uchovávají formulářové záznamy. Na základě této analýzy je nyní možné vybrat klíčové entity, s nimiž je nutné při správě formulářů SForms pracovat. Tyto klíčové entity a jejich vazby jsou zobrazené pomocí konceptuáního modelu na obrázku 5.1. Cílem této sekce je tedy představit hlavní entity, resp. stavební kameny, na jejichž základě staví celá implementace.

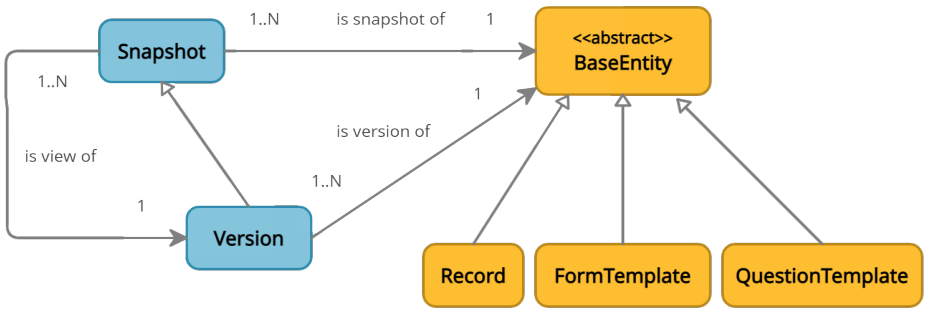
Tato sekce dané entity nejen zobrazuje, ale poskytuje také jejich vysvětlení a

kontext. Vzhledem k tomu, že zmíněný framework sám nedefinuje českou verzi jednotlivých termínů, součástí jejich popisu je také vlastní překlad. Ostatní kapitoly v textu pracují jak s originálními názvy entit, tak se zmíněnými překlady.



Obrázek 5.1: Konceptuální model vstupních formulářových dat.

Konceptuální model zobrazený pomocí diagramu 5.1 respektuje pravidla tvorby diagramů UML 2.0. Nicméně kromě standardních pravidel také používá barevné odlišení entit. Žlutě označené entity zde představují hlavní modelové třídy. Modře označené entity představují třídy, které jsou od hlavních entit nějakým způsobem odvozené. Zavedé jsou zde dva typy odvozených entit (Snapshot a Version), které se pro každou modelovou třídu odvozují jinak. Obecný vztah těchto entitních typů je zachycený na diagramu 5.2. Lze také říci, že odvozené entity reprezentují stav té původní v nějakém časovém intervalu. Hlavní entity, jako např. uživatelský záznam, se vyvíjí a každý jeho snapshot ho reprezentuje v čase od chvíle uložení do doby přepsání.



Obrázek 5.2: Diagram zachycující vztah hlavních a odvozených modelových tříd.

5.1.1 Record

Record nebo-li *uživatelský záznam*. Uživatelský záznam zachycuje informace o jednom uživateli, ke kterému se většinou váže formulářový záznam. Pro každý Record existuje právě jedna formulářová šablona a lze pro něj rozlišit jednotlivé verze a snapshoty.

5.1.2 RecordSnapshot

RecordSnapshot nebo-li *snapshot uživatelského záznamu* reprezentuje stav uživatelského záznamu v konkrétním čase. Je součástí historie právě jednoho Recordu. Podoba a struktura otázek, které byly uživateli položeny v čase uložení záznamu, jsou vyjádřeny pomocí formulářové verze (FormTemplate-Version), ke které se RecordSnapshot váže. Výjimku tvoří tzv. *empty record*, což je počáteční stav uživatelského záznamu, jemuž ještě nebyly přiřazeny otázky a který tedy nemá přiřazenou žádou formulářovou verzi.

5.1.3 RecordVersion

RecordVersion nebo-li *verze uživatelského záznamu* reprezentuje množinu snapshostů v historii uživatelského záznamu, jejichž všechny vlastnosti jsou stejné. To znamená, že množiny RecordSnapshotů, které se v rámci jednoho uživatelského záznamu liší pouze časem uložení jsou mít stejné verze. Recor-

dVersion je tedy pro každý snapshot uživatelského záznamu (nepočítáme-li empty record) stejná v případě, že:

- uživatel od prvního uložení nezměnil/nepřidal žádné odpovědi
- pro žádný RecordSnapshot se nezměnila struktura otázek
- nězměnila se ani ostatní uživatelská data (např. osobní údaje)

5.1.4 FormTemplateVersion

Verze formulářové šablony jednoznačně identifikuje množinu otázek a jejich pořadí, ve kterém byly uživateli položeny. FormTemplateVersion je spojená s jedním nebo více snapshoty uživatelských záznamů. Resp. formulářová šablona v dané verzi může být součástí jednoho nebo více RecordSnapshotů.

5.1.5 FormTemplate

FormTemplate nebo-li *formulářová šablona* reprezentuje typ formuláře podle účelu sběru dat. Identifikuje se pomocí kořenové otázky, která zpravidla obsahuje jméno studie, pro kterou je formulářová šablona vytvořená. Šablona může mít různé verze (FormTemplateVersion), které se liší strukturou otázek.

5.1.6 QuestionTemplate

Šablona otázky. Tato entita reprezentuje otázku, kterou lze položit ve formulářích. Neidentifikuje se pomocí slovního znění, nýbrž pomocí identifikátoru *question-origin*. V rámci různých formulářů může být položena s jiným textem, ale význam má stejný.

5.1.7 QuestionTemplateSnapshot

Snapshot šablony otázky. Tato entita je snapshotem otázkové šablony QuestionTemplate. Její identitu udává unikátní klíč příšlušné FormTemplateVersion

a také její pozice v rámci stromu otázek. Pro určení pozice ve stromě otázek se používá identifikátor *question-origin-path*, což je spojení identifikátorů *question-origin* jednotlivých nád-otázek cílové otázky.

5.1.8 Question

Otázka je realizací QuestionTemplateSnapshot, přičemž obsahuje např. i textové znění položené otázky.

5.1.9 Answer

Entita *Odpověď* představuje odpověď, která byla vyplněna v rámci nějakého formuláře. Jde pouze o hodnotu, která může být stejná pro různé otázky. Vždy patří do alespoň jednoho RecordSnapshotu.

5.1.10 Další termíny

- **Formulářový záznam** - pod tímto pojmem se rozumí formulářová data, která jsou součástí RecordSnapshot. Jedná se tedy o tu podmnožinu snapshotu uživalského záznamu, která se používá pro zobrazení SForms formuláře.
- **Verze formuláře** - tento pojem je synonymem pro verzi formulářové šablony.

5.2 Požadavky aplikace

Tato sekce se zabývá analýzou požadavků, které jsou kladeny na implementaci Manažeru sémantických formulářů. Dané požadavky se v první řadě dělí na funkční a nefunkční. Dále jsou zavedeny kategorie a skupiny požadavků. Kategorie požadavků odpovídají cílům, které si stanovila tato práce v zadání. Skupiny požadavků potom obsahují již konkrétní popis funkcí, jež zde mohou a nemusí být implementovány.

Kategorie požadavků podle cílů této práce jsou:

- Správa projektových dat
  - Kategorie těchto požadavků cílí na možnost uživatele spravovat formulářová data. Tím se rozumí např. zobrazení uživatelských záznamů, jejich historické snapshoty, odpovídající formulářové šablony apod.
- Analýza projektových dat
  - Tyto požadavky cílí především na možnost uživatele pracovat s projektovými daty i nad rámec možností, které poskytuje webové rozhraní aplikace. Jinými slovy, díky požadavkům v této kategorii by uživatel měl mít možnost pracovat s hrubým formátem importovaných dat. Tedy i vytvářet pokročilé dotazy, které mu mohou zodpovědět více informací o formulářových datechpo než základní stav webového rozhraní.
- Spolupráce nad projektem
  - Tato kategorie se zaměřuje na oblast spolupráce více uživatelů nad projektem, jehož součástí jsou formulářová data frameworku SForms. Požadavky v této kategorii se tedy zaměřují zejména na propojení aplikace s tiketovacím systémem.

Každá skupina požadavků se orientuje na jednu konkrétní funkci systému. Je tedy velmi časté, že požadavky v jedné skupině se navzájem rozšiřují a doplňují. Každá skupina i samotné požadavky jsou označené pomocí pořadového čísla x a y, kde x označuje číslo skupiny a y označuje číslo požadavku v rámci této skupiny. Použité značení tedy lze shrnout následovně:

- FPx - skupina funkčních požadavků
- NPx - skupina nefunkčních požadavků
- FPx.y - funkční požadavek
- NPx.y - nefunkční požadavek

Každý požadavek má dále stanovenou prioritu, která odpovídá důležitosti požadované funkce. K tomuto účelu je použitá metodika MoSCoW [42], která zavádí 4 kategorie požadavků podle priority:

- Must have (M) - *musí mít*
  - klíčový požadavek, který je nutný pro splnění zadání
- Should have (S) - *měl by mít*
  - požadavek, jehož naplnění je velmi vhodné
- Could have (C) - *mohl by mít*
  - požadavek, jehož implementace je žádoucí, ale ne nutná
- Won't have (W) - *nebude mít*
  - požadavek, s jehož implementací se nepočítá v navrženém řešení, ale který dává smysl

Požadavky v této kapitole samozřejmě vznikají na základě konzultací se zúčastněnými osobami, které se podílejí se na vývoji SForms a dalších s tím spjatých technologiích. I díky tomu je možné přihlédnou při stanovení těchto požadavků ke specifičnosti technologií sémantického webu a využít tak jejich předností.

5.2.1 Funkční požadavky

V následujících odstavcích jsou shrnuté funkční požadavky podle metodiky popsané v kapitole 5.2. Tedy jsou zde zavedené kategorie (oddělení pomocí podsekcí), skupiny požadavků (oddělení pomocí odstavců) a pak samotné požadavky s příslušným značením.

Správa projektových dat

Jeden evidovaný projekt odpovídá jedné studii a je pro něj evidována jedna formulářová šablona. Projektová data jsou data spojená pouze s jednou studií.

FP1 Správa projektu.

- FP1.1 (M) - systém umožní zaevidovat nový projekt, a tím umožní v rámci webového rozhraní spravovat projektová data

- FP1.2 (S) - systém umožní evidovat jeden nebo více projektů najednou
- FP1.3 (S) - systém umožní odebírat evidované projekty
- FP1.4 (C) - systém umožní editovat vlastnosti projektu
- FP1.5 (M) - systém zajistí, aby se data z jednotlivých projektů neprolínaly a navzájem neovlivňovaly projektové statistiky

FP2 Import projektových dat.

- FP2.1 (M) - systém umožní procházet formulářové záznamy ve vzdáleném formgen repozitáři
- FP2.2 (M) - systém během procházení vzdáleného formgen repozitáře umožní importovat snapshot uživatelského záznamu do systému
- FP2.3 (S) - systém umožní importovat snapshoty uživatelského záznamu do systému dávkově (v množství větším než jedna)
- FP2.4 (S) - systém umožní importovat všechny snapshoty uživatelských záznamů ve vzdáleném formgen repozitáři najednou
- FP2.5 (M) - systém uživateli poskytne informace o celkovém počtu importovatelných (a již importovaných) snapshotech uživatelských záznamů

FP3 Zobrazení formulářových záznamů pomocí SForms.

- FP3.1 (M) - systém umožní pomocí SForms zobrazit formulářový záznam bez nutnosti importu do systému
- FP3.2 (M) - systém umožní pomocí SForms zobrazit importovaný formulářový záznam, resp. snapshot uživatelského záznamu

FP4 Identifikace formulářové struktury.

- FP4.1 (M) - systém při importu formulářového záznamu jednoznačně identifiuke strukturu otázek formuláře a tuto informaci bude evidovat
- FP4.2 (M) - systém umožní procházet identifikované verze šablon formulářů

- FP4.3 (M) - systém umožní zobrazit verzi šablony formuláře pomocí SForms
- FP4.4 (M) - systém uživateli poskytne základní informace o verzi šablony formuláře
- FP4.5 (M) - systém uživateli poskytne informace o počtu importovaných formulářových záznamech, které odpovídají vybrané formulářové verzi
- FP4.6 (C) - systém uživateli umožní zavést interní pojmenování formulářové verze, které bude možné v rámci aplikace používat místo unikátního klíče formulářové verze (např. při práci s formulářovými záznamy)
- FP4.7 (W) - systém umožní filtrovat verze šablon formulářů přímo v aplikaci na základě vlastností této verze

FP5 Identifikace uživatelského záznamu.

- FP5.1 (M) - systém jednoznačně rozliší importované snapshoty uživatelských záznamu podle jejich autora
- FP5.2 (M) - systém umožní procházet jednotlivé uživatelské záznamy a jejich snapshoty
- FP5.3 (M) - systém uživateli poskytne informace o importovaném uživatelském záznamu a jeho snapshotech, konkrétně:
  - (S) - počet snapshotů v historii uživatelského záznamu
  - (M) - datum vytvoření uživatelského záznamu
  - (M) - datum vytvoření jednotlivých snapshotů
  - (C) - počet odpovědí v jednotlivých snapshotech
  - (S) - verze formulářové šablony jednotlivých snapshotů
- FP5.4 (C) - systém umožní filtrovat snapshoty uživatelských záznamů přímo v aplikaci na základě vlastností záznamu

FP6 Přehled importovaných dat.

- FP6.1 (M) - systém uživateli poskytne přehled četnosti všech klíčových entit, které vycházejí z importovaných dat, tedy:
  - počet importovaných snapshotů uživatelských záznamů
  - počet identifikovaných uživatelských záznamů

- počet identifikovaných verzí uživatelských záznamů
- počet identifikovaných formulářových verzí
- počet identifikovaných formulářových šablon
- počet identifikovaných otázkových šablon
- počet identifikovaných snapshotů otázkových šablon

FP7 Cachování uživatelských záznamů.

- FP7.1 (C) - systém při importování snapshotu uživatelského záznamu do systému shromáždí ve vlastní databázi všechna data tak, aby se pro stejný záznam nemusel dotazovat vzdálené projektové databáze víckrát (tedy vytvoří si lokální kopii všech použitých dat, která by mohl potřebovat použít v budoucnu)
  - Tento požadavek má smysl zejména u SForms formuláře, vygenerovaného pomocí Služby Formgen, jejíž opakované volání je časově náročné.
  - Výhodou implementace tohoto požadavku je také nezávislost na projektových databázích již ukončených projektů. Po zpracování celé projektové databáze je tedy možné projekt archivovat a neudržovat aktivní za účely dalších analýz ze strany Manažeru sémantických formulářů.

Spolupráce nad projektem

FP8 Integrace tiketovacího nástroje (TS) - vytvoření tiketu.

- FP8.1 (M) - systém umožní vytvářet tikety integrovaného TS přímo v uživatelském rozhraní aplikace
- FP8.2 (M) - systém umožní uživateli pojmenovat tiket a vytvořit jeho popis
- FP8.3 (M) - systém umožní asociovat tiket s formulářovým snapshotem
- FP8.4 (M) - systém umožní asociovat tiket s verzí šablony formuláře
- FP8.5 (M) - systém umožní asociovat tiket s jednou otázkou formuláře

- FP8.6 (C) - systém umožní asociovat tiket s více formulářovými otázkami zároveň
- FP8.7 (C) - systém jako součást tiketu vytvoří odkaz na formulářový snapshot nebo formulářovou verzi, při jehož navštívení se zobrazí asociovaný záznam v aplikaci Manažer sémantických formulářů
- FP8.8 (W) - systém umožní uživateli z aplikace navštívit detail tiketu přímo v TS pomocí odkazu přiloženého v tiketu

FP9 Integrace tiketovacího nástroje (TS) - zobrazení tiketů.

- FP9.1 (M) - systém poskytne přehled všech tiketů, které jsou asociované s vybraným projektem
- FP9.2 (M) - systém poskytne přehled všech tiketů, které jsou asociované s vybraným formulářovým snapshotem
- FP9.3 (M) - systém poskytne přehled všech tiketů, které jsou asociované s vybranou verzí šablony formuláře
- FP9.4 (M) - systém poskytne přehled všech tiketů, které jsou asociované s alespoň jednou otázkou vybrané formulářové verze
- FP9.5 (C) - systém pomocí SForms zobrazí konkrétní otázku, se kterou je asociovaný vybraný tiket
- FP9.6 (M) - systém v rámci asociovaných tiketů zobrazí základní informace o tiketu (jméno, popis, identifikátory asociovaných záznamů)
- FP9.7 (S) - systém umožní navštívit vybraný tiket přímo v TS
- FP9.8 (C) - systém zobrazí komentáře asociované s vybraným tiketem
- FP9.9 (C) - systém umožní komentovat jednotlivé tikety přímo v uživatelském rozhraní aplikace
- FP9.10 (W) - systém umožní komentovat jednotlivé jednotlivé prvky zobrazeného formulářového záznamu přímo v SForms
- FP9.11 (C) - systém umožní filtrovat tikety na základě vlastností tiketu přímo v aplikaci

Analýza projektových dat

FP10 Dotazování nad importovanými daty.

- FP10.1 (M) - systém uživateli poskytne rozhraní pro zadávání dotazů v jazyce SPARQL, které budou probíhat přímo nad importovanými daty
- FP10.2 (M) - systém uživateli pomůže vytvářet základní dotazy nad projektovými daty pomocí přednastavených dotazů
- FP10.3 (S) - systém uživateli poskytne přehled známých formulářových otázek včetně zobrazení jejich pozice ve stromové struktuře, které pak může použít při vytváření dotazů nad importovanými daty
- FP10.4 (C) - systém bude disponovat sadou přednastavených dotazů pro zjišťování pokročilých informací o importovaných formulářích v ohledech:
  - struktury formulářů
  - historie vyplňování
  - četnosti odpovědí u jednotlivých otázek (z hlediska také různých verzí formulářů)
  - rozmanitosti odpovědí u jednotlivých otázek
  - analýzy specifické interakce uživatele s formulářem
- FP10.5 (W) - systém umožní validovat formulářové snapshoty pomocí SHACL pravidel

FP11 Porovnání dvou formulářových verzí.

- FP11.1 (S) - systém uživateli poskytne možnost porovnat dvě formulářové verze z hlediska struktury otázek. Uživatel by se tedy pomocí této funkce měl dozvědět:
  - které otázky (snapshoty otázkových šablon) jsou v obou verzích stejné
  - které otázky (snapshoty otázkové šablony) v jedné verzi chybí, ale ve druhé jsou
  - které otázky (otázkové šablony) jsou položené v obou formulářových verzích, ale jsou na jiných pozicích

FP12 Porovnání odpovědí dvou formulářových záznamů.

- FP12.1 (S) - systém uživateli poskytne možnost porovnat odpovědi dvou po sobě jdoucích snapshotů jednoho uživatelského záznamu. Z tohoto porovnání by se měl uživatel dozvědět:

- kolik odpovědí je v obou verzích stejných
- které odpovědi v jednom snapshotu chybí, ale v druhém existují
- které odpovědi jsou změněné

FP13 Zobrazení statistik o importovaných entitách pomocí interaktivních grafů.

- FP13.1 (C) - systém bude uživateli prezentovat interaktivní graf s výskytem snapshotů uživatelských záznamů v čase podle formulářové verze (tzv. histogram formulářových záznamů podle verze)
- FP13.2 (C) - systém bude uživateli prezentovat interaktivní graf vybraného uživatelského záznamu s počty odpovědí u jednotlivých formulářových snapshotů v čase (tzv. histogram počtu odpovědí jednoho uživatelského záznamu)
- FP13.3 (C) - systém uživateli poskytne možnost data, která vyhledá pomocí jazyka SPARQL, zobrazit jako graf

5.2.2 Nefunkční požadavky

Nefukční požadavky jsou kategorií požadavků, které se zaměřují především na kvalitu poskytovaných služeb z hlediska celé aplikace, ne přímo na konkrétní funkce. Tyto požadavky jsou v této sekci shrnuty.

NP1 Výkon.

- NP1.1 (S) - systém zvládne zpracovat řádově desítky až stovky snapshotů uživatelských záznamů bez újmy na době odezvy aplikace (při provádění úkonů správy projektových dat)
- NP1.2 (S) - systém umožní souběžné používání webového rozhraní aplikace alespoň 2 uživatelům (při provádění úkonů správy projektových dat)

NP2 Přenositelnost aplikace.

- NP2.1 (M) - všechny části aplikace bude možné snadno přenést do jiného prostředí bez újmy na konzistenci dat
- NP2.2 (M) - moduly aplikace bude možné konfigurovat pomocí externích souborů

NP3 Logování.

- NP3.1 (M) - moduly aplikace budou opatřeny logovacím systémem, logy bude možné procházet externě
- NP3.2 (M) - zejména webové rozhraní bude opatřeno systémem zobrazování chybových hlášek, který pomůže uživateli určit případný chybový stav aplikace

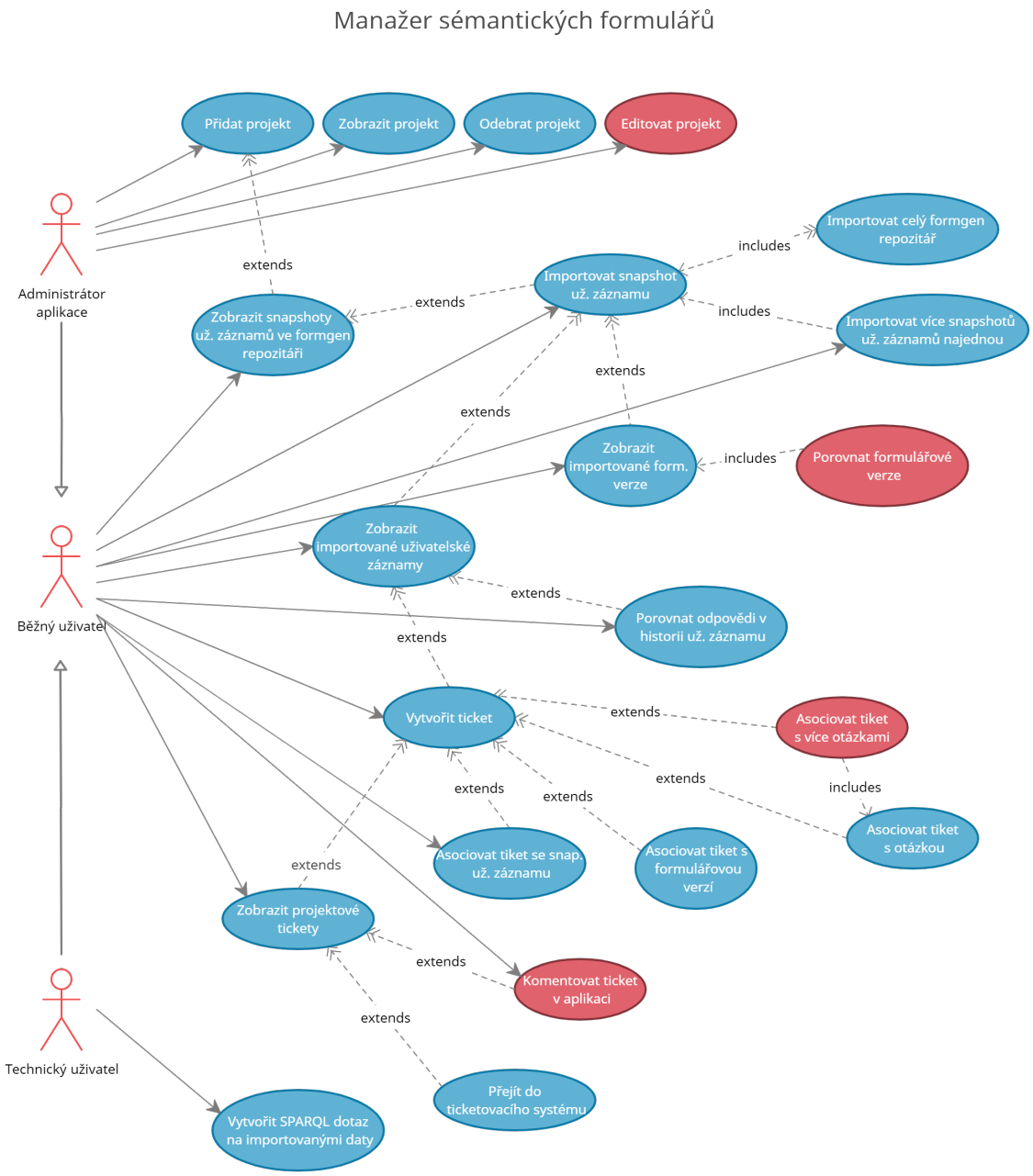
5.3 Use-case diagram

Use-case diagram zachycuje vnější pohled uživatele na modelovaný systém a jeho funkce. Jeho součástí jsou uživatelské role, pro které jsou jednotlivé funkce systému určené. Kromě interakce uživatele se systémem také diagram zachycuje vazby funkcí mezi sebou: vazba *extends* (rozšiřuje jinou funkci) a vazba *includes* (zahrnuje jinou funkci). Kromě toho je zde zavedené barevné odlišení use-casů, kde modrá barva značí standardní prioritu use-casu a červená barva značí, že funkce je mimo plánovaný rozsah, a je již známo, že nebude implementována.

Od systému se neočekává implementace přihlašovacího uživalského systému, který by oddělil jednotlivé funkce systému podle rolí. Nicméně uživatelé, kteří přijdou do kontaktu s aplikací se dají rozdělit do 3 kategorií:

- **Administrátor aplikace** - Zajišťuje nasazení a konfiguraci aplikace, připojení do jednotlivých projektových databází a tiketovacího systému.
- **Běžný uživatel** - Uživatel, který se zajímá především o základní projektová data, jako je počet vedených uživatelských záznamů, počet existujících formulářových verzí apod. Může vytvářet tikety a řídit tak průběh projektu.

- **Technický uživatel** - Uživatel, který je schopen pracovat i s jazykem SPARQL (kapitola 2.4) a dotazovat se tak na pokročilejší informace o importovaných datech.



Obrázek 5.3: Use-case diagram Manažeru sémantických formulářů.

5.4 Uživatelské scénaře

Uživatelské scénáře v této kapitole popisují interakce uživatele systému s webovým rozhraním aplikace. Cílem těchto scénářů je ukázat kontext a smysl jednotlivých funkcí systému, popsáných v požadavcích aplikace v kapitole 5.2.

5.4.1 Vytvoření projektu a import dat

Popis scénáře: Vytvořit projekt ve webovém rozhraní, procházet data ve vzdáleném formgen repozitáři. Importovat a zobrazit snapshot uživatelského záznamu.

Uživatel: Administrátor aplikace.
Počáteční stav aplikace: Aplikace neobsahuje žádná data. Uživatel se nachází na úvodní stránce aplikace.
Očekávaný výsledek: Aplikace obsahuje 1 importovaný uživatelský snapshot.

Průchod scénářem:

1. Uživatel vytvoří nový projekt v sekci projektů.
2. Uživatel přejde do sekce pro nahlížení vzdáleného formgen repozitáře.
3. Uživatel si v seznamu zobrazených položek vybere jeden záznam a ten importuje do systému pomocí tlačítka.
4. Uživatel přejde do sekce importovaných dat a v seznamu uživatelských záznamů najde 1 importovanou položku. V historii této položky najde 1 snapshot uživatelského záznamu.
5. Uživatel zobrazí snapshot uživatelského záznamu pomocí tlačítka jako SForms formulář.

5.4.2 Analýza stavu probíhajícího projektu

Popis scénáře: Uživatel chce zjistit, jaký je aktuální stav projektu z hlediska počtu respondetů u jeho formulářů a kolikrát se již změnila podoba vyplňovaného formuláře.

Uživatel: Běžný uživatel.
Počáteční stav aplikace: V aplikaci je vytvořený 1 projekt. Uživatel se nachází na úvodní stránce aplikace.
Očekávaný výsledek: Aplikace obsahuje všechny záznamy, které lze importovat z projektového formgen repozitáře. Uživatel zná počet respondetů, kteří vyplnili formulář, a počet formulářových verzí, které byly identifikovány.

Průchod scénářem:

1. Uživatel přejde do sekce pro nahlížení vzdáleného formgen repozitáře.
2. Uživatel si vybere možnost importovat všechna data ve vzdáleném formgen repozitáři.
3. Uživatel podle zobrazeného počtu importovatelných a již importovaných záznamů zjistí, zda již import úspěšně doběhl.
4. Po dokončení importu všech záznamů přejde uživatel do sekce přehledu importovaných dat.
5. Z informací na této stránce uživatel zjistí počet importovaných uživatelských záznamů (tedy počet respondentů) a počet zpracovaných formulářových verzí.

5.4.3 Zobrazení historie vyplňování formuláře

Popis scénáře: Uživatel zjišťuje, jak jeden konkrétní respondent postupoval při vyplňování formuláře.

Uživatel: Běžný uživatel.
Počáteční stav aplikace: V aplikaci je vytvořený 1 projekt a všechna data

z tohoto projektu jsou importovaná. V projektu existuje alespoň jeden uživatelský záznam, jehož historii tvoří více než jeden snapshot.
Očekávaný výsledek: Uživatel zná počet zobrazení formuláře a počet mezi-uložení formuláře vybraným respondentem.

Průchod scénářem:

1. Uživatel přejde do sekce pro přehled importovaných dat.
2. Uživatel si vybere jeden uživatelský záznam, jehož historii tvoří více jak jeden snapshot, a zobrazí jeho historii.
3. Uživatel zobrazí první položku v seznamu pomocí SForms, a zjistí tak stav formuláře po prvním zobrazení, resp. uložení.
4. Uživatel použije funkci "porovnání odpovědí dvou po sobě jdoucích snapshotů". Aplikace uživateli zobrazí odpovědi, které v jednom snapshotu chybí, a v druhém existují a které otázky byly změněny.
5. Uživatel použije tuto funkci i pro všechny ostatní po sobě jdoucí snapshoty. Díky tomu uživatel zjišťuje, které odpovědi byly respondentem postupně přidávány či odebírány.

5.4.4 Vytvoření tiketu

Popis scénáře: Uživatel vytvoří tiket, asociuje ho s formulářovou verzí a tento tiket zobrazí ve webovém rozhraní tiketovacího systému.

Uživatel: Běžný uživatel.
Počáteční stav aplikace: V aplikaci je vytvořený 1 projekt. Do aplikace je importovaný 1 snapshot uživatelského záznamu a pro projekt nejsou evidovány žádné tikety.
Očekávaný výsledek: V aplikaci je vytvořený jeden tiket, který je asociovaný s importovanou formulářovou verzí. Uživatel se na konci scénáře nachází ve webovém rozhraní tiketovacího systému.

Průchod scénářem:

1. Uživatel přejde do sekce pro přehled importovaných dat.
2. Uživatel zobrazí seznam identifikovaných formulářových verzí, kde se nachází pouze 1 položka.
3. Uživatel zobrazí detail formulářové verze.
4. Uživatel pomocí formuláře vytvoří tiket s libovolným jménem a popisem. Při vytváření tiketu zaškrtne, že tiket se má asociovat s formulářovou verzí.
5. Po zobrazení hlášky o úspěšném vytvoření tiketu uživatel přejde na přehled projektových tiketů, kde se nachází pouze jeden čerstvě vytvořený tiket.
6. Uživatel klikne na link, který je součástí popisu zobrazeného tiketu. Tento link ho přesměruje do webového rozhraní tiketovovacího systému.

5.4.5 Vytvoření analytického SPARQL dotazu

Popis scénáře: Uživatel vytvoří a zavolá analytický SPARQL dotaz. Použije k tomu sadu přednastavených dotazů a komponentu pro výběr formulářové otázky. Uživateli se zobrazí výsledek dotazu provedeného nad importovanými daty.

Uživatel: Technický uživatel.
Počáteční stav aplikace: V aplikaci je vytvořený 1 projekt a data z tohoto projektu jsou importovaná.
Očekávaný výsledek: Uživateli se zobrazí výsledek SPARQL dotazu jako tabulka.

Průchod scénářem:

1. Uživatel přejde do sekce pro vyhledávání v importovaných datech.
2. Uživatel si zvolí entity, pro které bude vytvářet analytický dotaz. Strukturu a názvy vlastností entit vyčte z pomocných přednastavených dotazů.
3. Uživatel si vybere některý z přednastavených dotazů, který může upravit. V tomto bodě zohlodnění minulý krok, při kterém měl možnost si prohlédnout strukturu a názvy vlastností k tomu, aby nyní upravil dotazs podle vlastních potřeb.

4. Pracuje-li vybraný dotaz s šablonou otázky, jejím snapshotem nebo jejím textovým zněním, může uživatel použít komponentu pro nalezení formulářové otázky. Komponenta pro nalezení formulářové otázky zobrazí identifikátory *question-origin*, *question-origin-path* a textové znění otázky. Tuto informaci pak uživatel může použít při tvorbě analytického dotazu.

5. Uživatel pomocí tlačítka zavolá provedení analytického dotazu. Aplikace zobrazí výsledek tohoto dotazu jako tabulku.

Kapitola 6

Architektura aplikace a technologie

Tato kapitola popisuje architekturu aplikace a technologie, které jsou při jejím budování použité. Jsou zde popsány externí systémy, ke kterým se aplikace připojuje.

Modelovou architekturou pro Manažer sémantických formulářů je Client-Server architektura.

6.1 Serverová část aplikace

Tato sekce poskytuje high-level pohled na architekturu serverové části aplikace. Popisuje jednak návrhový vzor, podle něhož se řídí, ale také některé klíčové technologie, ze kterých také architektura vychází. Konkrétní popis použitých technologií je však součástí kapitoly 6.1.2. Kromě toho je zde seznam externích systémů, ke kterým se serverová část aplikace připojuje.

Serverová část Manažeru sémantických formulářů je 3 vrstvá Java aplikace běžící na frameworku Spring Framework. Aplikace používá řadu technologií Sémantického webu, jako je např. RDF databáze pro persistenci dat. Pro entintí model je v aplikaci zavedena vlastní ontologie, pomocí níž je možné pracovat s daty prostřednictvím dotazovacího jazyka SPARQL (kapitola 2.4).

Použitým návrhovým vzorem pro design aplikace je vícevrstvá architektura [43], jejíž jednotlivé vrstvy jsou zde vysvětleny. Jedná se o standardní architekturu, jejímiž součástmi jsou prezenční, servisní, persistenční a databázová vrstva.

Prezenční vrstva. Vrstva, která se zabývá pouze prezentací interních dat vnějšímu světu. Tato vrstva je realizována pomocí sady tzv. Controllerů [44], jejichž součástí je implementace REST rozhraní [17].

Servisní vrstva. Cílem této vrstvy je zprostředkovat oboustrannou komunikaci mezi persistenční a prezenční vrstvou, případně navazovat komunikaci s externími systémy, jako je tiketovací systém nebo služba FormGen (popsaná v kapitole 4.3.2).

V rámci zprostředkování komunikace mezi jednotlivými vrstavmi či dalšími systémy tato vrstva aplikuje byznys logiku na data, se kterými pracuje. Byznys logikou se zde rozumí sada pravidel, která vycházejí z funkčních požadavků aplikace. Těmito požadavky jsou např. identifikace a evidence formulářových verzí a dalších modelových entit z importovaných dat.

Persistenční vrstva. Tato vrstva slouží jako vstupní bod pro komunikaci aplikace s databází. Data pomocí frameworku pro persistenci dat (implementováno pomocí JOPA frameworku [45]) převádí na databázové záznamy. Stejně tak databázové záznamy převádí na aplikační data.

Databázová vrstva. Je interním uložištěm dat, která aplikace používá. Realizována je zde pomocí RDF databáze, se kterou komunikuje framework JOPA.

6.1.1 Externí systémy

Externí systémy, ke kterým se serverová část aplikace připojuje, jsou:

- Služba FormGen (popsaná v kapitole 4.3.2)
- FormGen repozitář - RDF databáze
- Trello REST API (popsané v kapitole 3.1)

6.1.2 Technologie serverové části aplikace

Tato podsekce poskytuje přehled technologií použitých pro realizaci serverové části aplikace včetně popisu a verzí těchto technologií.

- **Java 8** [41] - Standardní technologie pro implementaci serverové části aplikace. Verze 8 je zde použita z důvodu kompatibility s některými knihovnami pro práci s ontologiemi.
- **Spring Framework 5.2.6** [46] - Aplikační framework pro vývoj enterprise aplikací. Jeho součástí je např. implementace Dependency Injection principů [47], logika pro REST API a další.
- **Maven 4.0.0** [1] - Buildovací systém.
- **JOPA 0.14.4** [45] - Framework pro persistenci sémantických dat a RDF grafů.
- **Podpůrné knihovny projektu RDF4J - verze 3.2.1** - Sada knihoven pro práci s repozitáři RDF4J.
- **RDF4J Server 2.5.5** - Tato technologie slouží jako server pro umisťování RDF4J repozitářů (RDF databáze). Implementace serverové části aplikace by na verzi této technologie měla být nezávislá. Nicméně doporučenou verzí v kombinaci s podpůrnými knihovnami ve verzi 3.2.1 je RDF4J Server 2.5.5 [2].

6.2 Klientská část aplikace

Klientská část aplikace je realizována jako JavaScript single-page webová aplikace pomocí níž uživatel interaguje se systémem. Klientská část komunikuje se serverovou části pomocí REST API.

6.2.1 Technologie klientské části aplikace

Tato podsekce poskytuje přehled technologií použitých pro realizaci klientské části aplikace včetně popisu a verzí těchto technologií.

[1] https://maven.apache.org/
[2] https://rdf4j.org/news/2019/10/13/rdf4j-2.5.5-released/

- **EcmaScript 6** - Skriptovací jazyk vhodný pro realizaci klientských aplikací.
- **React 16** - JavaScriptová knihovna pro vývoj webových aplikací vyvíjená Facebookem. Zajišťuje single-page vlastnosti aplikace.
- **React-Bootstrap 1.4** - Knihovna pro integraci sady nástrojů kaskádových stylů Bootstrap do React aplikace.
- **SForms 0.2.1** - Knihovna pro zobrazování sémantických formulářů (analyzována v kapitole 4).
- **Redux 4.0.5** - Knihovna pro správu stavu aplikace.
- **Chart.js** - Knihovna pro zobrazení grafů v React aplikaci.

Kapitola 7

Implementace

Tato kapitola poskytuje přehled implementovaných funkcí v Manažeru sémantických formulářů. V první části kapitoly je popsáno rozdělení uživatelského rozhraní do sekcí. Hlavním obsahem této kapitoly je však mapování implementovaných funkcí na požadavky aplikace z kapitoly 5.2. U požadavků je uvedeno, jestli jsou implementovány zcela, částečně nebo vůbec.

7.1 Struktura uživatelského rozhraní

Uživatelské rozhraní je rozděleno do 6 sekcí. Většina z těchto sekcí je zpřístupněna až poté, co je v aplikaci zaevidován alespoň jeden projekt, viz FP1. Následující seznam uvadí sekce dostupné v uživatelském rozhraní, a to včetně českého překladu:

- **Homepage** - *Úvodní strana.* Neobsahuje žádné klíčové informace.
- **Projects** - *Správa projektů.* Sekce pro správu projektů, kde lze přidávat či odebírat projekty.
- **Browse in Imported data** - *Přehled importovaných dat.* Sekce přehledu importovaných dat, kde je možné procházet a zobrazovat importované záznamy.
- **Search in Imported data** - *Hledání v importovaných datech.* Sekce pro vytváření analytických SPARQL dotazů.

- **Browse in Tickets** - *Přehled tiketů.* Přehled tiketů asociovaných s konkrétním projektem.
- **Browse in Remote data** - *Přehled vzdálených dat.* Sekce pro procházení formulářových záznamů vzdáleného formgen repozitáře.

7.2 Funkční požadavky

Tato sekce prochází jednotlivé skupiny funkčních požadavků z kapitoly 5.2.1, a pro každou tuto skupinu úvadí popis implementovaného řešení, tedy v jaké míře byly požadavky naplněny.

FP1 Správa projektu.

Implementované požadavky: FP1.1, FP1.2, FP1.5
Neimplementované požadavky: FP1.4
Částečně implementované požadavky: FP1.3

V uživatelském rozhraní je možné zobrazit, přidávat a odebírat tzv. projekty. Evidované projekty jsou součástí navigačních menu v horní části rozhraní. Limitací požadavku FP1.3 je, že při odebrání projektu je z databáze smazán pouze záznam samotného projektu, ale ne všechna projektová data. V případě potřeby odstranění všech projektových dat je nutné provést vymazání ručně z databáze. Na druhou stranu, pokud jde uživateli pouze o změnu údajů projektu, lze projekt smazat a znovu vytvořit se stejným jménem.

Definovatelné vlastnosti projektu jsou následující:

- **FormGen Repository URL** - Url vzdáleného formGen repozitáře. Položky v tomto repozitáři (formulářové záznamy) je po přidání projektu možné procházet v sekci *Přehled vzdálených dat.*
- **Service URL** - Url Služby FormGen, která pomocí app a formgen repozitářů generuje SForms formulář.
- **App Repository URL** - Url app repozitáře.
- **Record Recognition SPARQL** - Tento parametr umožňuje specifikovat SPARQL dotaz, který ve vzdáleném formgen repozitáři (při importu

snapshotů uživatelských záznamů) vybírá klíčové vlastnosti záznamu. Pomocí těchto vlastností se např. identifikuje uživatelský záznam, ke kterému patří importovaný snapshot. Vyplnění pole je volitelné, avšak pokud uživatel využije této možnosti, jména výsledných proměnných musí odpovídat přednastaveným názvům. Těmito vlastnostmi jsou:

- *recordCreateDate* - datum vytvoření záznamu
- *recordModifiedDate* - datum úpravy (zobrazení) záznamu
- *remoteRecordURI* - URI uživatelského záznamu
- *rootQuestionOrigin* - identifikátor *question-origin* kořenové otázky formulářového záznamu (pokud je přitomná)

Defaultní podoba dotazu, který se použije, pokud uživatel nezadá parametr projektu *Record Recognition SPARQL*, je možné vidět ve zdrojovém kódu 7.1.

```
PREFIX form: <http://onto.fel.cvut.cz/ontologies/form/>
PREFIX sm: <http://vfn.cz/ontologies/study-manager/>
PREFIX dc: <http://purl.org/dc/terms/>

SELECT ?recordCreateDate ?recordModifiedDate ?remoteRecordURI
 ↪  ?rootQuestionOrigin
WHERE {
  graph ?contextUri {
    ?s a sm:patient-record .
      ?s dc:created ?recordCreateDate .
      OPTIONAL {?s dc:modified ?recordModifiedDateOptional .}
      OPTIONAL {
        ?s sm:has-question> ?question .
        ?question form:has-question-origin
         ↪  ?rootQuestionOrigin .
      }
      BIND(
        COALESCE(
          ?recordModifiedDateOptional,?recordCreateDate
        ) as ?recordModifiedDate
      )
      BIND (str(?s) as ?remoteRecordURI)
  }
}
```

Zdrojový kód 7.1: Defaultní podoba dotazu Record Recognition SPARQL.

FP2 Import projektových dat.

Implementované požadavky: FP2.1, FP2.2, FP2.3, FP2.4, FP2.5
Neimplementované požadavky: -

Import snapshotů uživatelských záznamů je možné provést v sekci pro procházení vzdálených dat. Tato sekce obsahuje stránkovací mechanismus pro zobrazení položek v projektovém formgen repozitáři. Každá z těchto položek reprezentuje jeden snapshot uživatelského záznamu, který je možné pomocí tlačítka zobrazit jako SForms formulář nebo importovat do systému. Zároveň je možné pomocí uživatelského rozhraní importovat záznamy dávkově, a to v množství 10, 200 nebo všechny neimportované záznamy najednou. Součástí rozhraní je také informace o tom, kolik je ve vzdáleném formgen repozitáři celkově importovatelných záznamů a kolik z nich je již v systému importováno.

Při importu záznamu se v první řadě volá Služba FormGen (popsána v kapitole 4.3.2). Odpovědí tohoto volání je JSON-LD dokument s formulářovými daty vybraného snapshotu uživatelského záznamu. Tato data jsou uložena do vlastní databáze a provádí se na nich identifikace uživatelského záznamu, fomulářových otázek, formulářové šablony, jejich verze atd. Dávkové spouštění importu probíhá sekvenčně, tedy systém spouští import záznamů jeden po druhém.

FP3 Zobrazení formulářových záznamů pomocí SForms.

Implementované požadavky: FP3.1, FP3.2
Neimplementované požadavky: -

Formulářový záznam lze zobrazit v sekci pro procházení vzdálených dat a také v sekci importovaných dat. Požadavky jsou tedy splněné beze zbytku.

FP4 Identifikace formulářové struktury.

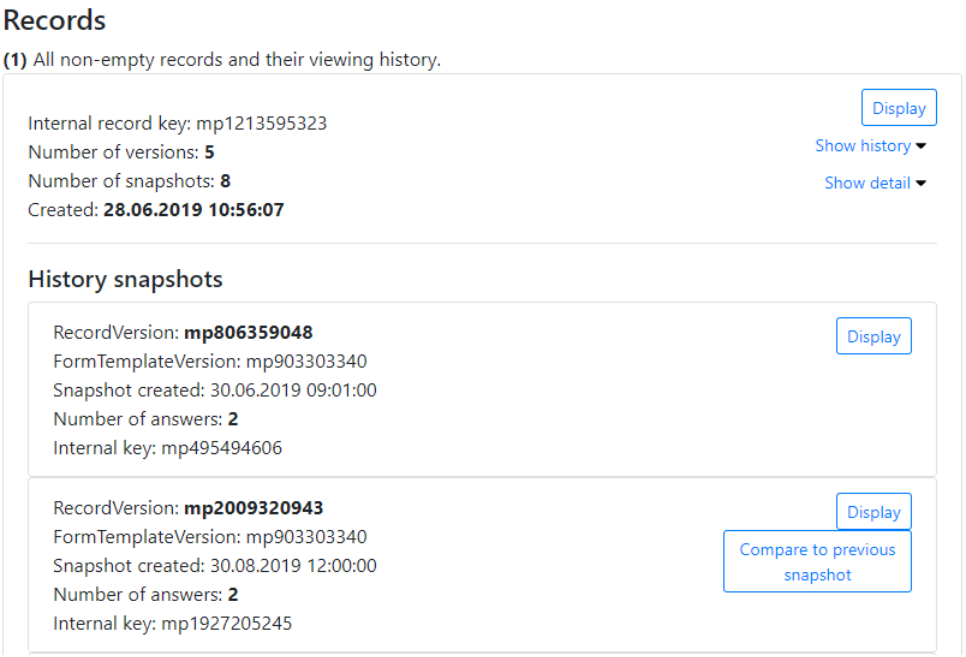
Implementované požadavky: FP4.1, FP4.2, FP4.3, FP4.4, FP4.5, FP4.6
Neimplementované požadavky: FP4.7

Identifikace formulářové struktury probíhá během importu záznamů z projektového formgen repozitáře. Struktura formulářových otázek určuje verzi šablony formuláře, která je přiřazena snapshotu uživatelského záznamu.

Algoritmus, který zpracovává strukturu otázek a určuje verzi formuláře, pracuje jednoznačně, a tedy pro stejný formulář vždy vygeneruje stejný výsledek. Algoritmus nicméně negeneruje naprosto unikatní výsledky, a v omezeném množství případů se může stát, že vygeneruje pro dva formuláře s různou strukturou otázek stejnou verzi. Tento případ může nastat ve chvíli, kdy dva přímí potomci (otázky) nějaké nadotázky změní svoji pozici bez změny identifikátoru. Algoritmus nebere v úvahu pozici otázky v rámci jedné nadotázky - při zpracování stromu totiž každou sadu podotázek seřadí podle identifikátorů *question-origin*.

Každá šablona verze formuláře disponuje identifikátorem snapshotu uživatelského záznamu, při jehož zpracování byla verze objevena. Tento snapshot se používá pro zobrazení formulářové verze pomocí SForms, a to tehdy, když si tuto možnost uživatel zvolí v přehledu importovaných dat (v seznamu všech identifikovaných formulářových verzích).

Při zobrazení seznamu formulářových verzí (v sekci přehledu importovaných dat) se uživateli zobrazí také počet těchto verzí. Na detailu každé verze je možné upravit interní pojmenování formulářové verze, což se hned na uživatelském rozhraní projeví. Toto interní pojmenování se používá např. v histogramu verzí z požadavku FP13.1.



Obrázek 7.1: Ukázka zobrazení importovaného uživatelského záznamu.

FP5 Identifikace uživatelského záznamu.

Implementované požadavky: FP5.1, FP5.2, FP5.3
Neimplementované požadavky: FP5.4

Identifikace uživatelského záznamu probíhá při importu dat z formgen repozitáře. Proces identifikace záznamu lze ovlivnit projektovou vlastností s názvem *Record Recognition SPARQL*. Touto vlastností je SPARQL dotaz, pomocí něhož se identifikují vlastnosti snapshotu uživatelského záznamu ve vzdáleném formgen repozitáři. Příslušné vlastnosti jsou shrnuté v popisu implementace požadavku (sekce 7.2). Unikátní identifikátor uživatelského záznamu se vytváří z kombinace URI vyžádaného záznamu a času jeho vytvoření.

Uživatelské záznamy lze procházet v sekci přehledu importovaných dat. Pro každý uživatelský záznam lze zobrazit historii snapshotů (obrázek 7.1. Informace o uživatelských záznamech, které jsou dostupné ve webovém rozhraní jsou:

- unikátní identifikátor záznamu
- počet verzí uživatelského záznamu v historii
- počet snapshotů v historii
- datum a čas vytvoření záznamu
- interní URI záznamu

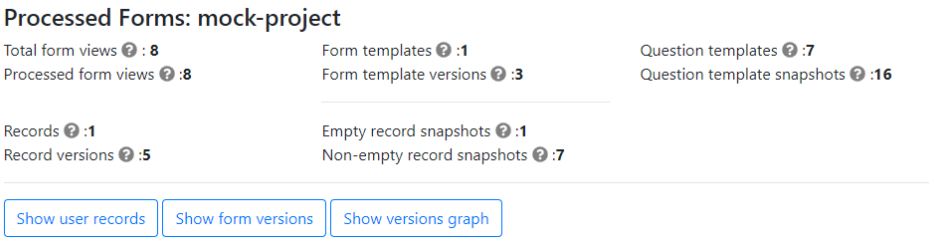
Informace zobrazitelné pro snapshoty uživatelských záznamů ve webovém rozhraní jsou:

- identifikátor verze uživatelského záznamu
- identifikátor formulářové verze
- datum a čas vytvoření snapshotu
- počet evidovaných odpovědí ve snapshotu
- unikátní identifikátor snapshotu

FP6 Přehled importovaných dat.

Implementované požadavky: FP6.1
Neimplementované požadavky: -

Tento požadavek je splněn beze zbytku. Přehled četnosti klíčových entit v systému je dostupný v přehledu importovaných dat a v sekci pro vyhledávání v projektových datech, a to včetně vysvětlení významu jednotlivých entit. Je možné ho vidět na obrázku **??**.



Obrázek 7.2: Náhled projektových statistik v aplikaci.

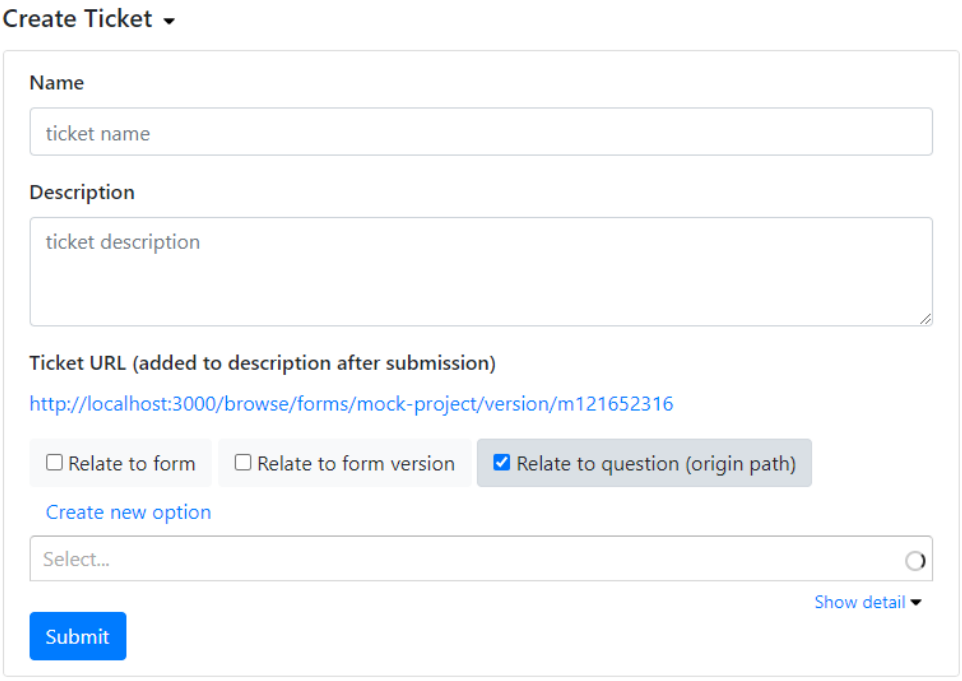
FP7 Cachování uživatelských záznamů.

Implementované požadavky: -
Neimplementované požadavky: -
Částečně implementované požadavky: FP7.1

Cachování uživatelských záznamu je v systému implementováno pouze částečně. Systém při importu dat ze vzdáleného formgen repozitáře ukládá čtená data do interní databáze. Identifikace uživatelských záznamů, formulářových verzí apod. probíhá nad touto lokální kopii, a pro stejná data tedy není potřeba dotazovat se vzdálených repozitářů. Nicméně systém není schopný generovat JSON-LD dokument v takovém formátu, aby byl v klientské části zpracovatelný pomocí knihovny SForms, a tedy aby šel zobrazit. Z tohoto důvodu, chce-li uživatel zobrazit formulářová data, musí k tomu vždy znovu použít vzdálenou Službu FormGen, která vytváří JSON-LD reprentaci formuláře vždy novou.

FP8 Integrace tiketovacího nástroje - vytvoření tiketu.

Implementované požadavky: FP8.1, FP8.2, FP8.3, FP8.4, FP8.5, FP8.7, FP8.8



Obrázek 7.3: Ukázka formuláře pro vytvoření tiketu v aplikaci.

Neimplementované požadavky: FP8.6

Jako tiketovací systém pro Manažer sémantických formulářů slouží Trello, viz kapitola 3.3. Uživatel může vytvořit Trello tiket přímo v rozhraní webové aplikace. K formuláři pro vytvoření nového tiketu vede cesta skrz zobrazení snapshotu uživatelského záznamu nebo skrz zobrazení formulářové verze. V sekci přehledu importovaných dat uživatel zobrazí SForms formulář pomocí nabídky u zvolené položky. Kromě SForms formuláře se však uživateli otevře také přehled tiketů spojených s vybraným snapshotem uživatelského záznamu a formulář pro vytvoření tiketu (obrázek 7.3).

Informace, které může uživatel o tiketu vyplnit, jsou *jméno* a *popis*. Mimo to si uživatel může zvolit, jestli asociuje tiket se snapshotem uživatelského zánamu, formulářovou verzí nebo s některou z otázek formuláře. S projektem je tiket asociován automaticky. Pro výběr formulářové otázky je rozhraní opatřeno chytrým vyhledávačem otázek, který je realizovaný pomocí knihovny Intelligent Tree Select [1]. Součástí rozhraní je také odkaz, který se mění v závislosti na tom, jestli je tiket asociovaný s pouhým snapshotem nebo formulářovou verzí. Link je po vytvoření tiketu automaticky přidán do popisu tiketu, a je možné pomocí něj zobrazit asociovaný záznam. Přičemž pokud je tiket spojený s uživatelským snapshotem i formulářovou verzí, aplikuje se odkaz pro uživatelský snapshot.

[1] `https://www.npmjs.com/package/intelligent-tree-select`

Po odeslání požadavku na vytvoření tiketu se uživateli zobrazí hláška o úspěšném vytvoření, jejíž součástí je odkaz do webové rozhraní samotného tiketovacího systému a detailu tiketu.

Pro asociaci tiketu s projektovými entitami je ve webovém rozhraní Trello nutné aktivovat bezplatný doplněk *Vlastní pole* [2]. V rámci tohoto doplňku je nutné přidat následující pole:

- **SpecificFormVersionKEY** - pro asociaci tiketu s formulářovou verzí
- **SpecificFormCU** - pro asociaci tiketu se snapshotem uživatelského záznamu
- **SpecificQuestionQOP** - pro asociaci tiketu s konkrétní otázkou formuláře

Nově vytvořený tiket se automaticky vkládá do prvního sloupce, který je na Trello nástěnce vedený jako první. Nicméně to, do jakého sloupce se má tiket ukládat, lze konfigurovat v serverové části aplikace pomocí *ID* nebo *jména* sloupce.

Tiket vytvořený přímo v Manažeru sémantických formulářů je automaticky opatřen tzv. štítkem, údávajícím příslušnost k projektu.

FP9 Integrace tiketovacího nástroje - zobrazení tiketů.

Implementované požadavky: FP9.1, FP9.2, FP9.3, FP9.4, FP9.6, FP9.7
Neimplementované požadavky: FP9.5,FP9.8,FP9.9,FP9.10,FP9.11

Existující tikety lze procházet v sekci přehledu projektových tiketů nebo v sekci přehledu importovaných dat.

Sekce přehledu projektových tiketů je vymezená pouze pro zobrazení tiketů, které jsou asociovány s vybraným projektem. Součástí zobrazení tiketů jsou zde všechny informace, které uživatel zadává při vytváření tiketu. Je zde *jméno*, *popis*, *odkaz do Trello* (odkaz vede přímo na detail tiketu) a *odkaz pro zobrazení asiované entity*. Zároveň jsou součástí zobrazení tiketu identifikátory asociovaných entit, jako je např. unikátní klíč pro identifikaci formulářové verze, context URI snapshostu uživatelského záznamu nebo identifikátor otázky *question-origin-path*.

Tikety v sekci přehledu importovaných dat uživatel vidí přímo nad formulářem pro vytváření tiketů. Je zde možnost zobrazit všechny tikety, které jsou

[2] `https://help.trello.com/article/1067-using-the-custom-fields-power-up`

asociované se snapshotem právě zobrazeného formuláře, nebo jeho verzí, či s některou z jeho otázek (tyto otázky se mohou vyskytovat i v jiných verzích formuláře). Informace o jednotlivých tiketech jsou zde zobrazeny stejně jako v dedikované sekci pro přehled tiketů.

FP10 Dotazování nad importovanými daty.

Implementované požadavky: FP10.1, FP10.2, FP10.3
Neimplementované požadavky: FP10.5
Částečně implementovné požadavky: FP10.4

V sekci pro vyhledávání v datech je možné provádět analýzu nad importovanými daty pomocí SPARQL dotazů. V této sekci je za účelem usnadnění vytváření dotazů několik pomocných mechanismů. Těmito mechanismy jsou:

- **Vyhledávač formulářových otázek.** Pomocí knihovny Intelligent Tree Select je zde možné fulltextově prohledávat strom formulářových otázek. Součástí tohoto stromu jsou všechny známé snapshoty otázkových šablon, u kterých se bere v úvahu i pozice ve formuláři, resp. identifikátor *question-origin-path*. Při výběru otázky se uživateli zobrazí identifikátory, které lze použít při tvorbě SPARQL dotazů:
  - *question-origin-path*
  - *question-origin*
  - *label*
- **Základní entitní dotazy.** Uživatel může vužít sadu základních přednastavených dotazů, které do SPARQL editoru vloží pomocí tlačítka. Tyto přednastavené dotazy jsou vytvořené tak, aby uživateli především ukázaly strukturu dat a definici jejich vlastností. Při použití a zavolání pomocného dotazu pro entitu RecordSnapshot se tedy v dolní části rozhraní ukáže tabulka všech evidovaných snapshostů uživatelských záznamů, součástí čehož jsou všechny hodnoty vlastnosti těchto dotazů.
- **Pokročilé dotazy.** Systém disponuje sadou přednastavených dotazů, pomocí nichž lze zjišťovat pokročilé informace o rozmanitosti odpovědí u otázek, struktuře formulářů a historii jejich vyplňování, ale neobsahuje otázky pro analýzu specifické interakce uživatele s formulářem.
- **Více editorů.** Uživatel může využít možnost zobrazení až 3 editorů pro vytváření SPARQL dotazů. Přednastavené dotazy jsou opatřeny tlačítkem pro nahrání dotazu do jednoho ze 3 zobrazitelných editorů. Dotazy nad importovanými daty je možné volat z libovolného z těchto editorů. Tato funkce má smysl zejména při práci na složitějších dotazech,

během čehož je nutné buď zobrazovat strukturu projektových entit, nebo ověřovat správnost některých tezí.

FP11 Porovnání dvou formulářových verzí.

Implementované požadavky: -
Neimplementované požadavky: FP11.1

Tento požadavek není implementován. Zamýšlená realizace požadavku je limitována pouze částečnou funkčností cachování formulářových záznamů do interní databáze. Možnou a preferovanou realizací tohoto požadavku je spojení dvou formulářových záznamů do jednoho a zobrazení pomocí SForms, přičemž v tomto formuláři by byly barevně rozlišené otázky, které v jednom formuláři přebývají oproti druhému. Vzhledem k tomu, že aplikace však není schopná generovat JSON-LD dokument ve formátu, který může zpracovat a zobrazit knihovna SForms, není možné vytvořit ani kombinovaný formulář.

FP12 Porovnání odpovědí dvou formulářových záznamů.

Implementované požadavky: FP12.1
Neimplementované požadavky: -

Odpovědi dvou formulářových záznamů je možné porovnat v sekci přehledu importovaných dat. Porovnat odpovědi je možné u uživatelských záznamů, jejichž historii tvoří alespoň dva snapshoty. Toto porovnání je možné provést na detailu uživatelského záznamu pro každé dva po sobě jdoucí snapshoty.

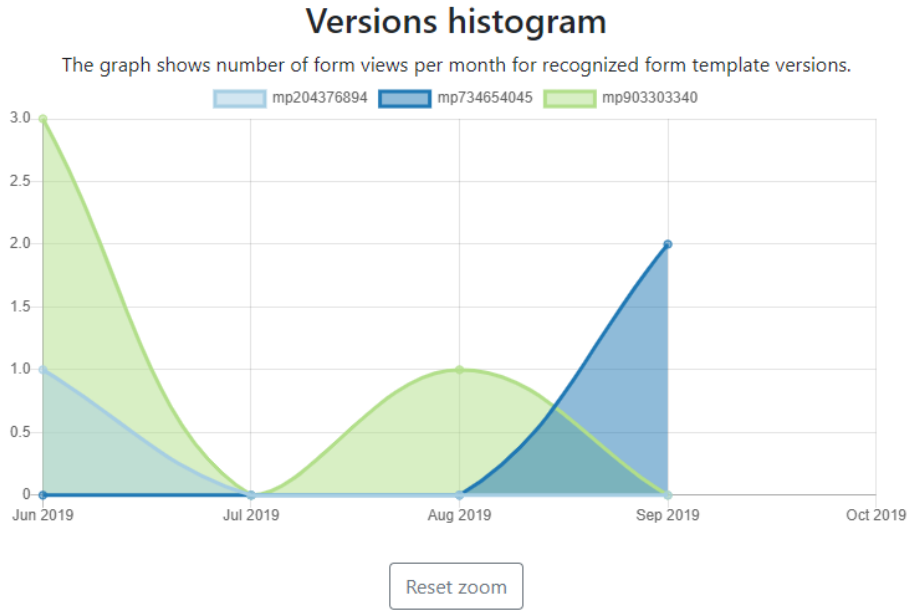
V rámci porovnání odpovědí se zde zohledňuje identifikátor otázek *question-origin*, podle kterého se vyhodnocují zodpovězené otázky. Jsou zde zobrazeny odpovědi, které v jednom záznamu chybí či přebývají nebo se pouze změnily. Je zde zobrazen počet odpovědí, které jsou pro oba formulářové záznamy stejné.

FP13 Zobrazení statistik o importovaných entitách pomocí interaktivních grafů.

Implementované požadavky: FP13.1
Neimplementované požadavky: FP13.2, FP13.3

V sekci přehledu importovaných dat je možné vidět tzv. histogram verzí. Tento histogram zobrazuje počet snapshotů, které jsou evidovány pro každou formulářovou verzi v historii vyplňování projektových dat, tedy od prvního po poslední známé vyplnění formuláře. Data jsou shlukována po měsících a lze tedy z grafu vyčíst např. informaci, kolik importovaných snapshotů uživatelských záznamů pro jednotlivé verze bylo zaevidováno v konkrétním měsíci.

Grafová komponenta umožňuje zobrazovat uživatelem všechny, nebo jen vybrané formulářové verze, či horizontálně přibližovat data (obrázek 7.4).



Obrázek 7.4: Histogram zobrazující 3 verze.

7.2.1 Nefunkční požadavky

Tato sekce prochází jednotlivé skupiny nefunkčních požadavků z kapitoly 5.2.2, a pro každou tuto skupinu uvádí v jaké míře byly požadavky naplněny.

NP1 Výkon.

Implementované požadavky: FP1.1, FP1.2
Neimplementované požadavky: -

Aplikace byla z hlediska výkonu otestována importem dat ze 3 testovacích

projektů s počtem importovaných snapshotů uživatelských záznamů zhruba 10, 120 a 200. Na aplikaci je po importu tohoto objemu dat vidět mírné zpomalení při načítání sekce importovaných dat, a to zejména u uživatelských záznamů a projektových statistik četnosti entit. Zpomalení je také možné pozorovat při vykonávání dotazů v části aplikace pro vyhledávání v datech. Nicméně doba odezvy v této sekci nezávisí na efektivitě implementovaného nástoje, nýbrž na složitosti volaných dotazů.

Z hlediska počtu uživatelů, kteří souběžně používali aplikaci, byl systém otestován v počtu 2. Při výše zmíněných objemech importovaných dat oba uživatelé registrovali mírné zpomalení systému při načítání sekce importovaných dat a zobrazování SForms formulářů.

NP2 Přenositelnost aplikace.

Implementované požadavky: FP2.1, FP2.2
Neimplementované požadavky: -

Z hlediska přenositelnosti byla aplikace během uživatelského testování nasazena v linuxovém a Windows prostředí, včetně přenesení databázových dat. Tento přenos systému neutrpěl žádnou újmu na konzistenci dat či výkonu aplikace.

Aplikaci je možné konfigurovat v klientské i serverové části aplikace pomocí konfiguračních souborů.

NP3 Logování.

Implementované požadavky: FP3.1, FP3.2
Neimplementované požadavky: -

Aplikace je opatřena logovacím systém logback [3], který lze konfigurovat tak, aby zapisoval logy na souborový systém.

Webové rozhraní je opatřeno systémem pro zobrazování informace o úspěchu či chybě při provedení uživatelských akcí, zejména při vyplňování formulářů. Některé chyby v aplikaci však nejsou předem předvídatelné a k jejich bližšímu zkoumání je nutné používat developerské nástroje dostupné ve většině moderních prohlížečů (např. developerská konzole). Tyto developerské nástroje jsou určeny především pro technické uživatele, ale je možné se pomocí nich dopátrat důvodu chybového stavu aplikace.

[3] `http://logback.qos.ch/`

7.3 Konfigurace aplikace

Klientská i serverová část aplikace disponuje možností konfigurovat některá systémová nastavení. Soubory README slouží k tomu, aby některá systémová nastavení vysvětlila.

Pro serverou část aplikace lze prostřednictvím souboru *application.properties* konfigurovat:

- třídu pro databázový driver
- adresu interní databáze
- port, na kterém serverová část aplikace běží
- debugovací mód
- identifikátory pro připojení tiketovacího systému Trello
- *aplikace dále umožňuje konfigurovat všechna dostupná nastavení podle pravidel frameworku Spring Framework*

Pro klientskou část aplikace lze prostřednictvím souboru *.env* konfigurovat následující možnosti:

- adresu pro připojení serverové části aplikace
- port, na kterém klientská část aplikace běží

Kapitola 8

Srovnání implementovaného nástroje a uživatelské testování

Tato kapitola poskytuje srovnání implementovaného nástroje Manažer sémantických formulářů s jedním z již existujících nástrojů, který se zabývá podobnou problematikou. Tímto nástrojem je JotForm, na nemž byla také prováděna rešerse v kapitole 3 existujících řešení .

Dále kapitola popisuje metodiku aplikovanou při uživatelském testování implementovaného nástroje a zobrazuje scénáře pro uživatelské testování. V poslední řadě nabízí přehled chybových nálezů, které odhalilo uživatelské testování a jeho celkové zhodnocení.

8.1 Srovnání s existujícími řešeními

Rešerše existujících řešení nám poskytla široký přehled funkcí, které implementují standardní nástroje pro správu formulářů. Tato rešerše se zúročila především v kapitole návrhu (5) při stanovení funknčích požadavků. Tyto požadavky však také byly ovlivněné specifičností technologií Sémantického webu (kapitola 2.1) a knihovnou SForms (4). Z tohoto důvodu je nyní vhodné učinit srovnání implementovaného nástroje s některým z již existujících nástrojů. Vybraným nástrojem pro srovnání je webová aplikace JotForm (popsaná v kapitole 3.5.1).

Při srovnání webové aplikace JotForm a Manažeru sémantických formulářů je nutné přihlédnout k faktu, že JotForm je aplikací nejen pro správu formulářů, ale také jejich vytváření. Z hlediska správy formulářů tedy následně

poskytuje JotForm mnohem lepší kontext, ve kterém se s formulářovými daty pracuje. Na druhou stranu, Manažer sémantických formulářů se soustředí na poměrně široký use-case zpracování ontologických formulářů. S takovým cílem se pojí řada poměrně vyzývavých problémů, jejichž řešení však vede k poměrně silnému nástroji s širokým použitím.

Uživatelské rozhraní Manažeru sémantických formulářů, narozdíl od JotForm, je na první pohled poměrně stručné - obsahuje pouze klíčové funkce a nepomáhá příliš uživateli se zorientovat v rozhraní aplikace. Zřejmé také je, že data, se kterými implementovaný nástroj pracuje, jsou poměrně technická a pro nezasvěceného člověka poměrně špatně pochopitelná. Obecnou vlastností, která oproti všem ostatním existujícím řešením zde chybí, je možnost přímo pracovat s formulárovými záznamy - např. upravovat jejich údaje nebo je přímo mazat. To však je dáno faktem, že Manažer sémantických formulářů je nástroj pouze pro čtení projektových databází a jeho cílem je primárně data číst a analyzovat.

Přehled formulářových dat je v JotForm realizovaný jako tabulkový editor. Za tímto účelem je v implementovaném nástoji nutné procházet uživatelské záznamy jeden po druhém nebo využít sekci pro vyhledávání v importovaných datech. Na druhou stranu, JotForm např. skoro nepracuje s různými verzemi formulářů (resp. podporuje pouze jednu a v omezené míře dokáže zobrazit historicé verze), zatímto implementovaný manažer poskytuje přehled všech evidovaných formulářových verzí jako jednu z hlavních částí aplikace a obecně s konceptem mnoha formulářových verzí pracuje snadno.

Souhrnné srovnání funkcí Manažeru sémantických formulářů a nástroje JotForm je vidět v tabulce 8.1.

Funkce	**Manažer sémantických formulářů**	**JotForm**
Uživatelský systém	NE	ANO
Import dat z externích zdrojů	ANO	ANO
Možnost vytvářet formuláře	NE	ANO
Souhrnné zobrazení odpovědí ve formulářích	NE	ANO
Zobrazení historie uživatelského záznamu	ANO	NE
Zobrazení historie verzí formuláře	ANO	ANO
Evidence více formulářových šablon	ANO	ANO
Integrace s tiketovacím nástrojem	ANO	ANO
Podpora sémantických formulářů	ANO	NE
Analytický nástroj pro vytváření pokročilých dotazů	ANO	ANO
Popdpora dotazů nad daty v jazyce SPARQL	ANO	NE
Vizualizace vlastní vybrané podmnožiny dat	NE	ANO

Tabulka 8.1: Srovnání funkcí Manažeru sémantických formulářů a aplikace JotForm.

8.2 Uživatelské testování

Tato sekce se věnuje popisu uživatelských testů, které byly za účelem zkvalitnění a zhodnocení výsledků práce vykonány.

8.2.1 Uživatelé pro testování

Za účelem testování Manažeru sémantických formulářů byli vybráni 3 uživatelé s minimálně bakalářským vzděláním v technickém oboru, kteří mají zkušenosti s programováním. Pro všechny tyto uživatele bylo osobně připraveno prostředí pro testování aplikace autorem této práce.

8.2.2 Scénáře uživatelských testů

Manažer sémantických formulářů je poměrně specifickým projektem založeným na ne příliš známé oblasti technologií, a to technologiích Sémantického webu popsaných v kapitole 2.1. Před provedením uživatelských testů bylo tedy nutné seznámit testovací subjekty s alespoň základními znalostmi Sémantického webu, knihovnou SForms (kapitola 4) a principem vytváření sémentických formulářů.

Testovací prostředí a datové uložiště, které se pro uživatelské testy použilo obsahuje 1 uživatelský záznam, jehož historii tvoří 8 snapshotů a 3 různé formulářové verze.

Uživatelský scénář 1: Vytvoření projektu a import dat

Popis scénáře: Uživatel vytvoří projekt a importuje záznamy ze vzdáleného formgen repozitáře.
Počáteční stav aplikace: Uživatel se nachází na úvodní stránce aplikace.

Instrukce pro průchod scénářem:

1. Vyvtvořte nový projekt s následujícími vlastnostmi:
   - **Project Name** - *"libovolné jméno projektu bez mezer"*
   - **FormGen repository URL** - "http://localhost:8090/rdf4j-server/repositories/patients-repository"
   - **Service URL** - "my service URL"
   - **App Repository URL** - "my app repository URL"
   - **Record recognition SPARQL** - *prázdné*
2. Přejděte do sekce pro procházení vzdálených dat pro váš nově vytvořený projekt.
3. Vyberte jednu položku ze vzdáleného repozitáře a tuto položku zobrazte jako SForms formulář. Následně tuto položku importujte do systému.
4. Importujte také všechny ostatní záznamy pomocí dávkového zpracování.
5. Ujistěte se, že jsou všechny položky již importované.

Otázky položené po dokončení scénáře:

1. Jsou již opravdu importované všechny záznamy ve vzdáleném repozitáři?
2. Kolik je ve vašem projektu importováno snapshotů uživatelských záznamů?
3. Kolik je ve vzdáleném repozitáři celkem importovatelných záznamů?

Uživatelský scénář 2: Zobrazení historie vyplňování formuláře

Popis scénáře: Uživatel zjišťuje, jak jeden konkrétní respondent postupoval při vyplňování formuláře.

Počáteční stav aplikace: V aplikace je vytvořený alespoň 1 projekt a všechna data z tohoto projektu jsou importovaná. V projektu existuje 1 uživatelský záznam, jehož historii tvoří 8 snapshotů a 3 různé formulářové verze.

Instrukce pro průchod scénářem:

1. Přejděte do přehledu importovaných dat vámi vytvořeného projektu.
2. Zobrazte detail a historii jediného importovaného uživatelského záznamu ve vašem projektu.
3. Porovnejte odpovědi prvních dvou snapshotů daného uživatelského záznamu.
4. Porovnejte odpovědi druhého a třetího snapshotu daného uživatelského záznamu.
5. Zobrazte čtvrtý snapshot daného uživatelského záznamu jako SForms formulář.

Otázky položené po dokončení scénáře:

1. Kolik je v projektu importováno uživatelských záznamů?
2. Kolik je v projektu identifikováno formulářových verzí?
3. Kolik formulářových verzí čítá historie uživatelského záznamu ze scénáře?
4. Kolik zhruba času zabralo uživateli vyplňování formuláře?
5. Změnil někdy uživatel některou ze svých odpovědí?

Uživatelský scénář 3: Vytvoření a zobrazení tiketu

Popis scénáře: Uživatel vytvoří tiket, asociuje ho s formulářovou verzí a tento tiket zobrazí ve webovém rozhraní tiketovacího systému. Následně rozbrazí formulářovou verzi zpět v rozhraní Manažeru sémantického formuláře.

Počáteční stav aplikace: V aplikaci je vytvořený 1 projekt. Do aplikace je importovaný 1 snapshot uživatelského záznamu a pro projekt je evidováno 5 tiketů.

Průchod scénářem:

1. Přejděte do sekce pro přehled importovaných dat vašeho projektu.

2. Zobrazte přehled evidovaných formulářových verzí.
3. Zobrazte formulářovou verzi jako SForms.
4. Vytvořte tiket s následujícími vlastnostmi:
   - **jméno** - "Tiket 1"
   - **popis** - "můj popis"
   - asociujte tiket s formulářovou verzí
5. Zobrazte tiket v systému Trello.
6. Z webového rozhraní Trello přejděte na detail formulářové verze.
7. Zobrazte nově vytvořený tiket v seznamu tiketů spojených s vybranou formulářovou verzí.

Otázky položené po dokončení scénáře:

1. Kolik tiketů je asociováno se snapshotem uživatelského záznamu, který reprezentuje ve scénáři použitou formulářovou verzi?
2. Kolik tiketů je asociováno s touto formulářovou verzí?
3. Kolik tiketů je asociováno s některou z otázek z dané formulářové verze?

8.2.3 Nálezy

Nález 1 - Nedostatečná informovanost uživatele při dávkovém importu.
Vážnost nálezu: Střední
Popis nálezu: Po zavolání dávkového importu uživatel neví, jestli import stále probíhá nebo ne. Pro zjištění počtu zatím zpracovaných záznamů musí obnovit stránku.

Nález 2 - Opakovaný import stejného záznamu způsobí duplicitu dat.
Vážnost nálezu: Vysoká
Popis nálezu: Po zavolání dávkového importu uživatel neví, jestli import stále probíhá nebo ne. Pro zjištění počtu zatím zpracovaných záznamů musí obnovit stránku.

Nález 3 - Informace o verzi na detailu uživatelského záznamu je matoucí.
Vážnost nálezu: Nízká
Popis nálezu: Při zobrazení uživatelského záznamu se uživatelé domnívají, že informace "Number of versions"hovoří o počtu formulářových verzí a ne o počtu verzí uživatelského záznamu.

Nález 4 - Uživatel musí zobrazit SForms proto, aby mohl vytvořit tiket.
Vážnost nálezu: Střední
Popis nálezu: Uživatel proto, aby mohl vytvořit nebo zobrazit tiket v uživatelském rozhraní, musí nejprve zobrazit formulář SForms a potom teprve může přejít k sekci tiketů. To je neintuitivní.

Nález 5 - Ze sekce projektu nelze přejít do stejné sekce jiného projektu.
Vážnost nálezu: Vysoká
Popis nálezu: Chce-li uživatel přejít z jedné sekce projektu do stejné sekce jiného projektu, název projektu na stránce se změní, avšak data zůstavají stejná. Pro to, aby uživatel přešel do stejné sekce jiného projektu musí nejprve vstoupit do jiné sekce kteréhokoliv projektu a potom teprve do původně zamýšlené sekce.

8.2.4 Vyhodnocení uživatelského testování

Dotázaní uživatelé se v názoru na aplikaci a její použitelnost mírně liší. Převládá názor, že aplikaci je možné používat ke správě sémantických formulářů, ale pro její používání musí uživatel poměrně dobře znát koncept těchto formulářů a musí rozumnět struktuře dat, která v jsou do aplikace importována. Jinak řečeno, aplikace je poměrně technicky založená a pro její používaní musí znát uživatel mnoho interních výrazů a význam jednotlivých projektových entit.

Podle jednoho z responentů aplikace neobsahuje dostatek systémových upozornění, která by uživatele informovala o stavu probíhajících procesů apod. (např. dávkový import dat).

Přínosným prvkem aplikace však podle všech respondentů byl interaktivní histogram verzí, který je k nahlédnutí v přehledu importovaných dat.

Kapitola 9

Závěr

Nástroj implementovaný v této práci umožňuje se připojit do vzdálené databáze projektu, používajícího knihovnu SForms pro sběr dat. Tato data importuje do interní databáze a provádí na nich identifikaci klíčových prvků, jako jsou uživatelské záznamy, formulářové verze, šablony otázek a další.

Pro správu formulářů je ve webovém rozhraní aplikace sekce importovaných dat, kde lze procházet uživatelské záznamy a jejich historii. Je zde možné zobrazit formuláře ve stavu, ve kterém se nacházely v době všech zaznamenaných uložení nebo dokonce automaticky porovnat odpovědi dvou po sobě jdoucích historických záznamů. Zobrazit je také možné všechny identifikované verze formulářů a podle nich rozlišit jednotlivé položky v historii uživatelských záznamů.

Aplikace disponuje sekcí výhradně pro vytváření analytických dotazů, pomocí nichž je možné zjišťovat pokročilé informace o evidovaných uživatelích, formulářích nebo otázkách. Pro účely provádění analýzy je také aplikace opatřena celou řadou předpřipravených šablon, které uživateli pomáhají vytvářet sofistikované dotazy v jazyce SPARQL.

Integrovaný tiketovací systém Trello umožňuje uvnitř webového rozhraní aplikace vytvářet a zobrazovat tikety, které souvisí s konkrétním projektem. Tyto tikety je možné asociovat s konkrétním formulářovým záznamem (tedy položkou v historii uživatelského záznamu), formulářovou verzí nebo přímo otázkou formuláře. Takové funkce umožňují uživatelům spolupracovat při návrhu struktury formulářů a při odhalování chybných stavů.

Poměrně zevrubná analýza existujících řešení ukázala širokou řadu zajímavých funkcí, které lze implementovat pro webového správce formulářů. V kombinaci s možnostmi knihovny SForms byly stanoveny poměrně ambiciózní požadavky na implementovanou aplikaci. Mnoho z požadovaných funkcí je součástí finálního řešení, ale i tak je v aplikaci velký prostor pro zlepšení. Velmi přínosné funkce, které by obohatily použitelnost aplikace, jsou porovnání

formulářových verzí a např. možnost zobrazit výsledek SPARQL dotazů jako grafy. Větší pozornost by také mohla být věnována sadě předpřipravených šablon, které je možné použít pro vytváření SPARQL dotazů. Dále aplikace stále obsahuje chyby, jejichž oprava nebyla při finalizaci implementace prioritou. Nicméně aplikace splnila zadání ve stanovených bodech.

Příloha A

Literatura

[1] Berners-Lee, Tim, et al. "The world-wide web."Communications of the ACM 37.8 (1994): 76-82.

[2] Berners-Lee, Tim, James Hendler, and Ora Lassila. "The semantic web."Scientific american 284.5 (2001): 34-43.

[3] Wood, David. 'What's New in RDF 1.1.' W3C Working Group Note (2014).

[4] Tzitzikas, Yannis, Christina Lantzaki, and Dimitris Zeginis. "Blank node matching and RDF/S comparison functions."International Semantic Web Conference. Springer, Berlin, Heidelberg, 2012.

[5] Brickley, Dan, Ramanathan V. Guha, and Brian McBride. "RDF Schema 1.1."W3C recommendation 25 (2014): 2004-2014.

[6] Gruber, Tom. "Ontology."(2018).

[7] Pérez, Jorge, Marcelo Arenas, and Claudio Gutierrez. "Semantics and complexity of SPARQL."ACM Transactions on Database Systems (TODS) 34.3 (2009): 1-45.

[8] Melton, Jim, and Alan R. Simon. Understanding the new SQL: a complete guide. Morgan Kaufmann, 1993.

[9] Jorge Pérez, Marcelo Arenas, and Claudio Gutierrez, Semantics of SPARQL, 2006

[10] Mendes Lübeck, Rafael, Milton Luiz Wittmann, and Luciana Flores Battistella. "Electronic ticketing system as a process of innovation."Journal of technology management innovation 7.1 (2012): 17-30.

[11] Levy, Sidney M. Project management in construction. McGraw-Hill Education, 2018.

[12] Highsmith, Jim. Agile project management: creating innovative products. Pearson education, 2009.

[13] Ahmad, Muhammad Ovais, Jouni Markkula, and Markku Oivo. "Kanban in software development: A systematic literature review."2013 39th Euromicro conference on software engineering and advanced applications. IEEE, 2013.

[14] Schwaber, Ken. Agile project management with Scrum. Microsoft press, 2004.

[15] Petersen, Kai, Claes Wohlin, and Dejan Baca. "The waterfall model in large-scale development."International Conference on Product-Focused Software Process Improvement. Springer, Berlin, Heidelberg, 2009.

[16] Johnson, Heather A. "Trello."Journal of the Medical Library Association: JMLA 105.2 (2017): 209.

[17] Masse, Mark. REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces. "O'Reilly Media, Inc.", 2011.

[18] 1. Arnold K, Gosling J, Holmes D. The Java programming language. Addison Wesley Professional; 2005.

[19] Berners-Lee, Tim, Roy Fielding, and Henrik Frystyk. "Hypertext transfer protocol–HTTP/1.0."(1996).

[20] Sarkan, Hasliza Md, Tengku Puteri Suhilah Ahmad, and Azuraini Abu Bakar. "Using JIRA and Redmine in requirement development for agile methodology."2011 Malaysian Conference in Software Engineering. IEEE, 2011.

[21] Legris, Paul, and Pierre Collerette. "A roadmap for IT project implementation: Integrating stakeholders and change management issues."Project Management Journal 37.5 (2006): 64-75.

[22] Gackenheimer, Cory. Introduction to React. Apress, 2015.

[23] Vasantha Raju, N., and N. S. Harinarayana. "Online survey tools: A case study of Google Forms."National Conference on Scientific, Computational Information Research Trends in Engineering, GSSS-IETW, Mysore. 2016.

[24] Oualline, Steve, and Grace Oualline. "Using Google Sheets."Practical Free Alternatives to Commercial Software. Apress, Berkeley, CA, 2018. 389-404.

[25] McKenna, Kelly, et al. "Visual-form learning analytics: a tool for critical reflection and feedback."Contemporary Educational Technology 10.3 (2019): 214-228.

[26] Cavelaars, Marinel, et al. "OpenClinica."Journal of clinical bioinformatics. Vol. 5. No. 1. BioMed Central, 2015.

[27] Patricio Ledesma, https://www.sofpromed.com/what-is-an-ecrf/, 2019 (navštíveno 10.6.2021)

[28] Walther, Brigitte, et al. "Comparison of electronic data capture (EDC) with the standard data capture method for clinical trial data."PloS one 6.9 (2011): e25348.

[29] Bellary, Shantala et al. “Basics of case report form designing in clinical research.” Perspectives in clinical research vol. 5,4 (2014): 159-66. doi:10.4103/2229-3485.140555

[30] Krishnankutty, Binny, et al. "Data management in clinical research: An overview."Indian journal of pharmacology 44.2 (2012): 168.

[31] Momjian, Bruce. PostgreSQL: introduction and concepts. Vol. 192. New York: Addison-Wesley, 2001.

[32] JSON-LD 1.1 Processing Algorithms and API: Flattening, https://w3c.github.io/json-ld-api/tests/flatten-manifest.html (navštíveno 3.6.2021)

[33] Klíma, Tomáš. Sémantický manažer prospektivní klinické studie. BS thesis. České vysoké učení technické v Praze. Vypočetní a informační centrum. 2018.

[34] Abdalimov, Zakir. Pokročilý Firemní Issue Tracker. BS thesis. České vysoké učení technické v Praze. Vypočetní a informační centrum., 2019.

[35] Eclipse RDF4J developers. Welcome · Eclipse RDF4J™ | The Eclipse Foundation. https://rdf4j.org/ (navštíveno 1.5.2021)

[36] Doroshenko Yan. Editor sémantických datových proudů, BS thesis. České vysoké učení technické v Praze. Vypočetní a informační centrum. 2018.

[37] SPARQLMotion. https://sparqlmotion.org/. (navštíveno 10.5.2021).

[38] Java 8 Stream API. https://www.baeldung.com/java-8-streams. (navštíveno 10.7.2021)

[39] Oracle Java Documentation: What Is a Class?. https://docs.oracle.com/javase/tutorial/java/concepts/class.html. (navštíveno 1.6.2021)

[40] Tomáš Klíma. Editor sémantických webových formulářů. BS thesis. České vysoké učení technické v Praze. Vypočetní a informační centrum. 2021.

[41] The Java Language Specification, Java SE 8 Edition https://docs.oracle.com/javase/specs/jls/se8/html/index.html (navštíveno 6.8.2021)

[42] Waters, Kelly. "Prioritization using moscow."Agile Planning 12 (2009): 31.

[43] Layered Architecture - Software Architecture Patterns. https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch01.html (navštíveno 5.8.2021)

[44] Spring Framework documentation. Web MVC framework. https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/mvc.html (navštíveno 5.8.2021)

[45] [5] M. Ledvinka and P. Křemen: JOPA: Accessing Ontologies in an Object-oriented Way. In Proceedings of the 17th International Conference on Enterprise Information Systems. Porto: SciTePress - Science and Technology Publications, 2015, p. 212-222. ISBN 978-989-758-096-0.

[46] Johnson, Rod, et al. "The spring framework–reference documentation."interface 21 (2004): 27.

[47] Prasanna, Dhananjay. Dependency injection: design patterns using spring and guice. Simon and Schuster, 2009.

[48] Sporny, Manu, et al. "JSON-LD 1.0."W3C recommendation 16 (2014): 41.

[49] McGuinness, Deborah L., and Frank Van Harmelen. "OWL web ontology language overview."W3C recommendation 10.10 (2004): 2004.

[50] Atlassian Developer: Trello REST API specification. https://developer.atlassian.com/cloud/trello/rest/api-group-actions/ . (navštíveno 10.7.2021)

[51] Franks, John, et al. "HTTP authentication: Basic and digest access authentication."(1999): 78.

[52] Hammer-Lahav, Eran, David Recordon, and D. Hardt. The oauth 1.0 protocol. RFC 5849, April, 2010.

Příloha B

Návod k instalaci

Požadavky na prostředí

- **Java 8**
- **Maven 4** - ke stažení na adrese `https://maven.apache.org/download.cgi#`
- **Node.js 10+** - ke stažení na adrese `https://nodejs.org/en/download/`

Potřebné soubory

- **s-forms-manager.zip** - viz příloha
- **s-forms-manager-ui.zip** - viz příloha
- **RDF4J Server + Workbench 2.2.5+** - ke stažení: `http://rdf4j.org/download/`
- **Apache Tomcat 8.5.+** - ke stažení: `https://tomcat.apache.org/download-80.cgi`

Instalace

1. ujistěte se, že splňujete požadavky z výše uvedeného seznamu a případně upravte prostředí
2. extrahujte s-forms-manager.zip a s-forms-manager-ui.zip z přílohy
3. extrahujte Apache Tomcat a vytvořte jeho kopii tak, abyste měli 2 Apache Tomcat
   - 1 slouží jako prostředí pro Java aplikaci

- 1 slouží jako prostředí pro RDJ4J server + Workbench

4. přejděte do složky "conf"v kopii Apache Tomcat
5. otevřete soubor "server.xml“
6. v souboru změňte všechny výskyty čísla 8080 na 8090 a uložte soubor
7. extrahujte RDF4J Server a ze složky "war“ vyjměte obsah a vložte ho do složky "webapps“ v kopii Apache Tomcat
8. kopii Apache Tomcat spusťte pomocí souboru startup.bat (nebo startup.sh) ve složce bin
9. v prohlížeči přejděte na adresu `http://localhost:8090/rdf4j-workbench/` a vytvořte nativní uložiště se jménem "sforms_manager"
10. v přehledu informací o nově vytvořeném repozitáři zkopírujte adresu repozitáře
11. otevřete konfigurační soubor aplikace sforms-manager ve složce s-forms-manager/src/main/resources/application.properties
12. nastavte hodnotu konfigurace "repositoryUrl"na hodnotu zkopírovanou adresu RDF repozitáře
13. nastavte hodnotu konfigurace "server.port"na hodnotu 8080
14. v kořenovém adresáři projektu s-forms-manager zavolejte příkaz "mvn clean install“
15. po dokončení příkazu zkopírujte war soubor ze složky "target"do složky webapps v Apache Tomcat (ne do kopie)
16. Apache Tomcat spusťte pomocí souboru startup.bat (nebo startup.sh) ve složce bin
17. nyní by měl běžet databázový RDF4J server i serverová část aplikace
18. pro spuštění klientská části aplikace přejděte do extrahovaného adresáře s-forms-manager-ui
19. soubor ".env-template"zkopírujte a kopii pojmenujte ".env"
20. ujistěte se, že v souboru je hodnota BACKEND_API_URL=http://localhost:8080
21. následně spusťte příkaz "npm install"a "npm start"
22. nyní by měla běžet i klientská část aplikace, která se automaticky připojí k serverové části
23. nástroj je možné používat na adrese http://localhost:3000