České vysoké učení technické v Praze
Fakulta elektrotechnická


# Sémantický manažer pro prospektivní klinické studie

# Semantic manager for prospective clinical trials

Bakalářská práce

Autor: **Tomáš Klíma**

Vedoucí práce: **Mgr. Miroslav Blaško, Ph.D.**

Akademický rok: 2017/2018

Semestr: letní

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

## I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Klíma** Jméno: **Tomáš** Osobní číslo: **459933**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

## II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

**Sémantický manažer prospektivní klinické studie**

Název bakalářské práce anglicky:

**Semantic manager for prospective clinical trials**

Pokyny pro vypracování:

StudyManager je eCRF systém pro management jednoduchých klinických studií založených na technologiích Sémantického webu. Aplikace zajišťuje správu uživatelů a záznamů o pacientech. Konkrétní formuláře pacientů jsou definované externě vůči aplikaci pomocí RESTové webové služby poskytující deklarativní popis struktury a interakce formulářů pro sběr dat o pacientech. Aplikace podporuje jenom retrospektivní studie s jediným typem formuláře. Cílem práce je rozšíření aplikace, případně vytvoření nové aplikace pro prospektivní klinické studie s podporou jednoduché randomizace, plánování a notifikací uživatelů.

1) přezkoumejte a popište možnosti aplikace StudyManager včetně možností dynamických formulářů pro sběr dat o pacientech
2) přezkoumejte požadavky prospektivních studií a porovnejte možnosti existujících nástrojů pro správu prospektivních studií
3) definujte uživatelské scénáře pro práci s generickým nástrojem pro prospektivní studie dle předcházejících analýz
4) navrhněte a implementujte aplikaci
5) otestujte použitelnost aplikace na definovaných scénářích minimálně na 3 uživatelích
6) porovnejte implementované řešení s existujícími nástroji

Seznam doporučené literatury:

[1] Bellary, Shantala, Binny Krishnankutty, and M. S. Latha. "Basics of case report form designing in clinical research." Perspectives in clinical research 5.4 (2014): 159.
[2] Blaško, Miroslav and Petr Křemen, "SForms" (online at https://kbss.felk.cvut.cz/web/kbss/s-forms)
[3] Křemen, Petr, and Zdeněk Kouba. "Ontology-driven information system design." IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 42.3 (2012): 334-344.
[4] Lanthaler, Markus, and Christian Gütl. "On using JSON-LD to create evolvable RESTful services." Proceedings of the Third International Workshop on RESTful Design. ACM, 2012

Jméno a pracoviště vedoucí(ho) bakalářské práce:

**Mgr. Miroslav Blaško, Ph.D., Skupina znalostních softwarových systémů FEL**

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

Datum zadání bakalářské práce: **09.02.2018** Termín odevzdání bakalářské práce: _____

Platnost zadání bakalářské práce: **30.09.2019**

| Mgr. Miroslav Blaško, Ph.D.<br>podpis vedoucí(ho) práce | podpis vedoucí(ho) ústavu/katedry | prof. Ing. Pavel Ripka, CSc.<br>podpis děkana(ky) |
|---|---|---|

*Poděkování:*
Chtěl bych zde poděkovat především svému školiteli Mgr. Miroslavu Blaškovi, Ph.D. za pečlivost, ochotu, vstřícnost a odborné i lidské zázemí při vedení mé bakalářské práce. Dále MUDr. Anně Germanové Ph.D. za zkontrolování teoretické části klinických studií.

*Čestné prohlášení:*
Prohlašuji, že jsem tuto práci vypracoval samostatně a uvedl jsem všechnu použitou literaturu.

V Praze dne 23. května 2018 Tomáš Klíma

*Název práce:*

**Sémantický manažer pro prospektivní klinické studie**

*Autor:* Tomáš Klíma

*Obor:* Softwarové inženýrství a technologie

*Druh práce:* Bakalářská práce

*Vedoucí práce:* Mgr. Miroslav Blaško, Ph.D., České vysoké učení technické v Praze, Fakulta elektrotechnická, Skupina znalostních softwarových systémů

*Abstrakt:* Cílem bakalářské práce je analýza klinických studií a následné vytvoření aplikace pro prospektivní klinické studie založené na technologiích Sémenatického webu, případně rozšíření aplikace StudyManager, což je existující ontologický webový nástroj pro správu retrospektivních klinických studií. Text provází čtenáře od analýzy existujících řešení přes popis aplikace StudyManager až po návrh nových funkcionalit. Výstupem práce je rozšířená webová aplikace, která pomáhá klinickým výzkumníkům zefektivnit proces sběru dat a provádět prospektivní klinické studie. Serverová část je implementována v jazyce Java za použití frameworku Spring, klientská část potom pomocí JavaScriptové knihovny React a Redux.

*Klíčová slova:* Prospektivní klinické studie, CRF, CTMS, Java, Spring, SPipes, JavaScript, React, Redux, SForms, JOPA, RDF, OWL, Ontologie, Sémantický Web, Linked Data

*Title:*

**Semantic manager for prospective clinical trials**

*Author:* Tomáš Klíma

*Abstract:* The goal of this bachelor thesis is to analyze clinical studies and thereafter create an ontology-based application for prospective clinical studies or extend existing one StudyManager, which is an ontological web tool for retrospective clinical studies. The text leads the reader from analysis of existing tools through description of application to the design of new functionalities. The output of the work is extended web application which will help clinical researchers to streamline the data capture process and build the clinical study. The server side of the system is based on Java programming language using Spring framework. The client side is developed thanks to JavaScript library React and Redux.

*Key words:* Prospective clinical study, CRF, CTMS, Java, Spring, SPipes, JavaScript, React, Redux, SForms, JOPA, RDF, OWL, Ontology, Sémantic Web, Linked Data

# Obsah

# Úvod

Předtím než je nový typ léčby dostupný veřejnosti, probíhá i několikaletý výzkum v podobě klinických studií [1]. Ten slouží k ověření účinnosti a bezpečnosti nového léčebného postupu spolu se srovnáním s již schválenými a zavedenými medicínskými postupy. Sběr dat je možné provádět jak cestou klasickou pomocí papíru a tužky či využitím různých kancelářských programů, nebo způsobem, který může pomoci zefektivnit průběh a zvýšit úspešnost klinických studií. Prostředkem k tomu jsou aplikace pro správu klinických studií.

Cílem této bakalářské práce je zanalyzovat požadavky prospektivních klinických studií a vytvořit nový, případně rozšířit existující nástroj pro klinické studie StudyManager. StudyManager je webová aplikace založená na technologiích Sémantického Webu, jež umožňují jednodušší integraci s existujícími webovými zdroji jakou jsou například standardní medicínské slovníky. Také poskytují možnost publikace dat ve formě Linked Data [2], což usnadňuje jejich opětovné využití na webu.

Aplikace byla vytvořena za účelem tvorby nekomerčních studií tak, aby si zdravotnické instituce mohly samy provádět výzkum. Cílovou skupinou jsou převážně fakultní nemocnice, které mají zájem o jednoduchou studii a chtějí spolupracovat s ostatními institucemi na mezinárodní úrovni. StudyManager byl koncipován pro retrospektivní klinické studie a byl již třikrát použit. Standardem v klinickém výzkumu jsou ale studie prospektivní, a to hlavně randomizovaná kontrolovaná studie, která je v medicíně založené na věrohodnosti důkazů na druhé příčce po metaanalýze[1] [3]. Z tohoto důvodu jsem se rozhodl vylepšit a rozšířit webovou aplikaci StudyManager, aby podporovala i studie prospektivní.

Obsahem této práce je uvedení do tématu v kapitole 1. Dále se věnuji analýze možných způsobů sběru klinických dat pro výzkumné studie v kapitole analýza existujících řešení, která má pomoci vyvarovat se chyb a špatných řešení a také usnadnit výběr nových funkcí pro aplikaci. V další kapitole popisuji aplikaci StudyManager a stav předtím, než jsem na ní začal pracovat. Zabývám se také tím, co je aktuálně možné v aplikaci dělat spolu s použitými technologiemi. Samostatnému návrhu na rozšíření aplikace se věnuji v kapitole 4. Na to navazuji představením implementací jednotlivých navržených funkcionalit v aplikaci. V neposlední řadě rozebírám způsoby testování rozšířené aplikace.

[1] Metaanalýza - studie založená na kombinaci výsledků více studií

# Kapitola 1

# Uvedení do tématu

V úvodu této kapitoly definuji základní pojmy, které budou použity v rámci celé bakalářské práce. Následně rozeberu, co jsou to klinické studie, o které se opírá celá tato práce.

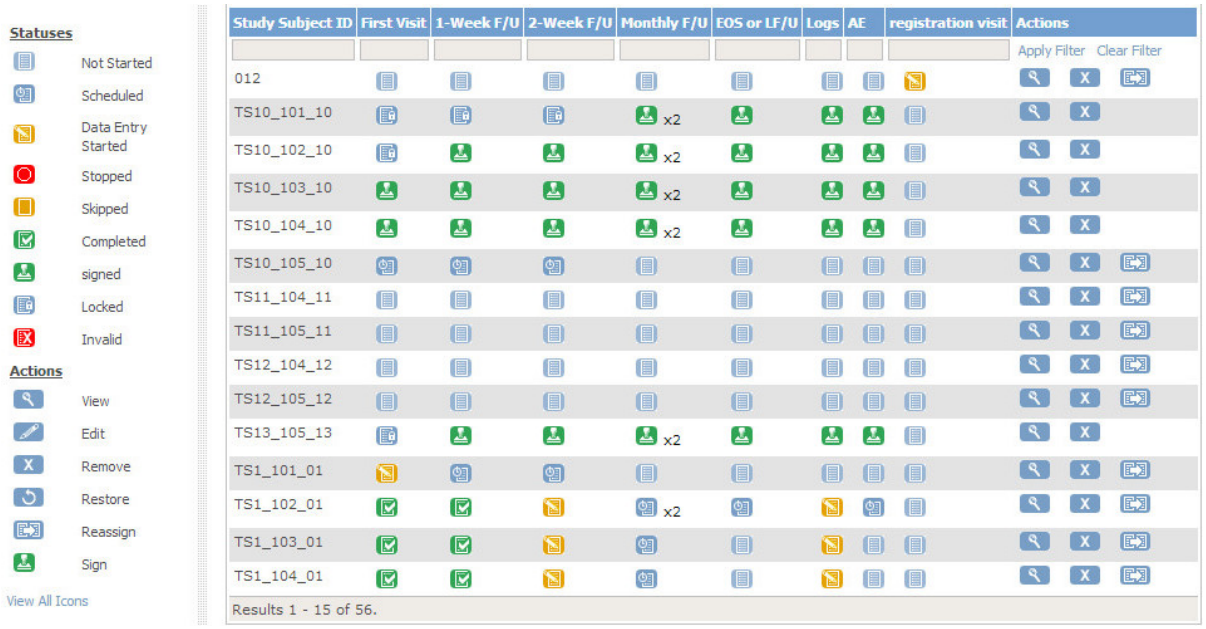
Tabulka 1.1: Základní pojmy

| **Pojem** | **Definice** |
|---|---|
| Klinické hodnocení | Klinické hodnocení či klinická studie (dále v textu také jako „studie“), je systematické ověření bezpečnosti a nových léčebných postupů [4]. |
| Subjekt hodnocení | Subjekt hodnocení (dále v textu jen „subjekt“) je pacient nebo zdravý dobrovolník zařazený do klinického hodnocení. [1] |
| Retrospektivní klinická studie | Data o subjektu se zjišťují zpětně z minulosti, například z dokumentace. |
| Prospektivní klinická studie | Sběr dat o subjektu se provádí průběžně po dobu několika let. |
| Kohorta | Skupina subjektů mající určité společné vlastnosti. Například může být kohorta kuřáků a nekuřáků [5]. |
| Rameno studie | Skupina subjektů dostávající stejný typ léčby [6]. |
| Prognostický faktor | Jakákoli vlastnost nemoci či subjektu, která může pozitivně či negativně ovlivnit průběh nemoci [5]. |
| Stratifikační kritérium | Kritérium, na základě kterého se rozdělují subjekty do ramen studie. Může jím být například pohlaví, věk či předchozí léčba. [5]. |
| CRF | CRF[1] je papírový (pCRF[2]) či elektronický formulář (eCRF[3]) používaný u pacientů zařazených do klinické studie. Představuje důležitou část průběhu klinické studie a může mít vliv na její kladný výsledek [7]. Slouží ke sběru řady údajů o subjektu, jako je jeho nemoc a informace o probíhající léčbě, o nežádoucích příhodách, o výsledcích laboratorních vyšetření a dalších dat souvisejících s léčbou. |

[1] CRF - Case Report Form
[2] pCRF - paper Case Report Form
[3] eCRF - electronical Case Report Form

| | |
|---|---|
| Lékařské číselníky | Využívají standardu MKN[4], který zařazuje každé lidské onemocnění, úraz a další zdravotní problémy do kategorií spolu s názvem, unikátním přiděleným kódem a stručným popisem daného zdravotního problému [8]. |
| Matice subjektů | Matice subjektů, kterou je možné vidět na obrázku 1.1 na straně 14 je tabulka, která graficky znázorňuje informace o aktuální stavu všech jednotlivých záznamů o subjektech studie. Na každém řádku tabulky se nachází jeden subjekt, v prvním sloupci jeho označení (Study Subject ID) a v dalších sloupcích názvy jednotlivých částí studie. Každá buňka obsahuje barevnou ikonu, která indikuje aktuální stav dané části studie a je možné na ni kliknout a dostat se na eCRF. Pro každý subjekt se zde navíc nacházejí akce, jako například prohlednutí, upravení, či smazání subjektu. |

Obrázek 1.1: Matice subjektů v aplikaci OpenClinica




Zdroj: `https://github.com/OpenClinica/OpenClinica` k 10.11.2017

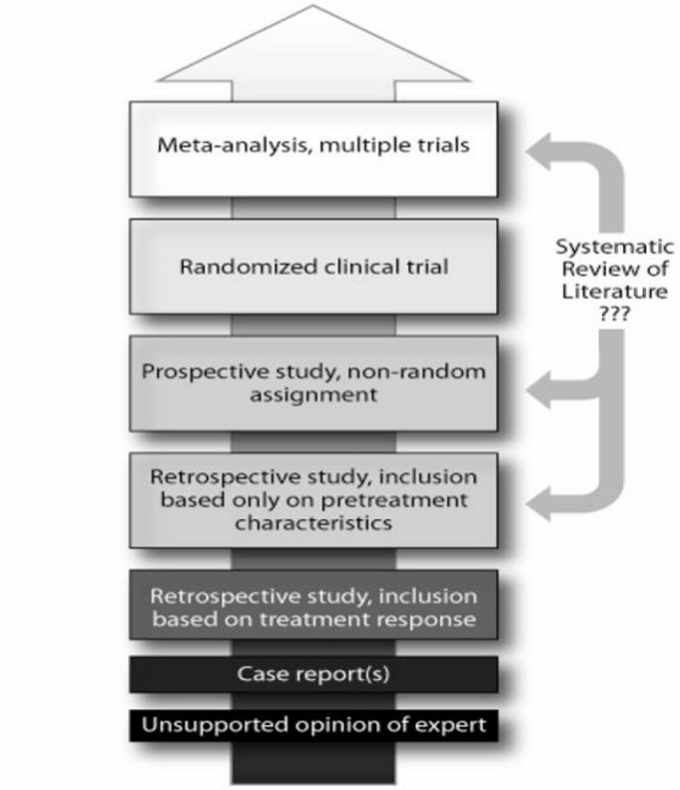
## 1.1 Klinické studie

Klinické studie patří spolu se základním lékařským výzkumem a epidemiologickým výzkumem do lékařského výzkumu. Základní lékařský výzkum se soustředí hlavně na experimenty a na nalezení a zlepšení vědeckých postupů. Epidemiologický výzkum se zaměřuje na výskyt a historické změny v četnosti onemocnění a jejich příčiny [9].

Pro tuto práci jsou důležité klinické studie. Dělí na dva typy: intervenční neboli experimentální a neintervenční neboli observační. Na obrázku 1.2, je možné vidět hierarchii věrohodnosti většiny studií v tomto odvětví.

[4]MKN - Mezinárodní klasifikace nemocí a souvisejících zdravotních problémů

Obrázek 1.2: Hierarchie věrohodnosti klinických studií




Zdroj: `https://www.ncbi.nlm.nih.gov/pubmed/28651753` k 27.4.2018

### 1.1.1 Observační studie

V těchto studiích je subjekt pozorován, zkoušející nezasahuje do průběhu choroby, jelikož nejde o experiment. Observační studie se dělí na dva typy, a to na prospektivní studii, jež sleduje subjekt po určitou dobu a retrospektivní, která zjišťuje data o subjektu z minulosti [10]. Nejčastějšími typy jsou prospektivní a retrospektivní kohortová studie a studie případů a kontrol.

#### 1.1.1.1 Kohortové studie

U prospektivní kohortové studie hledáme odpověď na otázku, zda určitý suspektní faktor vyvolává nemoc. K této studii najmeme a rozdělíme lidi do dvou skupin neboli kohort. Jedna z nich se během života vystavila určitému rizikovému faktoru, například kouření cigaret. Druhá kohorta tato rizika nepodstoupila. Kohorta, která se vystavila rizikovým faktorům, se označuje jako exponovaná, opačná se nazývá neexponovaná. Tyto dvě kohorty sledujeme po dobu několika let. Na konci studie zjišťujeme vztah mezi suspektním faktorem a projevenými nemocemi [11].

U retrospektivní kohortové studie jsou subjekty také rozděleny na exponované a neexponované a je zde porovnáván vztah mezi suspektním faktorem a danou chorobou. Rozdíl je pouze v tom, že suspektní faktory a výsledek zjišťujeme retrospektivně [11].

#### 1.1.1.2 Studie případů a kontrol

Studie případů a kontrol patří do retrospektivní studie a oproti kohortové studii zkoumá určitou chorobu a hledá k ní rizikové faktory. Porovnáváme zde dvě skupiny lidí. Případy, což je skupina pacientů postižených chorobou a kontroly, což jsou zdraví lidé se stejnými prognostickými faktory, již slouží jako kontrola [3].

### 1.1.2 Experimentální studie

Experimentální studie jsou charakteristické intervencí, což znamená, že do průběhu nemoci, na rozdíl od studie observační, zasahujeme použitím farmak nebo jinými léčebnými procesy (např. rehabilitací). Jsou pouze prospektivní a subjekty sledujeme několik let po určitých intervalech s určitým množstvím procedur a s možností procedur mimořádných z důvodu případných komplikací. Pomocí těchto studií, zjišťujeme například efektivitu nového léku nebo odhalujeme nežádoucí účinky [12]. Než je lék schválen, projde po předklinických zkouškách v laboratoři několika fázemi klinických studií [3]:

**1. fáze** V této fázi se stanoví maximální možná tolerovaná dávka léku, ze které se vychází v dalších fázích. Zjišťují se také možné nežádoucí účinky. Testy se provádí na malém počtu pacientů, na zdravých dobrovolnících i na subjektech, kteří již mají vyčerpanou veškerou onkologickou léčbu [12].

**2. fáze** U této fáze zjišťujeme účinnost a bezpečnost dávky léku na určitý typ nemoci. Této etapy se účastní více subjektů než v 1. fázi. Pokud jsou výsledky příznivé, postupujeme dál [12].

**3. fáze** V této etapě již testujeme lék na velkém množství subjektů (viz sekce 1.1.2.2). Výsledek srovnáváme s placebem nebo standardně používaným léčivem. Díky této fázi může být nový lék schválen [3].

**4. fáze** Tato fáze slouží k odhalení nežádoucích účinků při dlouhodobém používaní, popřípadě se může vést jako fáze třetí ke zjištění nových informací [1].

#### 1.1.2.1 Průběh studie

Jednoduchý průběh klinické studie je možné vidět na obrázku 1.3. Hlavními aktéry jsou [1]:

**Zadavatel** Osoba, společnost, instituce nebo organizace realizující studii.

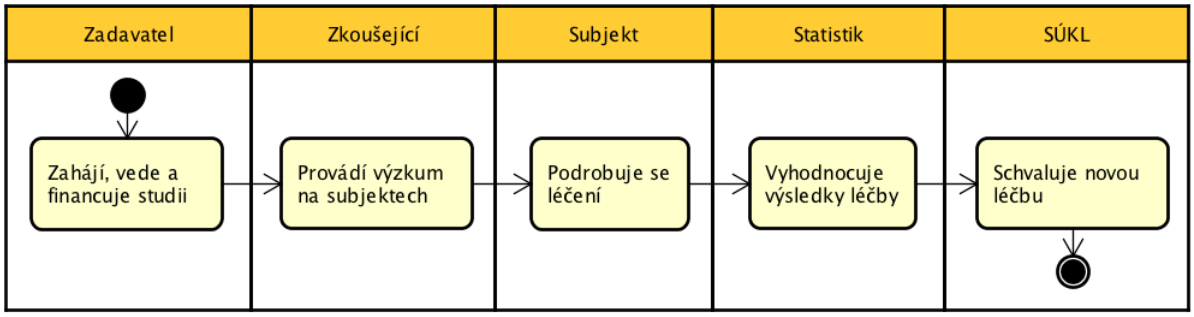
**Zkoušející** Osoby provádějící výzkum. Mohou jimi být lékaři a studijní sestry.

**Subjekt**

**Statistik** Vyhodnocuje výsledky léčby [3].

**SÚKL** Státní ústav pro kontrolu léčiv schvalující lék.

Obrázek 1.3: Průběh intervenční studie



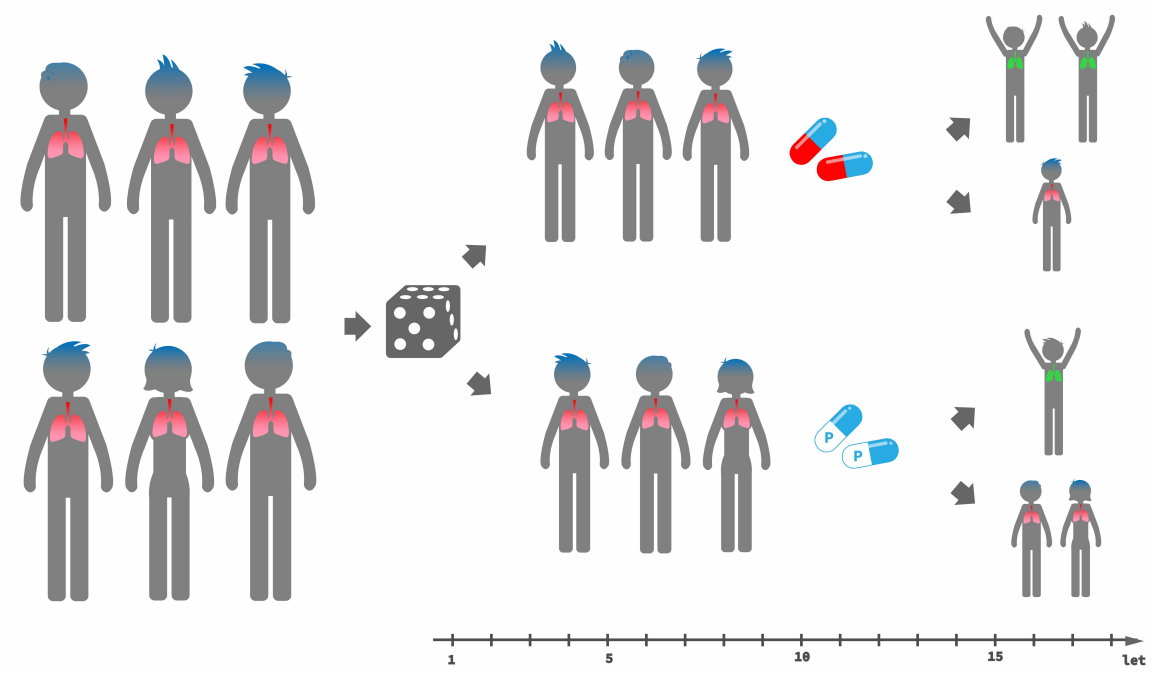

UML 2.0 - Diagram aktivit [14]

#### 1.1.2.2 Randomizovaná kontrolovaná studie

Nejdůležitější a nejefektivnější studií pro schválení nové léčby je randomizovaná kontrolovaná studie (dále v textu jen „RCT“) s alespoň jedním experimentálním ramenem a kontrolním ramenem. Dle pana P. S. Mulimaniho nejlepší metoda pro vytvoření spolehlivých a neovlivněných výsledků [13]. Léčba probíhá ve všech ramenech souběžně [3]. Subjekty přiřazení do experimentálního ramene berou nový lék, ostatní užívají placebo či nějaké již standardizované léčivo.

K rozdělení subjektů do ramen se používá randomizace (viz kapitola 1.2). Subjekt nemusí vědět, zda je doopravdy léčen, či zda dostává pouze placebo. Nemusí to dokonce vědět ani zkoušející. Taková studie se nazývá zaslepená, více v sekci 1.1.3. Ilustraci RCT s dvěma rameny je možné vidět na obrázku 1.4.

Obrázek 1.4: Randomizovaná kontrolovaná studie



### 1.1.3 Zaslepení studie

Randomizované studie bývají často zaslepené, a to buď jednoduše, dvojitě či trojitě. Studie jednoduše zaslepená znamená, že subjekt neví a nepozná, zda dostává léčbu A či B. Například placebo či testovací lék. Dvojitě zaslepená znamená, že ani zkoušející neví, jakou léčbu subjekt dostává a trojitě zaslepená znamená, že ani statistik vyhodnocující výsledky neví, co léčba A a B znamená [10].

Pokud studie není zaslepená, může znalost ramene, do něhož subjekty patří, ovlivnit odpovědi během procedur, které se týkají jejich psychického a fyzického stavu. Pokud subjekt ví, že dostává novou léčbu, může mít strach z nové léčby, ale zároveň i kladná očekávání, že nová léčba je lepší než léčba standardní. Naopak ti, co ví, že dostávají standardní léčbu, mohou pociťovat sklíčenost, nebo naopak úlevu [15].

## 1.2 Randomizace

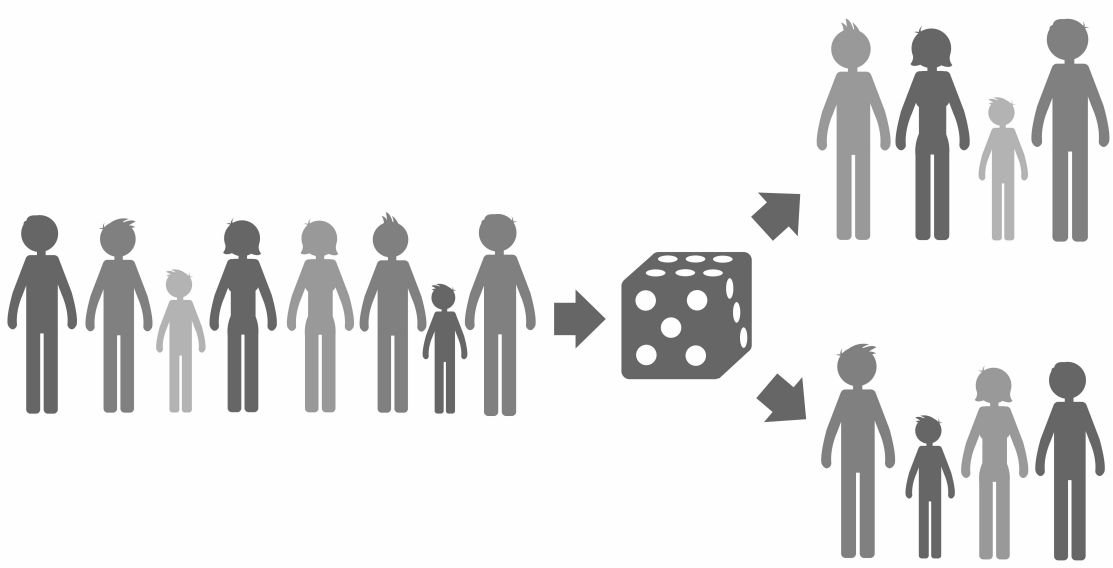
Randomizace je metoda hojně využívaná v klinických studiích a dalších biologických experimentech. Slouží k rozřazení subjektů studií do dvou a více ramen na základě náhody. Hlavními výhodami randomizace je zajištění, že každý subjekt má danou míru pravděpodobnosti na to být přiřazen do ramene a také, že ramena si budou podobná až na způsob léčení [16].

Jak již bylo stručně řečeno, důvodů k využití randomizace je mnoho. Jedním z nich je, že ramena by se neměla na základě určených prognostických faktorů příliš lišit, aby výsledek studie nebyl ovlivněný [10]. Předpokládejme například, že máme dvě ramena a že pro studii je zásadní věk subjektu, pokud by v jednom z ramen byl o mnoho vyšší věkový průměr, mohlo by to mít zásadní vliv na výsledek celé studie. Ilustraci randomizace je možné najít na obrázku 1.5

Dalším důvodem je zamezení subjektivního a selektivního rozdělování subjektů studie do ramen, což znamená, že zkoušející, subjekt ani ostatní nemohou ovlivnit, do jakého ramene bude subjekt zařazen. Dále také zajištění požadovaného poměru počtu subjektů v každém rameni [6].

Schul a Grimes uvedli, že studie s nedostatečnou nebo nejasnou randomizací vedly k nadhodnocení léčebných účinků až o 40% v porovnání s těmi, jež používaly správnou randomizaci. Výsledek studie tedy může být zásadně negativně ovlivněn randomizací [16].

Obrázek 1.5: Randomizace



### 1.2.1 Typy randomizace

Existuje mnoho různých metod, jak rozdělit subjekty do ramen, některé jsou však nepřípustné, což je například randomizace podle data narození, pořadového čísla či iniciálů subjektu. Některé jsou méně vhodné a některé doporučené [6]. Metody se dělí do čtyř kategorií: jednoduchá, permutační bloková, stratifikovaná permutační bloková a adaptivní randomizace.

#### 1.2.1.1 Jednoduchá randomizace

Jednoduchá randomizace patří do méně vhodných randomizačních metod, jelikož není brán zřetel na stratifikační kritéria a u malých studií je velmi pravděpodobné, že bude velký rozdíl mezi počtem subjektů v každém rameni. Při 100 subjektech je pouze 8% pravděpodobnost shodného počtu subjektů v každém rameni [6]. Rozdělení subjektů do ramen může probíhat například na základě hodu mincí či kostkou.

#### 1.2.1.2 Permutační bloková randomizace

Tato randomizace patří mezi doporučené, jelikož zajišťuje vyrovnaný počet subjektů v každém rameni, ale nezaručuje rovnoměrné rozdělení subjektů podle prognostických faktorů. Funguje na principu rozdělení subjektů do bloků po 4, 6 či 8 a poté vypočítání všech vyvážených variací přiřazení v bloku. Například rozdělení AABB, je možné variovat jako ABAB, ABBA, BBAA, BABA a BAAB. Bloky se následně náhodně vybírají a podle nich se přiřazují subjekty do ramen.

#### 1.2.1.3 Stratifikovaná permutační bloková randomizace

Stratifikovaná permutační bloková randomizace má všechny výhody permutační blokové randomizace, ale bere navíc v úvahu stratifikační kritéria. Nevýhodou této metody je nutná znalost všech subjektů hned na začátku randomizace a komplikovanost při větším počtu stratifikačních kritérií [6]. Funguje na stejném principu jako permutační bloková randomizace, jen se navíc vytvoří podskupiny na základě všech možných kombinací stratifikačních kritérií. Na obrázku 1.6 pod tímto odstavcem je možné vidět bloky pro rozdělení subjektů, u kterých jsou kritérii pohlaví a věk.

Obrázek 1.6: Stratifikovaná permutační bloková randomizace v rámci podskupin

| | Skupina | | | |
|---|---|---|---|---|
| | Muži | | Ženy | |
| Blok | ≤50 | >50 | ≤50 | >50 |
| | A | A | B | A |
| | B | A | B | A |
| | B | B | A | B |
| 1 | A | B | A | B |
| | B | A | B | A |
| | B | B | A | B |
| | A | A | A | A |
| 2 | A | B | B | B |
| | . | . | . | . |

Zdroj: http://ksvi.mff.cuni.cz/ holan/Prednaska2.pps

#### 1.2.1.4 Adaptivní randomizace

Adaptivní randomizace zajišťuje rovnoměrný počet subjektů a rozdělení prognostických faktorů v ramenech. Hlavní výhodou oproti stratifikované permutační blokové randomizaci je, že můžeme přidávat subjekty do ramen průběžně. Mezi hlavní adaptivní metody patří metoda play the winner, metoda osudí nebo minimalizace, která není založena na náhodě, ale na základě toho, v jakém rameni by subjekt minimalizoval rozdíly mezi rameny.

**Osudí:** Subjekty jsou přiřazeni s pravděpodobností větší 0,5 do ramene s menším počtem již randomizovaných subjektů. Pokud je v každém rameni stejný počet, poté s pravděpodobností 0,5. Je vhodné pro studie s malým počtem subjektů, kvůli zabránění nevyváženého počtu subjektů v každém rameni [6]. Pravděpodobnost přiřazení do ramene spočteme pomocí vzorce:

$$P_{r(A)} = \frac{(max_{n_a,n_b} + 1) - n_a}{((max_{n_a,n_b} + 1) - n_a) + ((max_{n_a,n_b} + 1) - n_b)}$$

kde:

$P_{r(A)}$ je pravděpodobnost přiřazení subjektu do ramene A

$n_a$ je počet již přiřazených subjektů do ramene A

$n_b$ je počet již přiřazených subjektů do ramene B

$max_{n_a,n_b}$ je nejvyšší počet již přiřazených subjektů z ramene A a B

**Minimalizace:** Subjekty jsou přiřazováni do ramen na základě rozložení prognostických faktorů z dosavadního průběhu studie [6]. V tabulce 1.2 pod tímto textem je možné vidět aktuální stav rozložení prognostických faktorů ve studii o 40 subjektech.

Tabulka 1.2: Aktuální stav ve studii po 40 přiřazených subjektech

| **Stratifikační kritérium** | **Hodnota** | **Počet v rameni A** | **Počet v rameni B** |
|---|---|---|---|
| Pohlaví | Muž | 13 | 12 |
| | Žena | 8 | 7 |
| Věk | <= 20 | 12 | 10 |
| | <= 40 | 8 | 6 |
| | > 40 | 1 | 3 |
| **Celkem** | | **21** | **19** |

Pokud nyní chceme přiřadit nový subjekt, jímž je 28letá žena, uvažujeme pouze hodnoty pro tento subjekt z tabulky 1.2. Přiřazení vypadá následovně. Spočteme si, kolik subjektů splňuje stratifikační kritéria v obou ramenech. V rameni, kde nám součet vyjde nižší, přiřadíme nový subjekt (viz tabulka 1.3). Pokud součet vyjde stejně, přiřadíme nový subjekt s pravděpodobností 0,5 do jednoho z nich.

Tabulka 1.3: Randomizace nového subjektu dle prognostických faktorů

| **Prognostický faktor** | **Počet v rameni A** | **Počet v rameni B** |
|---|---|---|
| Žena | 8 | 7 |
| 28 let | 8 | 6 |
| **Celkem** | **16** | **13** |

Rameno A má větší hodnotu.

=> Přiřadíme nový subjekt do ramene B.

# Kapitola 2

# Analýza existujících řešení

Tato kapitola popisuje analýzu existujících řešení pro klinické studie, kterou využiji k vyvarování se chyb a špatným řešením při návrhu aplikace. Pro hlubší analýzu jsem vybral tradiční sběr pomocí pCRF či kancelářského balíku a 5 aplikací určených pro výzkumné studie. Zvolil jsem aplikace REDCap, OpenClinica, ClinCapture, CastorEDC a OpenMRS, jelikož, až na ClinCapture, mají volně dostupné demo na webových stránkách. Ostatní nástroje, které jsem našel pomocí vyhledávače Google[1] a srovnávací aplikace Capterra[2] demo nemají, či je nutné objednat přímo od společnosti dodávající aplikaci.

ClinCapture jsem zahrnul do analýzy, jelikož aplikace OpenClinica, která má základ v aplikaci ClinCapture, je dle srovnávací aplikace Capterra nejvíce žhavý produkt mezi aplikacemi pro klinické studie, a zdá se mi proto vhodné tuto aplikaci také podrobit analýze.

## 2.1 Papírové versus elektronické CRF

Tradičním způsobem sběru dat je používání papírových CRF. Tento způsob je použitelný, pouze pokud je náš výzkum malý či velmi různorodý, zatímco elektronická verze je vhodná pro velké studie, které mají podobný průběh [17].

V dnešní době jsou eCRF upřednostňovány před tradičními pCRF, jelikož snižují čas strávený zadáváním dat. eCRF jsou propojené jedny s druhými. Systémy automaticky generují data, jako je například předmět studie, číslo studie, informace o subjektu a datum, čímž eliminují počet zadávaných duplicitních dat [18].

Snižují také počet chyb ve studii, jelikož eCRF mohou obsahovat různé validace dat, automatické vyplňování nebo integraci standardních medicínských taxonomií, zatímco u papírových CRF chyby zůstanou nepovšimnuté. Dalším problémem může být historie změn, pokud se něco přepíše v papírovém formuláři, už se nemusí nikdy zjistit, kdo změnu udělal a jaká data byla nahrazena.

eCRF jsou také mnohem levnější. Dle výzkumu provedeného mezi lety 2001 až 2011 na 27 klinických studiích, ze kterých 16 používalo pCRF a 11 eCRF, se zjistilo, že cena za jednoho pacienta při využívání pCRF byla 1135€ ± 1234, zatímco při využití eCRF byla 374€ ± 351 bez započítání licenčního poplatku za používání některého z manažeru klinických studií neboli CTMS[3] [19].

I přes mnoho pozitiv, která tu ani nebyla zmíněna, nejsou eCRF velmi rozšířené. Hlavním důvodem je technická neznalost či neochota klinických výzkumníků se učit něčemu novému. Také vysoká obtížnost instalace a nastavení, údržba a vysoká počáteční investice do CTMS [18].

[1] Google - Vyhledávač dostupný na adrese: https://www.google.cz/

[2] Capterra - Srovnaní dostupné na adrese: https://www.capterra.com/clinical-trial-management-software/ k 5.5.2018

[3] CTMS - Clinical Trial Management System

## 2.2 Aplikace pro práci s dokumenty

Aplikacemi pro práci s dokumenty jsou myšleny například aplikace Word a Excel z kancelářského balíku Microsoft Office či Dokumenty a Tabulky z Google Docs. Aplikace mají téměř totožné základní funkce, tudíž je nebudu v porovnání rozlišovat. Jsou dostupné téměř v každém jazyce a většina lidí se s nimi setkala a umí je alespoň základně používat. Používání umožňují jak online, tak offline a jeden soubor zároveň může upravovat více lidí. Využívají se téměř v každém průmyslu, včetně klinických výzkumů. Jsou výborné pro správu dat, můžeme zde omezeně programovat, využívat různé validace, ale nejsou navržené pro správu celé klinické studie.

Jedním z hlavních problému těchto aplikací je bezpečnost. Aplikace mají velmi omezenou kontrolu oprávnění, chybějí zde oproti EDC[4] systému uživatelské role. Části dokumentů je sice možné uzamknout heslem, ale pouze pro všechny uživatele anebo pro žádné. Logování je taktéž velmi omezené, nachází se zde historie změn, ale pro kontrolu změn v klinické studii nedostačující. Můžeme narušit konzistenci studie, jelikož lze lehce porušit časovou posloupnost výzkumu. Pokud je studie velká, může také nastat, že se přestaneme ve studii orientovat. Potřebujeme mít různé soubory pro různé typy aktivit a bude nám trvat delší dobu, než najdeme konkrétní dokument, oproti EDC systémů, kde máme vše na jednom místě.

I přesto existuje mnoho vytvořených užitečných šablon, které jsou strukturovány pro klinické studie a můžeme je zdarma využívat. Jsou k dispozici například na webu UC Davis Health centra[5]. Pro Google Docs Tabulky existují různé rozšíření pro klinické studie. Jedno z nich je OntoMaton[6], který umožňuje vyhledávání ontologických výrazů z webů Bioportal[7], LOV[8] a OLS[9] a přidávání anotací k jednotlivým buňkám.

## 2.3 Aplikace pro klinické studie

EDC je systém, který nahrazuje tradiční sběr dat pomocí pCRF za eCRF. Prostřednictvím grafického rozhraní umožňuje sbírat klinická data v elektronické formě. Mnoho společností nabízí EDC spolu s integrovaným CTMS, což je nástroj, který se zaměřuje na provozní část výzkumu, jako je například plánování subjektů, správa financí a hlášení o průběhu studie.

### 2.3.1 REDCap

Research Electronic Data CAPture neboli REDCap[10] je webová a mobilní aplikace určena ke sběru nejen klinických dat pro prospektivní i retrospektivní výzkumné studie vytvořená na univerzitě v městě Vanderbilt. Webová aplikace je napsána v jazyce PHP a využívá MySQL databázi. Není open-source, ale je poskytována zdarma pro nekomerční účely. Aplikaci používá přes 630000 uživatelů v téměř půl milionu studiích ve více než 2600 institucích.

V REDCap můžeme vytvořit prázdný, či naimportovat již existující projekt v XML nebo lze využít existujících šablon. eCRF je možné naimportovat ve formátu CSV nebo vytvořit pomocí grafického rozhraní, kde je možné nadefinovat prvky ve formuláři spolu s různými validacemi. Data lze překontrolovat i zpětně pomocí modulu, který zkontroluje například, zda jsou všechna pole ve studiích vyplněna, či zda

[4] EDC - Electronical data capture

[5] Dostupné na adrese: `http://www.ucdmc.ucdavis.edu/clinicaltrials/StudyTools/StudyTools.html` k 12.11.2017

[6] Dostupné na adrese: `https://github.com/ISA-tools/OntoMaton` k 12.11.2017

[7] Dostupné na adrese: `https://bioportal.bioontology.org/` k 12.11.2017

[8] Dostupné na adrese: `http://lov.okfn.org/dataset/lov/` k 12.11.2017

[9] Dostupné na adrese: `https://www.ebi.ac.uk/ols/index` k 12.11.2017

[10] Dostupné na adrese: `https://projectredcap.org/` k 13.11.2017

splňují zvolené požadavky. Mezi další moduly patří randomizace. Také je možné definovat vlastní uživatelské role. Projekt, formuláře i záznamy je možné vyexportovat ve formátu XML nebo CSV. REDCap dále nabízí možnost zaslepení studie, vytváření anket, porovnání dvou záznamů, nahrávání dokumentů a obrázků, chat, historii změn a různé statistiky.

Dle mého názoru není REDCap příliš graficky zdařilý a uživatelsky přívětivý, dělalo mi problém se v něm orientovat. Na druhou stranu obsahuje velké množství video tutoriálů, které vysvětlují, jak téměř každá část aplikace funguje. Nedostatky nástroje lze řešit pomocí různých rozšíření. Je dostupný pouze v anglickém jazyce, překlad lze zajistit pouze při určité programátorské zdatnosti.

Analýza proběhla v listopadu 2017 v demo aplikaci[11] na verzi 7.22.

### 2.3.2 OpenClinica

OpenClinica[12] je webová aplikace určena ke sběru klinických dat pro prospektivní i retrospektivní studie. Frontend aplikace je vytvořen pomocí JSP[13], backend v jazyce Java s PostgreSQL databází. Nabízí se ve dvou verzích. Komunitní edice je poskytována zdarma pod licencí GNU LGPL[14] a je nutné ji provozovat na svém vlastním serveru. Oproti tomu enterprise edice stojí více než půl milionu korun ročně[15], ale obsahuje více funkcí, plnou podporu a je možné ji provozovat jak na svém serveru, tak na serveru společnosti. OpenClinica používá přes 24 000 lidí ve více než 3 000 studiích z více než 100 zemí po celém světě.

Mezi funkce patří například vyhledávání záznamů podle id, matice subjektů, historie změn a kalendář pro plánování návštěv pacientů. Enteprise edice obsahuje navíc randomizaci, zaslepení studie a zasílání dotazníků subjektům k vyplnění. V aplikaci jsou tři typy uživatelů, technický administrátor, byznys administrátor a uživatel. Pro vytvoření eCRF je potřeba stáhnout si šablonu ve formátu XLS, následně ji upravit například v aplikaci Microsoft Excel a poté naimportovat zpět do aplikace. Pokud chceme formulář s různými validacemi a automatickými výpočty, je nutné je nadefinovat přímo v šabloně, avšak toto dle mého názoru vyžaduje určité programátorské znalosti. Import a export dat je možný ve formátech XML a XLS.

Neměl jsem možnost nástroj OpenClinica vyzkoušet, jelikož nenabízí demo aplikaci dostupnou na webu a nepodařila se mi zprovoznit ani na mém počítači. Analýza tedy proběhla na základě informací dostupných na webu a v rozsáhlé dokumentaci[16]v listopadu 2017 na verzi 3.13.

### 2.3.3 ClinCapture

ClinCapture[17] je webová aplikace určena ke sběru klinických dat pro prospektivní a retrospektivní studie. Vychází z aplikace OpenClinica verze 3.13, která je napsána v jazyce Java s PostreSQL databází. Je vytvořena společností Clinovo, která chtěla vytvořit aplikaci, která zlepší kvalitu kódu jádra aplikace OpenClinica, zpřehlední aplikaci a přidá nové funkce [20]. ClinCapture je nabízeno jak ve verzi zdarma pod licencí GNU LGPL[18], tak v placené verzi, která umožňuje například více studií, lepší zabezpečení, aktualizace, vylepšené CRF a překlad do více jazyků. Verzi zdarma je nutné provozovat na svém vlastním serveru a lze mít pouze 1 studii na 12 měsíců. Placenou verzi je možné provozovat na serverech společnosti, ale stojí od 2200$ na měsíc, záleží na počtu studií.

[11] Dostupné na adrese: `https://redcapdemo.vanderbilt.edu/` k 13.11.2017
[12] Dostupné na adrese: `https://www.openclinica.com/` k 14.11.2017
[13] JSP - JavaServer Pages
[14] Více informací dostupných na adrese: `https://www.gnu.org/licenses/lgpl-3.0.html` k 12.11.2017
[15] Cena za enterprise verzi je zhruba 30000$. Zjištěno po e-mailové komunikaci.
[16] Dostupné na adrese: `https://docs.openclinica.com/` k 14.11.2017
[17] Dostupné na adrese: `https://clincapture.com/` k 12.11.2017
[18] Více informací dostupných na adrese: https://www.gnu.org/licenses/lgpl-3.0.html k 12.11.2017

ClinCapture nabízí již vytvořené eCRF nebo si můžeme vytvořit vlastní pomocí grafického rozhraní, kde je možné nadefinovat políčka ve formuláři spolu s různými validacemi. Import a export dat z eCRF je možný ve formátu XML. Celou studii není možné vyexportovat. Obsahuje také různé statistiky, vyhledávání záznamu podle id, historii změn, matice subjektů a workflow, které graficky znázorňuje například proces vyplňování eCRF. Mezi další funkce patří ukládání částečných dat, čili si můžeme uložit záznam i bez vyplnění všech potřebných polí, a dvojí zadávání dat, což znamená, že eCRF je nutno vyplnit dvakrát, buď jedním uživatelem s rozestupem minimálně 12 hodin, anebo dvěma uživateli. Data jsou poté porovnána a jsou zobrazeny kolize. ClinCaptue rozlišuje dva typy uživatelů, administrátora a uživatele. Ty se potom dělí na dalších 9 uživatelských rolí, a to na administrátora systému, administrátora studie, sponzora, koordinátora výzkumu a další. Placená verze navíc obsahuje například kalendář, randomizaci, zaslepení studie, podporu lékařských číselníků a CRF masking, což znamená, že administrátor studie může nějaké eCRF skrýt pro určité uživatelské role.

Dle mého názoru je aplikace vcelku složitá, je komplikované ji nastavit a naučit se s ní pracovat. Na druhou stranu existuje rozsáhlá dokumentace[19] a mnoho videí[20], které vysvětlují, jak téměř každá část aplikace funguje.

Analýza proběhla v listopadu 2017 v demo aplikaci[21] na verzi 2.1.15.19.

### 2.3.4 CastorEDC

CastorEDC[22] je webová aplikace určená ke sběru nejen klinických dat pro prospektivní a retrospektivní studie. Je poskytována zdarma pro malé nekomerční studie, buď s 125 subjekty v prospektivní studii na 12 měsíců, anebo s 150 subjekty v retrospektivní studii na 12 měsíců, pro komerční účely či pro větší nekomerční studii je poskytována za poplatek. Aplikace běží na serveru společnosti a využívá jí více než 10000 uživatelů.

Aplikace nabízí vytvoření eCRF pomocí grafického rozhraní, kde lze nadefinovat políčka ve formuláři spolu s různými validacemi. eCRF můžeme také stáhnout z Castor Form Exchange a poté je naimportovat do projektu. Také je možné naimportovat a vyexportovat celý projekt, eCRF i jednotlivá data v XML formátu. Aplikace dále nabízí historii změn, matici subjektů, statistiky, vyhledávání podle čísla záznamu a různé filtrování. Uživatelské role je možné vytvořit vlastní nebo použít 3 existující. Za poplatek je možné přikoupit randomizaci spolu se zaslepením studie, premium podporu a dotazníky, které je možné rozeslat subjektům studie.

Dle mého názoru je aplikace uživatelsky přívětivá. Uživatelské rozhraní je jednoduché, intuitivní a přehledné. Pokud něco není jasné, je dostupná příručka[23], kde je vše popsáno i s obrázky.

Analýza proběhla v listopadu 2017 v online aplikaci[24] na verzi 2017.8.

### 2.3.5 OpenMRS

OpenMRS[25] je webová aplikace, která slouží jako elektronická zdravotní karta pacienta, je vhodná pro prospektivní i retrospektivní klinické studie. Aplikace je napsána v jazyce Java pod licencí Mozilla

[19] Dostupné na adrese: `http://docs.clincapture.com/clincapture-user-guide-v2-2/` k 15.11.2017
[20] Dostupné na adrese: `http://videos68.com/channel/UCZTFTSYiYPlVoO4I5Fh-\_3Q` k 15.11.2017
[21] Dostupné na adrese: `https://studio.clincapture.com/` k 15.11.2017
[22] Dostupné na adrese: `https://www.castoredc.com/` k 16.11.2017
[23] Dostupné na adrese: `https://support.castoredc.com/portal/kb/` k 16.11.2017
[24] Dostupné na adrese: `https://data.castoredc.com/` k 16.11.2017
[25] Dostupné na adrese: `https://openmrs.org/` k 20.11.2017

Public License[26], je tedy open-source a poskytována zdarma. Používá se po celém světě[27] a je přeložena do pěti jazyků, do angličtiny, francouzštiny, španělštiny, portugalštiny a italštiny.

OpenMRS umožňuje změnit písmo a barvu jednotlivých částí aplikace. eCRF s validacemi lze vytvořit přímo v aplikaci pomocí znalosti HTML, CSS a jazyku JavaScript anebo pomocí OpenMRS XForms Designer aplikace. Uživatelské role lze nadefinovat vlastní nebo je možné si vybrat z mnoha existujících. Mezi další funkce patří historie změn, vyhledávání podle jména nebo id, vytisknutí záznamu o návštěvě subjektu, kalendář návštěv subjektů a slovník pojmů, který je možný rozšířit a slouží k nápovědě při vyplňování formuláře. K dispozici je mnoho modulů, které dodávají další funkce aplikaci, například vylepšené randomizace, vytváření formulářů, import a export dat.

Dle mého názoru je aplikace uživatelsky přívětivá. Ovládání je velmi jednoduché a intuitivní. Pokud něco není jasné, je možné použít dokumentaci[28], která ovšem moc přehledná není.

Analýza proběhla v listopadu 2017 v online aplikaci[29] na verzi 2.0.5.

## 2.4 Shrnutí

V tabulce 2.1 je možné vidět porovnaní vybraných vlastností analyzovaných existujících řešení aplikací.

Tabulka 2.1: Shrnutí důležitých vlastností aplikace

| | REDCap | OpenClinica | ClinCapture | CastorEDC | OpenMRS |
|---|---|---|---|---|---|
| Platforma | Web, Mobilní aplikace | Web | Web | Web | Web |
| Responzivnost | Ne | Ne | Ne | Ne | Ano |
| Vytvoření eCRF | V grafickém rozhraní | V Excelu | V grafickém rozhraní | V grafickém rozhraní | Naprogramováním, v externí aplikaci |
| Uživatelské role | Možné vytvoření vlastních | Běžné pro studie | Běžné pro studie | Možné vytvoření vlastních | Možné vytvoření vlastních |
| Import / export dat | Veškerá data | Veškerá data | eCRF, záznamy | Veškerá data | Pouze import záznamů* |
| Historie změn | Ano | Ano | Ano | Ano | Ano |
| Randomizace | Ano* | Ano** | Ano** | Ano** | Ano* |

* - pomocí přídavného modulu, ** - placená funkcionalita

Z této analýzy se bude vycházet při návrhu výsledné aplikace. Slouží k zjištění důležitých funkcí a zároveň k vyvarování se chyb a nedostatků existujících aplikací. V prvním řádku jsou analyzované aplikace, v prvním sloupci potom důležité vlastnosti aplikací.

[26]Dostupné na adrese: `https://support.castoredc.com/portal/kb/` k 20.11.2017

[27]Mapa, kde se OpenMRS používá: `https://atlas.openmrs.org/`

[28]Dostupné na adrese: `https://wiki.openmrs.org/` k 20.11.2017

[29]Dostupné na adrese: `https://demo.openmrs.org/` k 20.11.2017

# Kapitola 3

# Aplikace StudyManager

Tato kapitola popisuje převzatý stav aplikace StudyManager spolu s funkcemi a použitými technologiemi a v neposlední řadě způsob generování formuláře.

## 3.1 Popis

StudyManager je webová aplikace vyvíjená na ČVUT na Fakultě elektronické panem Mgr. Miroslavem Blaškem, Ph.D. a panem Ing. Martinem Ledvinkou. Je to aplikace určená ke sběru klinických dat pro retrospektivní klinické studie. K tomu, aby bylo možné StudyManager používat, je potřeba nástroj dodávající specifikaci eCRF v formě JSON-LD. K tomu je možné využít například aplikaci SPipes (viz sekce 3.3.5). StudyManager je statická část aplikace, která je pro každou studii stejná. Naopak SPipes je část dynamická, jež se pro každou studii chová jinak. Je zpravidla tvořena datovým proudem, jenž je definován pomocí SPipes skriptu, který integruje různé typy ontologií. Příkladem je ontologie pro reprezentaci formulářů, kterou využívá i SForms, ontologie pro biomedicínské výzkumy[1] nebo pro medicínské slovníky[2]

Aplikace je koncipována tak, že zadavatel studie vytvoří každé zdravotnické instituci, která se chce zapojit do studie, v systému instituci. Do této instituce poté přidá doktory, kteří mohou vytvářet záznamy o pacientech. Doktoři mají přístup pouze ke své instituci a uživatelům a záznamům o pacientech v rámci jejich instituce.
Nástroj je využíván například při těchto scénářích:

- Sekretářka vytváří novou instituci
- Sekretářka vytváří nový účet zkoušejícímu
- Zkoušející vytváří nový záznam o pacientovi
- Statistik prochází záznamy o pacientech a sleduje průběh studie
- Statistik vyhodnocuje studii

[1] Například OBI - dostupné na adrese `http://obi-ontology.org/` k 21.5.2018
[2] Například NCI Thesaurus - dostupné na adrese `https://ncit.nci.nih.gov/ncitbrowser/` k 21.5.2018

## 3.2 Počáteční stav

### 3.2.1 Funkce aplikace

- Uživatelé
    1. Zobrazit / upravit / smazat profil uživatele
    2. Vytvořit nového uživatele v roli administrátora či doktora
- Instituce
    1. Vytvořit / zobrazit / upravit / smazat instituci
    2. Zobrazit uživatele instituce
    3. Zobrazit záznamy o pacientech instituce
- Pacienti
    1. Zobrazit záznamy o pacientech
    2. Vytvořit / zobrazit / upravit / smazat záznam o pacientovi
- Ostatní
    1. Přihlásit se
    2. Odhlásit se
    3. Zobrazit hlavní menu
    4. Zobrazit a upravit svůj profil

### 3.2.2 Uživatelské role

V systému se momentálně nacházejí tři uživatelské role.

- Administrátor - Hlavní správce systému, který má přístup ke všem funkcím v rámci aplikace. Funkce této role je možné nalézt v tabulce 3.1.
- Doktor - Uživatel, který může v rámci své instituce provádět výzkum. Funkce této role je možné nalézt v tabulce 3.2.
- Nepřihlášený uživatel - Uživatel, jenž se může pouze přihlásit.

Tabulka 3.1: Zobrazení funkcí uživatelské role Administátor

| Uživatelská role | Kategorie | Funkce |
|---|---|---|
| Administrátor | Uživatelé | Zobrazit / upravit / smazat profil uživatele |
| | | Vytvořit nového uživatele jakékoli role |
| | Instituce | Vytvořit / zobrazit / upravit / smazat instituci |
| | | Zobrazit všechny uživatele jakékoli instituce |
| | | Zobrazit všechny záznamy o pacientech jakékoli instituce |
| | Pacienti | Zobrazit všechny záznamy o pacientech |
| | | Vytvořit / zobrazit / upravit / smazat záznam o pacientovi |
| | Ostatní | Zobrazit hlavní menu |
| | | Zobrazit a upravit svůj profil |
| | | Odhlásit se |

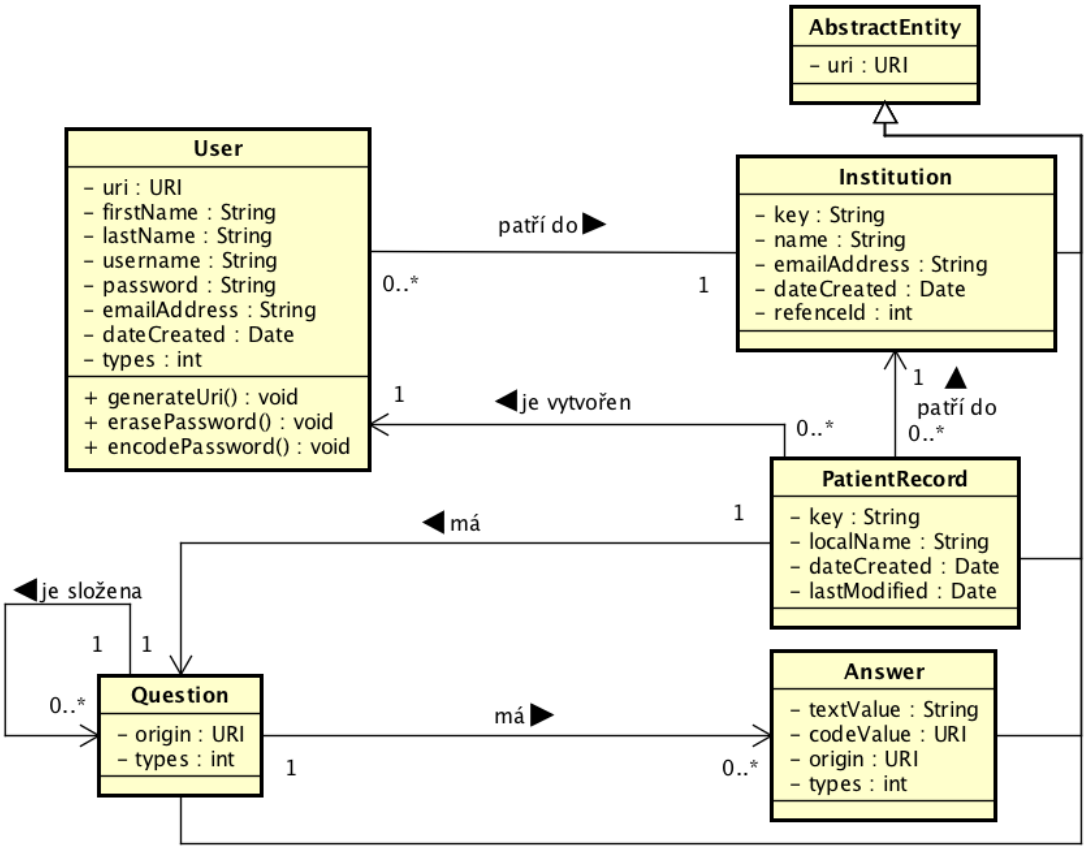
Tabulka 3.2: Zobrazení funkcí uživatelské role Doktor

| Uživatelská role | Kategorie | Funkce |
|---|---|---|
| Doktor | Instituce | Zobrazit jakoukoli instituci |
| | | Zobrazit všechny uživatele své instituce |
| | | Zobrazit všechny záznamy o pacientech své instituce |
| | Pacienti | Zobrazit všechny záznamy o pacientech své instituce |
| | | Vytvořit / zobrazit / upravit / smazat záznam o pacientovi |
| | Ostatní | Zobrazit hlavní menu |
| | | Zobrazit a upravit svůj profil |
| | | Odhlásit se |

### 3.2.3 Diagram tříd

Na obrázku 3.1 je možné vidět vnitřní strukturu aplikace StudyManager. eCRF je reprezentováno pomocí stromu otázek, který je vyjádřen třídou „Question“. Atribut origin potom slouží k identifikaci, z které otázky „Question“ nebo „Answer“ vzniklo.

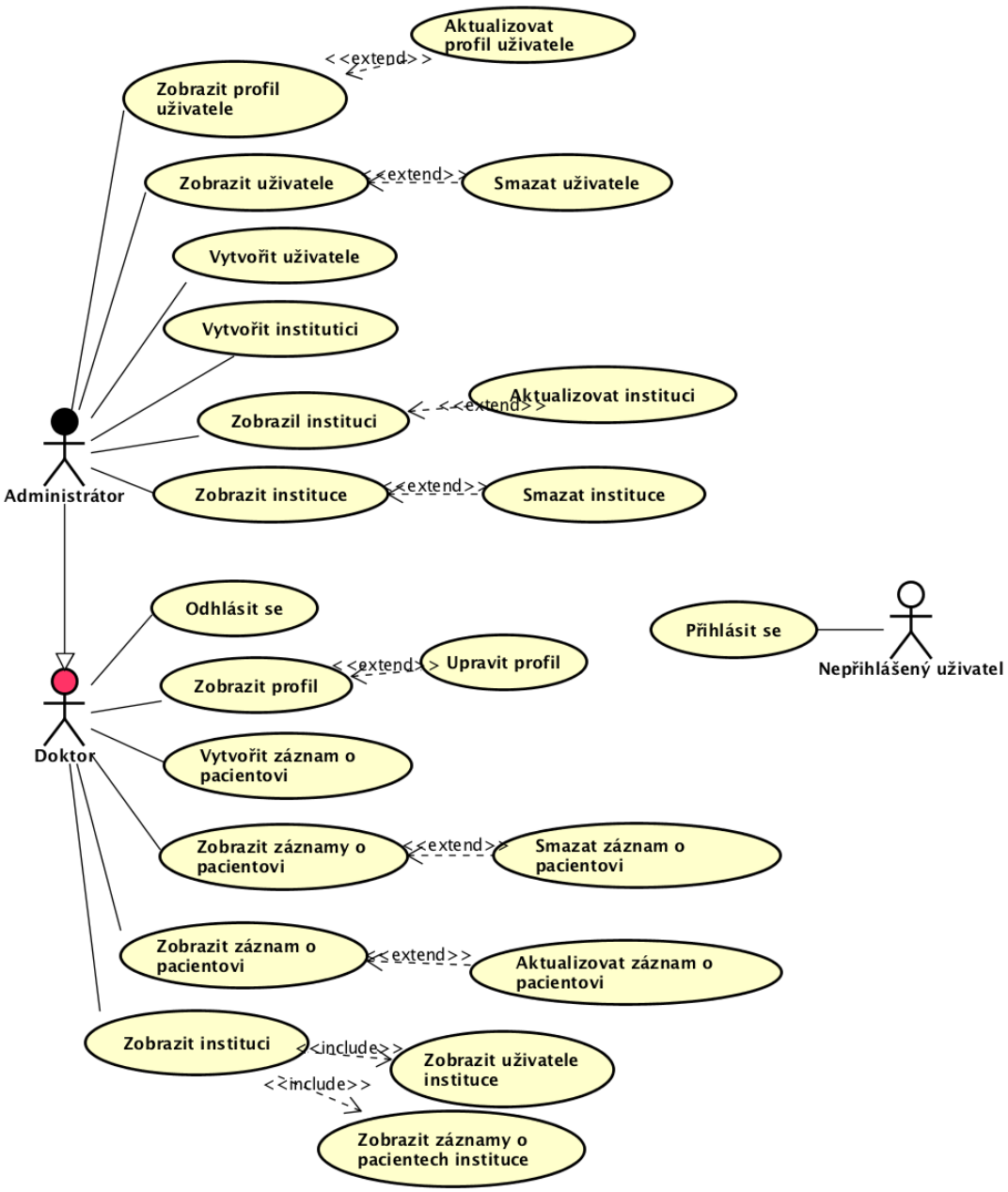
Obrázek 3.1: Struktura aplikace StudyManager




UML 2.0 - Diagram aktivit [14]

### 3.2.4 Případy užití

Obrázek 3.2: Případy užití před rozšířením aplikace

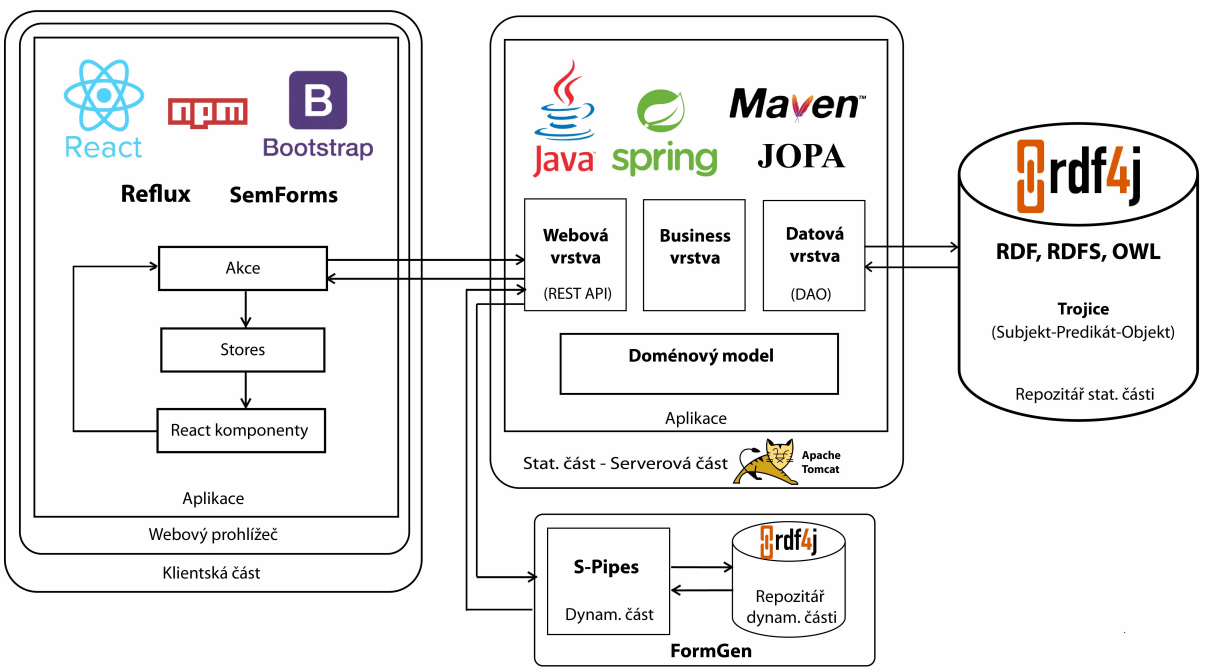



UML 2.0 - Diagram případů užití [14]

## 3.3 Použité technologie

### 3.3.1 Architektura aplikace

Obrázek 3.3: Architektura počátečního stavu aplikace




### 3.3.2 Klientská část

StudyManager je webová aplikace. Klientská část je napsána v jazyce JavaScript spolu s HTML a CSS. Pro zefektivnění práce se používá JavaScriptová knihovna React doplněna o knihovnu Reflux a Bootstrap. Pro vytvoření eCRF se používá knihovna SForms, která z dat poskytlých dynamickou částí aplikace vytvoří interaktivní formulář.

#### 3.3.2.1 React

Knihovna React je mocný nástroj programovacího jazyka JavaScript, který umožňuje vytvářet Single Page aplikace, což znamená, že je možné změnit jen část webu bez opětovného načítání stránky. Je proto vhodný pro aplikace, které vyžadují časté aktualizace jednotlivých dat [21].

#### 3.3.2.2 Reflux

Reflux je knihovna sloužící k řízení jednosměrného toku dat. Definuje rozložení struktury aplikace na akce, story a komponenty, tedy využívá návrhového vzoru MVC[3]. Komponenty vytvářejí akce, které poté putují do stores, což jsou jakési sklady na data. Ty na základě akce tato data mění a díky změně dat se komponenty aktualizují a zobrazují aktuální data [22].

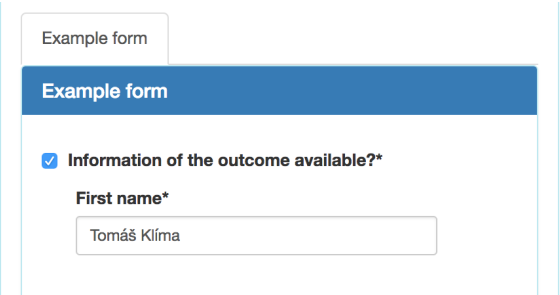
[3]MVC - Model-View-Controller. Oblíbený návrhový vzor, který rozdělujeme aplikaci do třech nezávislých vrstev [14]. Model - datový model aplikace, View - uživatelské prostředí, Controller - řídící logika.

#### 3.3.2.3 SForms

SForms je knihovna vytvořená panem Ing. Martinem Ledvinkou a Mgr. Miroslavem Blaškem, Ph.D. z ČVUT FEL. Umožňuje vizualizovat dynamický formulář z poskytlých ontologických dat ve formátu JSON-LD, ve kterých jsou definované jak konkrétní otázky pro sběr dat, tak i způsob chování formuláře. Zdrojový kód jednoduchého formuláře spolu s popisem je možné najít v příloze v sekci D.3. Mezi funkce, které SForms v rámci dynamického formuláře umožňuje, patří:

- rozdělení vyplňování formuláře do několika wizardů[4], které jsou zobrazovány na základě vyplněných dat.
- zobrazování a skrývaní polí formuláře na základě vyplněných dat.
- různé typy polí formuláře.

Obrázek 3.4: Ukázka jednoduchého dynamicky generovaného formuláře




#### 3.3.2.4 Bootstrap

Bootstrap je HTML, CSS a JavaScriptová knihovna pro vývoj responzivních webů. Obsahuje předem definované nástroje, které usnadňují a zefektivňují tvorbu webu [23].

### 3.3.3 Serverová část

Pro serverovou část aplikace byl vybrán programovací jazyk Java s frameworkem Spring a nástrojem Maven.

#### 3.3.3.1 Spring

Spring je Java framework, který usnadňuje tvorbu rozsáhlých Java aplikací. Mezi hlavní výhody použití patří [24]:

- Snadno integrovatelný s mnoha frameworky a knihovnami
- Nevyžaduje aplikační server
- Využívá návrhový vzor Inversion of Control [5]

[4] Wizard - uživatelské rozhraní, které je rozdělené do několika sekcí, které uživatel prochází krok za krokem
[5] Inversion of Control - Přesouvá zodpovědnost vytváření a provázání objektů z aplikace na framework

#### 3.3.3.2 Maven

Maven je nástroj, který se stará o sestavení aplikace a správu závislostí. Kontroluje a zajišťuje přítomnost všech definovaných knihoven [25].

### 3.3.4 Datová reprezentace a samotná databáze

Nynější web je obrovská knihovna lidmi psaných dokumentů, ve které je čím dál tím těžší nalézt relevantní informace. Pro stroje je velmi obtížné tyto informace vyhledávat a automaticky zpracovávat. Z tohoto důvodu se zrodil Sémantický Web [26], kterého principy využívá i tato aplikace.

#### 3.3.4.1 Sémantický Web

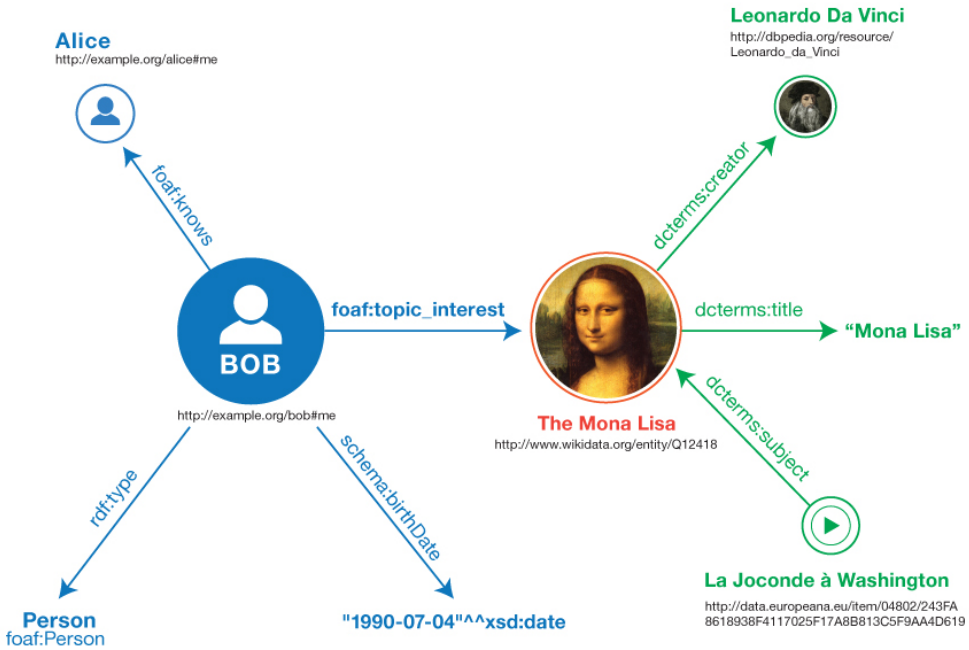
Semantický Web se snaží vylepšit aktuální web tak, aby informace nebyly srozumitelné pouze pro lidi, ale i pro stroje. Účelem je popsat jakékoli věci a vztahy mezi nimi pomocí ontologií [27].

#### 3.3.4.2 Ontologie

Ontologie jsou termíny, které se používají k popisu reálných objektů pomocí instancí, tříd, atributů a vazeb [28]. Data jsou reprezentována jako trojice: Subjekt-Predikát-Objekt, například Umělec (subjekt) - vytváří (predikát) - umělecké dílo (objekt).

K reprezentaci objektů se využívají jazyky OWL[6], RDF[7] a RDFS[8]. Na obrázku 3.5 je možné vidět ukázku ontologie.

Obrázek 3.5: Ukázka ontologie




Zdroj: `https://www.w3.org/TR/rdf11-primer/` k 17.5.2018

[6]OWL - Web Ontology Language, více informací na `https://www.w3.org/OWL/` k 17.5.2018
[7]RDF - Resource Description Framework, více informací na `https://www.w3.org/RDF/`k 17.5.2018
[8]RDFS - RDF Schema, více informací na `https://www.w3.org/TR/rdf-schema/` k 17.5.2018

#### 3.3.4.3 Reprezentace dat

Identifikace objektů je zajištěna pomocí technologie URI[9], díky které můžeme popsat jakýkoli web, dokument, obrázek, video, ale i knihu v reálném světě [28].

Ontologická data jsou reprezentována pomocí jazyku OWL, RDF, RDFS, což jsou ontologické jazyky, který používají XML syntaxi.

#### 3.3.4.4 RDF

RDF je W3C[10] standard pro modelování dat. Pomocí této metody můžeme pomocí trojic popsat jakákoli data, která mohou být identifikovaná pomocí URI [29]. Zdrojový kód 3.1 [30] pod tímto textem zachycuje obrázek 3.5 ve formátu RDF/XML, což je jedna z možností serializace RDF.

Zdrojový kód 3.1: Ukázka RDF

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF xmlns:dcterms="http://purl.org/dc/terms/"
   xmlns:foaf="http://xmlns.com/foaf/0.1/"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:schema="http://schema.org/">
   <rdf:Description rdf:about="http://example.org/bob#me">
   <rdf:type rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
   <schema:birthDate rdf:datatype="http://www.w3.org/2001/XMLSchema#date">1990-07-04
   </schema:birthDate> <foaf:knows rdf:resource="http://example.org/alice#me"/>
   <foaf:topic_interest rdf:resource="http://www.wikidata.org/entity/Q12418"/>
   </rdf:Description>
   <rdf:Description rdf:about="http://example.org/bob#me">
   <rdf:type rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
   <schema:birthDate
       rdf:datatype="http://www.w3.org/2001/XMLSchema#date">1990-07-04</schema:birthDate>
       <foaf:knows rdf:resource="http://example.org/alice#me"/><foaf:topic_interest
       rdf:resource="http://www.wikidata.org/entity/Q12418"/></rdf:Description>
</rdf:RDF>
```

#### 3.3.4.5 JSON-LD

JSON-LD neboli JavaScript Object Notation[11] for Linked Data[12] je metoda serializace Linked Data do formátu JSON. Umožňuje, aby existující JSON mohl být s minimálními změnami interpretován jako RDF a mohl uchovávat význam dat tak, aby jim rozuměl i počítač [31]. Zdrojový kód 3.2 [30] zachycuje obrázek 3.5 ve formátu JSON-LD.

[9] URI - Uniform Resource Identifier, více informací na `https://www.w3.org/TR/uri-clarification/` k 16.5.2018

[10] W3C - World Wide Web Consortium, dostupné na `https://www.w3.org/` k 17.5.2018

[11] JSON - datový formát

[12] Linked Data - Způsob propojení dat na webu

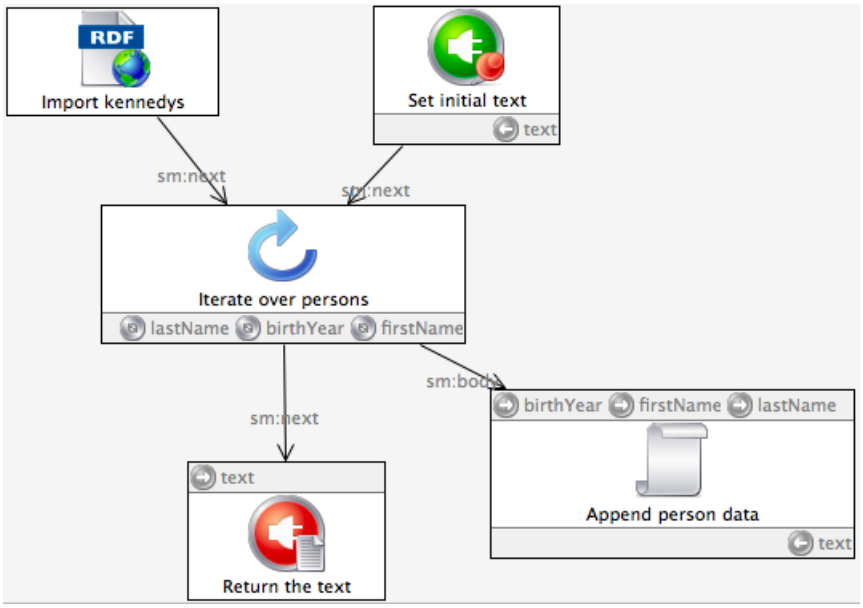
Zdrojový kód 3.2: Ukázka JSON-LD

```
{
  "@context": "example-context.json",
  "@id": "http://example.org/bob#me",
  "@type": "Person",
  "birthdate": "1990-07-04",
  "knows": "http://example.org/alice#me",
  "interest": {
    "@id": "http://www.wikidata.org/entity/Q12418",
    "title": "Mona Lisa",
    "subject_of": "http://data.europeana.eu/item/04802/243FA8618938F4117",
    "creator": "http://dbpedia.org/resource/Leonardo_da_Vinci"
  }
}
```

### 3.3.5 SPipes

SPipes je nástroj napsaný v jazyce Java na ČVUT FEL na katedře KBSS panem Mgr. Miroslavem Blaškem, Ph.D. a Ing. Petrem Křemenem, Ph.D. Je určen k vytváření sémantických proudů[13] pro zpracování dat. Jazyk SPipes je podmnožinou skriptovacího jazyka SPARQLMotion [32], který je založený na RDF s grafickým prostředím k vytváření a zobrazování proudů [33].

Obrázek 3.6: SPARQLMotion příklad skriptu




Zdroj: `http://sparqlmotion.org/` k 10.5.2018

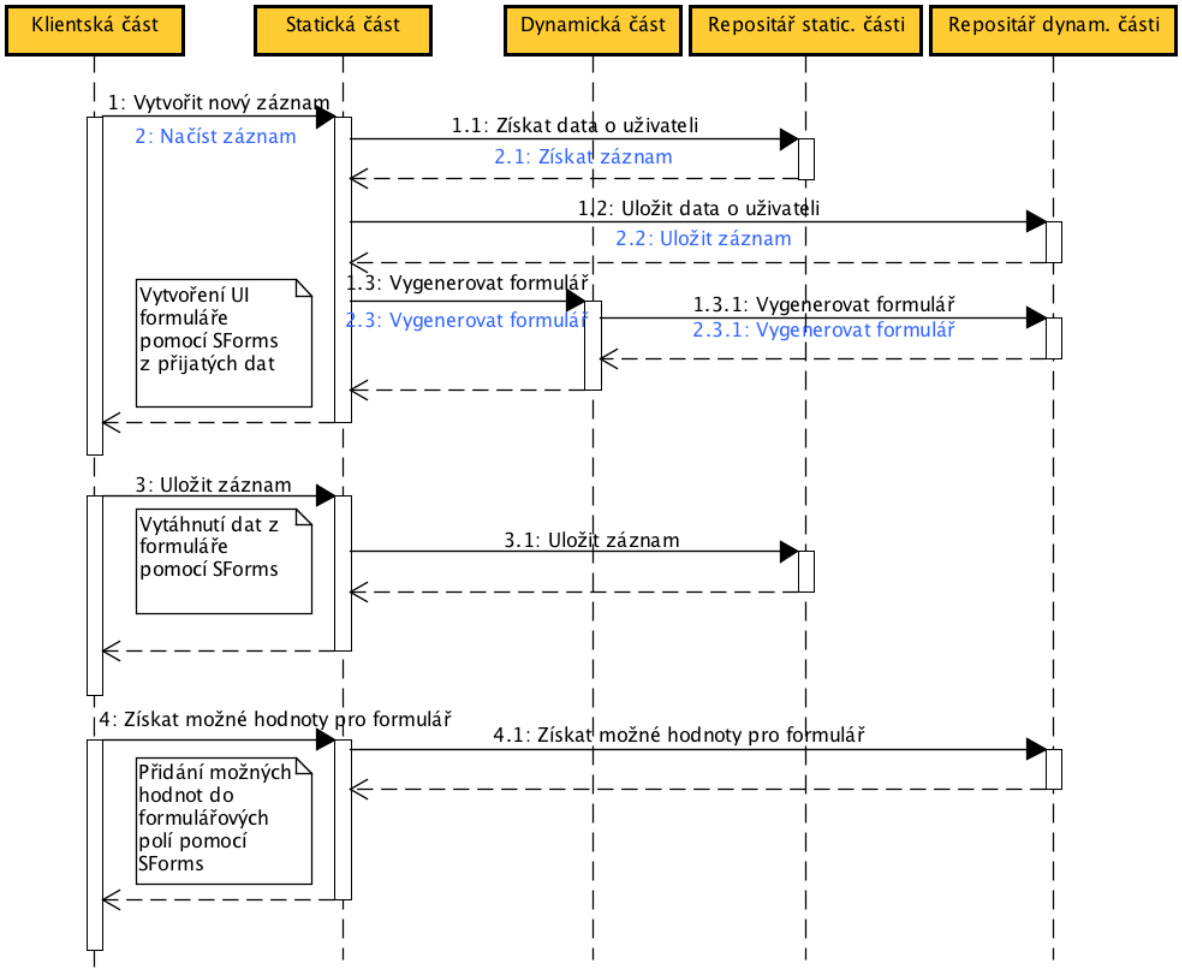
#### 3.3.5.1 Proces tvorby dynamicky generovaného formuláře

Na obrázku 3.7 je možné vidět využití nástroje SPipes. Znázorňuje, jak probíhá interakce v aplikaci StudyManager s dynamicky generovaným formulářem, který je obsluhován knihovnou SForms v

[13] Proud - Propojení jednotlivých kroků zpracovaní dat tak, že výstup jednoho kroku, je vstupem kroku dalšího [33]

klientské části aplikace a nástrojem SPipes, jakožto dynamickým nástrojem pro tvorbu dynamicky generovaných formulářů.

Obrázek 3.7: Interakce s dynamicky generovaným formulářem




modré zprávy vyjadřují další sekvenci shodnou s sekvencí 1
UML 2.0 - Sekvenční diagram [14]

#### 3.3.5.2 Databáze

Aplikace využívá k ukládání dat RDF4J Server[14], který obsahuje repositáře, v kterých je možné uchovávat ontologická data.

#### 3.3.5.3 JOPA

JOPA neboli Java OWL Persistence API je framework inspirovaný JPA[15], který slouží ke komunikaci mezi serverem aplikace a RDF4J Server. Zajišťuje automatické mapování ontologických dat na objekty.

[14]RDF4J - Více informací na adrese `http://rdf4j.org/` k 21.5.2018
[15]JPA - framework umožňující objektově relační mapování

Podporuje parametrizované SPARQL[16] dotazy do repositáře.

## 3.4 Shrnutí

Mezi problémy aplikace po stránce implementační patří velké množství různě závažných chyb, které je možné najít v příloze A. Tyto chyby je nutné pro bezpečné a uživatelsky přívětivé používaní opravit. Z hlediska používání se vyskytly spíše potíže s náročným obsluhováním aplikace. Například když administrátor vytváří zkoušejícím nové uživatelské účty, musí každému manuálně zaslat uživatelské jméno spolu s heslem. Dalším problémem je zapomenutí hesla od uživatelského účtu, kdy se uživatelé obraceli na administrátora s žádostí o zaslání jejich hesla.

[16] SPARQL - sémantický jazyk, který slouží pro dotazování dat uchovaných ve formátu RDF

# Kapitola 4

# Návrh řešení

Nástroj StudyManager má dle mého názoru, i přes drobné nedostatky, uspokojující základ pro podporu prospektivních studií, proto jsem se v rámci této práce rozhodl StudyManager dále rozvíjet a nevytvářet aplikaci novou.

## 4.1 Analýza požadavků na rozšíření nástroje StudyManager

Požadavky jsou rozděleny na funkční, nefunkční a na požadavky na zavedené řešení. Pro snadnější odkazování jsem každý požadavek opatřil identifikačním číslem. Dále jsem u každého požadavku určil, na základě analýzy z kapitoly 2 a zpětné vazby z již probíhajících studií využívajících StudyManager, prioritu pomocí metody MoSCoW [34]. Dále jsem navrhl řešení či odůvodnil, z jakého důvodu požadavek nebude analyzován a implementován. Pokud důvod není uveden, platí, že požadavek není důležitý pro momentální stav aplikace, protože neomezuje hlavní funkce systému. Pouze usnadňuje používání aplikace.
MoSCoW rozděluje požadavky do čtyř kategorií:

**M** Must have

- Kritické požadavky, bez kterých není možné či bezpečné aplikaci používat.

**S** Should have

- Důležité požadavky, bez kterých je možné aplikaci používat.

**C** Could have

- Žádoucí požadavky, ale nemusí na implementaci zbýt čas.

**W** Won't have

- Požadavky, na které čas nezbyde a budou splněny v příští verzi.

### 4.1.1 Funkční požadavky

#### 4.1.1.1 Usnadnění

**C FP1** Řazení dat podle atributů

1. Systém umožní uživateli seřadit uživatele podle jména, přezdívky, názvu instituce a e-mailu

2. Systém umožní uživateli seřadit instituce podle jména a kontaktního e-mailu
3. Systém umožní uživateli seřadit záznamy o pacientech podle id, identifikátoru pacienta, času poslední úpravy a stavu dokončení

**C FP2** Vyhledávání

1. Systém umožní uživateli vyhledat uživatele podle přezdívky a e-mailu
2. Systém umožní uživateli vyhledat instituce podle jména a kontaktního e-mailu
3. Systém umožní uživateli vyhledat záznam o pacientovi podle identifikátoru a aktuálním stavu

- Tento požadavek není důležitý pro momentální stav aplikace, jelikož uživatel může vyhledávat na stránce v prohlížeči pomocí klávesové zkratky Crtl + F.

**S FP3** Vytvoření, zobrazení a smazání uživatele přímo v instituci

1. Systém umožní vytvořit uživatele, který bude mít již přidělenou instituci
2. Systém umožní smazat uživatele přímo z detailu instituce, pokud nemá žádné záznamy o pacientech
3. Systém umožní zobrazit uživatele přímo v detailu instituce

- Uživatelů a institucí může být v rámci jedné studie mnoho, pro efektivnější práci je žádoucí, aby bylo možné manipulovat s uživateli přímo v detailu určité instituce.

**S FP4** Vygenerování uživatelského jména

1. Systém umožní vygenerování uživatelského jména při vytváření nového uživatele

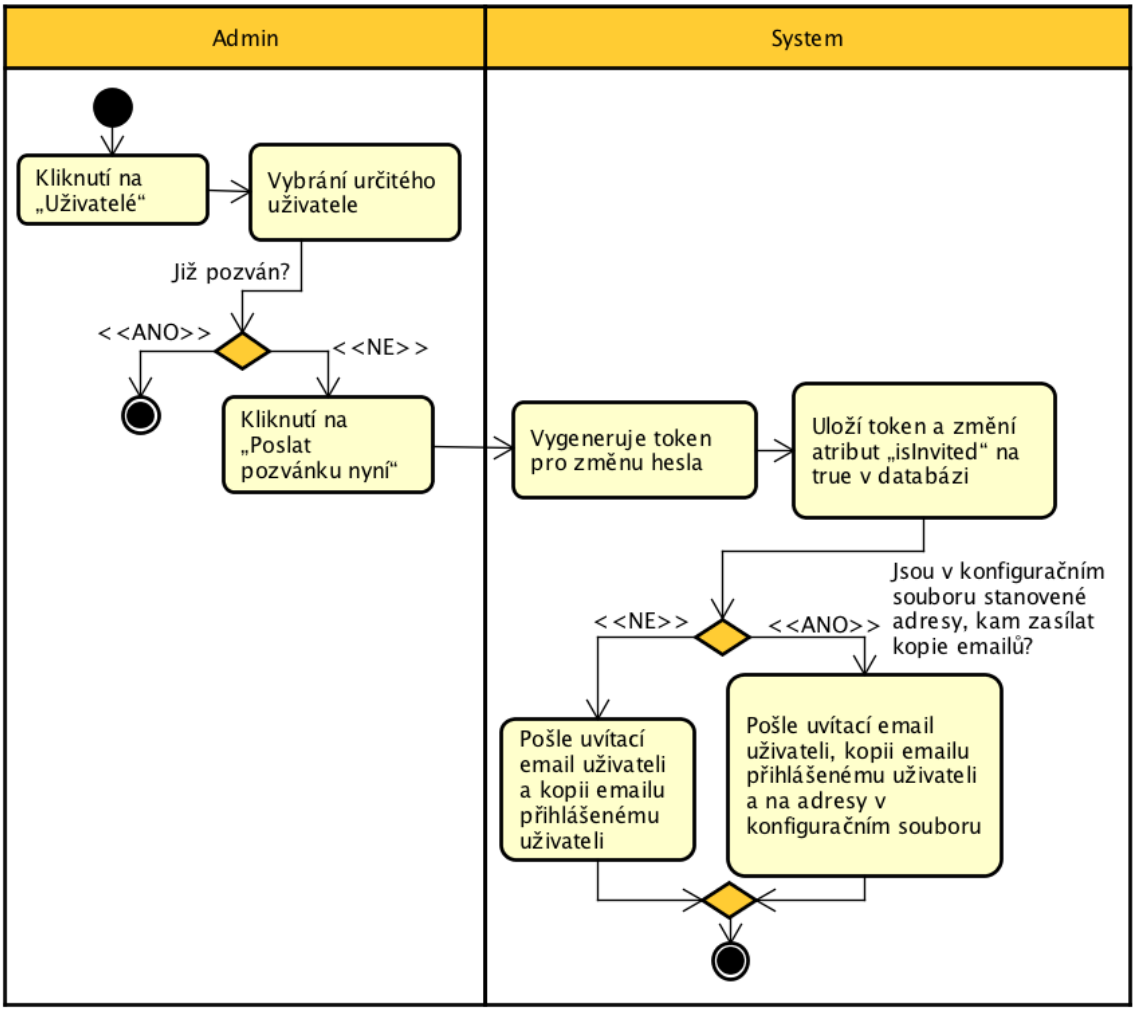
- Ze stejného důvodu jako u FP3 naimplementuji do systému funkci, která při kliknutí na tlačítko u uživatelského jména vygeneruje uživatelské jméno na základě zvolené role.

**M FP5** Pozvání nového uživatele do studie

1. Systém umožní administrátorovi pozvat uživatele do studie přes e-mail (viz obrázek 4.1)

- Tato funkcionalita usnadní administrátorovi práci, jelikož nebude muset každému novému uživateli zasílat uvítací email spolu s uživatelským jménem a heslem manuálně. Zároveň naplním jeden z cílů této práce, kterým je plánování, jelikož bude možné vše v aplikaci připravit a teprve poté do ní uživatele pozvat.

Obrázek 4.1: Průběh zaslání uživateli pozvánky do studie




UML 2.0 - Diagram aktivit [14]

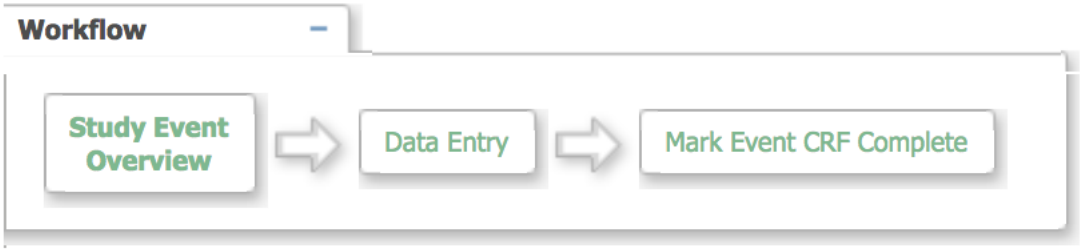
**C FP6** Matice subjektů

1. Systém zobrazí tabulku, která zobrazuje informace o aktuálním stavu jednotlivých záznamů o pacientech (viz obrázek 1.1 v kapitole 1)

**W FP7** Workflow

1. Systém uživateli graficky znázorní průběh procesů v aplikaci (viz obrázek 4.2)

Obrázek 4.2: Workflow v aplikaci ClinCapture




Zdroj: aplikace ClinCapture k 15.11.2017

**M FP8** Indikátory stavu

1. Systém umožní informovat uživatele o probíhajících, úspěšných a neúspěšných akcích

- Uživatel v současné verzi není na mnoha místech v aplikaci informován o tom, že se něco načítá nebo že nějaká akce úspěšně či neúspěšně proběhla. Dle heuristické analýzy od Jakoba Nielsna by měla aplikace vždy dát uživateli vědět, co se právě v systému odehrává [37].

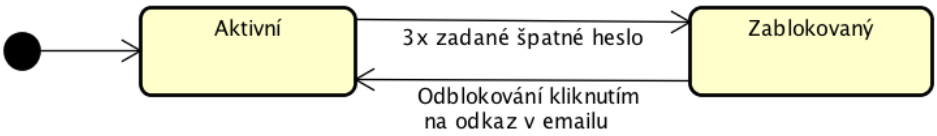
#### 4.1.1.2 Správa účtu a zabezpečení

**C FP9** Blokace účtu

1. Systém umožní zablokovat uživateli účet, pokud třikrát zadá špatné heslo
2. Systém umožní zaslat uživateli e-mail s odkazem na odblokování účtu

- Na obrázku 4.3 je vidět návrh přechodů mezi stavy aktivní a zablokovaný. Tento požadavek však nebude z časového důvodu implementován.

Obrázek 4.3: Přechody mezi stavy účtu




UML 2.0 - Stavový diagram [14]

**M FP10** Zapomenuté heslo

1. Systém umožní uživateli obnovit zapomenuté heslo k účtu pomocí e-mailu

- Resetování zapomenutého hesla patří mezi základní funkce každého systému, do kterého je možné se přihlásit.

**M FP11** Změna hesla

1. Systém umožní uživatelům změnit své heslo
2. Systém umožní administrátorovi systému změnit heslo všem uživatelům

- Změna hesla patří mezi základní funkce každého systému, do kterého je možné se přihlásit.

**S FP12** Notifikace přes e-mail

1. Systém umožní uživatele informovat o změnách v jeho profilu

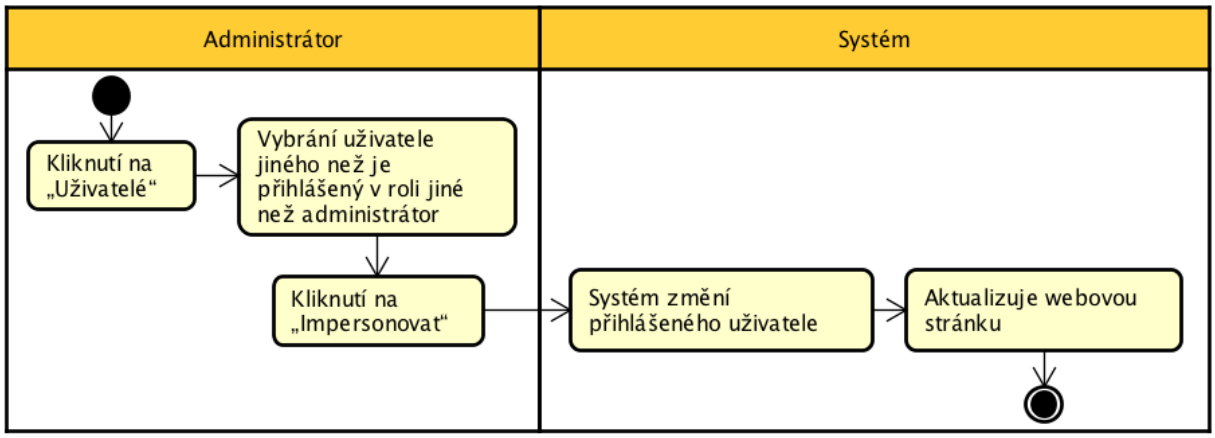
- Jedním z cílů bakalářské práce je rozšíření aplikace o notifikace uživatelů. Jako prostředek pro zpravení uživatelů jsem zvolil e-mail.

**S FP13** Impersonace

1. Systém umožní administrátorovi přihlásit se na jiného uživatele bez hesla

- Tato funkce je užitečná například, pokud uživatel zjistil nějakou chybu používáním aplikace. Administrátor se může na problém u uživatele podívat bez znalosti hesla a odstranit ho. Návrh průběhu impersonace je možné vidět na obrázku 4.4.

Obrázek 4.4: Průběh impersonace




UML 2.0 - Diagram aktivit [14]

**C FP14** Více uživatelských rolí

1. Systém umožní vytvořit uživatele v roli administrátora systému, administrátora instituce, doktora a hosta

- Dle zpětné vazby z již proběhlých studií pomocí StudyManager, se ukázalo, že rozšíření o nové role není kritické. Požadavek obnáší velké množství změn implementace aplikace, tudíž nebude z omezených časových možností analyzován a implementován.

#### 4.1.1.3 eCRF

**W FP15** Nástroj pro tvorbu formulářů

1. Systém umožní uživateli jednoduše vytvořit formulář pomocí grafického rozhraní

- Nástroj pro tvorbu dynamicky generovaných formulářů se týká dynamické části aplikace, proto nebude v rámci této práce analyzován.

**C FP16** Formuláře s validací a automatickými výpočty

1. Systém při vyplňování políčka formuláře zkontroluje správný formát
2. Systém při vyplňování dopočítá určité hodnoty

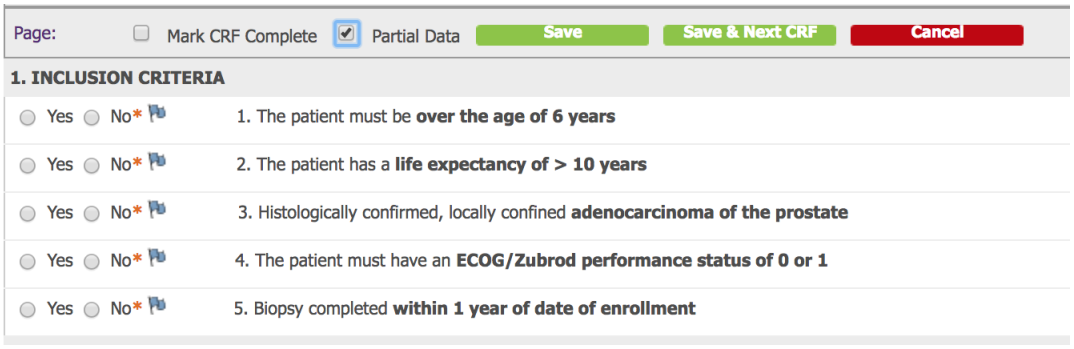
- Tento požadavek je možné řešit pomocí dynamické části aplikace.

**C FP17** Částečná data

1. Systém umožní uložit záznam o pacientovi, který nemá vyplněné všechny požadované části

- Tento požadavek je možné řešit pomocí dynamické části aplikace.

Obrázek 4.5: Částečná data v aplikaci ClinCapture




Zdroj: aplikace ClinCapture k 15.11.2017

**C FP18** Zpětná kontrola

1. Systém umožní zkontrolovat, zda jsou všechna data v formuláři validně vyplněna

- Tento požadavek je možné vyřešit pomocí dynamické části aplikace.

**W FP19** Dvojí zadávání dat

1. Systém umožní vyplnit jeden formulář dvakrát dvěma různými uživateli či v časovém odstupu 12 hodin a poté zobrazí rozdílnosti s možností úpravy a finálního uložení záznamu.

- Tento požadavek je možné řešit pomocí dynamické části aplikace.

**C FP20** Skrytí části či celého záznamu o pacientovi

1. Systém umožní skrýt/odkrýt celý záznam o pacientovi před ostatními doktory
2. Systém umožní skrýt/odkrýt část záznamu o pacientovi před ostatními doktory

- Lze vyřešit pomocí dynamické části aplikace skrývaním otázek, na které nemá uživatel práva.

**C FP21** Verifikace

1. Systém umožní pověřenému uživateli označit záznam o pacientovi jako zkontrolovaný a správně vyplněný

- Na obrázku 4.6 je možné vidět tuto funkcionalitu v aplikaci Viedoc. Kontrolu otázek je možné naimplementovat pomocí rozšíření modelu „Question“ a „Answer“ o atribut, který by vyjadřoval, zda je otázka již zkontrolována. Tento požadavek nebude implementován, kvůli časovým možnostem.

Obrázek 4.6: Verifikace v aplikaci Viedoc


Zdroj: `https://www.viedoc.com/` k 16.11.2017

#### 4.1.1.4 Historie a analýza dat

**S FP22** Statistiky

1. Systém zobrazí statistiku, kolik je subjektů a zkoušejících ve studii.

- Statistiky mohou pomoci k sledování průběhu a při vyhodnocování studie, proto vytvořím v aplikaci novou sekci, která bude obsahovat tabulku s různými statistikami o průběhu studie.

**S FP23** Historie změn a interakce se systémem

1. Systém zaznamená změny, které byly v systému provedeny
2. Systém umožní zobrazit historii změn
3. Systém zaznamená pohyb uživatele po aplikaci
4. Systém umožní zobrazit pohyb uživatele po aplikaci

5. Systém umožní vyhledávat podle určité akce v systému či podle uživatele

- Data klinických studií jsou velmi citlivá, proto je nutné logovat pohyby v aplikaci, změny nad daty a chyby, které v systému objeví. Posbíraná data je, kromě bezpečnosti, možné využít mnoha způsoby, například na:
  - vytváření statistik, které mohou pomoci při vyhodnocování studie.
  - analyzování průběhu studie či klinických pracovníků.
  - zefektivnění práce.
- Pro splnění tohoto požadavku využiji změny knihovny Reflux pro správu stavu aplikace na Redux a budu ukládat akce v systému, které uživatel vykoná do databáze.

#### 4.1.1.5 Správa dat

**W FP24** Import a export dat

1. Systém umožní import a export celého projektu
2. Systém umožní import a export instituce
3. Systém umožní import a export záznamu o pacientovi

- Tento požadavek není nutné implementovat, jelikož předpokládáme, že statistik je schopen stáhnout/nahrát data z/do databáze.
- Dynamicky generovaný formulář je možné využívat napříč různými studiemi pomocí přepoužití ontologií a skriptů generujích dynamická data.

**C FP25** Správa dokumentů

1. Systém umožní nahrát soubor k záznamu o pacientovi

- Tento požadavek není důležitý pro momentální stav aplikace, protože není příliš častý v klinických studiích.

**C FP26** Vytisknutí záznamu o pacientovi

1. Systém umožní vytisknout záznam o pacientovi v čitelné formě

- Tento požadavek není důležitý pro momentální stav aplikace, jelikož uživatel může vytisknout obsah stránky pomocí klávesové zkratky Ctrl + P.

#### 4.1.1.6 Klinické studie

**M FP27** Randomizace

1. Systém umožní náhodné rozdělení subjektů do ramen (viz kapitola 1.2)

- Z důvodů rozsahu analýzy je možné návrh nalézt v sekci 4.1.4 ke konci této kapitoly.

**M FP28** Rozdělení subjektů do skupin

1. Systém umožní rozdělení subjektů do skupin

- Tento požadavek je možné vyřešit pomocí dynamické části aplikace. Nebude proto implementován v statické části aplikace.

**M FP29** Zaslepení

1. Systém umožní skrýt druh léčby, kterou subjekt dostává

- U jednoduše zaslepené studie stačí subjektu neříct do jaké skupiny patří. Ale nastává problém, pokud subjekt nahlédne zkoušejícímu do záznamu o pacientovi, kde by skupina byla uvedená. Existují dvě možná řešení tohoto problému:

  **1. řešení bez implementace**
  - Zadavatel studie zvolí takové názvy skupin, které neprozrazují nic o druhu léčby. Poté podle typu zaslepení sdělí, nebo nesdělí zkoušejícím a statistikům, co daný název znamená.

  **2. řešení s implementací**
  - Přidat do konfiguračního souboru aplikace typ zaslepení.
  - Na základě typu zaslepení ukázat zkoušejícímu a statistikovi název skupiny po kliknutí na tlačítko „Zobrazit typ léčby“ v detailu o záznamu o pacientovi, aby se předešlo tomu, že subjekt uvidí, do jaké skupiny patří.

#### 4.1.1.7 Ostatní

**M FP30** Plánování návštěv

1. Systém umožní naplánovat akci na určité datum a čas a notifikuje uživatele

- Tento požadavek není důležitý pro momentální stav aplikace.

**C FP31** Lékařské číselníky

1. Systém umožní uživateli využívaní lékařských číselníků
2. Systém umožní uživateli definovat nové choroby spolu s názvem, kódem a stručným popisem

- Tento požadavek je možné vyřešit pomocí dynamické části aplikace. Nebude proto implementován v statické části aplikace.

**C FP32** Kalendář

1. Systém umožní plánování návštěv subjektů
2. Systém umožní plánování schůzek a úkolů
3. Systém zobrazí kalendář naplánovaných akcí celé instituce
4. Systém zobrazí kalendář naplánovaných akcí uživatele

**W FP33** Správa financí

1. Systém umožní nastavit rozpočet instituce
2. Systém umožní nahrávání faktur

**W FP34** Dotazníky

1. Systém umožní vytvářet dotazníky pro subjekty
2. Systém umožní rozesílat dotazníky subjektům
3. Systém zobrazí výsledky dotazníku

**M FP35** Nepodporovaný prohlížeč

1. Systém zobrazí upozornění, že uživatel používá nepodporovaný prohlížeč

- Knihovna React je podporovaná až od určité verze prohlížečů. Jelikož ve zdravotních zařízeních mohou uživatelé používat staré prohlížeče, je nutné informovat, že aplikace StudyManager nemusí správně fungovat.

### 4.1.2 Nefunkční požadavky

• **M NFP1** Hashovaní hesel

1. Systém bude ukládat do databáze osolené a zahashované heslo

- Hashování hesel je důležité, abychom neukládali hesla v databázi jako prostý text. Pokud by někdo data z databáze ukradl, může se lehce přihlásit do systému a zobrazit či změnit klinická data.

• **S NFP2** Responzivní design

1. Systém bude možné používat na tabletech a mobilních zařízeních.

- Více než 50% lidí na celém světě v roce 2016 přistupovalo na webové stránky z mobilních zařízení [35] a uživatelé klinických studií nejsou výjimkou. Je proto nutné, aby StudyManager bylo možné používat i na tabletech a mobilních zařízeních.

• **M NFP3** Použití technologie Redux

- Redux je JavaScriptová knihovna pro uchovávání stavu aplikace, která v poslední době nahrazuje starší Reflux. Existuje spousta vývojářských nástrojů pro Redux, které spolu s možností vracení a upravování stavu, pomáhají k jednodušší správě aplikace. Skládá se ze tří částí: store, akce a reducer. V objektu store je uložen stav aplikace. Akce je objekt, který popisuje změnu stavu, kterou chceme v objektu store udělat. Má atribut typ akce a případná data, která můžeme s možnými změnami v objektu store uložit. Posledním je reducer, ten na základě akce modifikuje data v store [36].

  Hlavním důvodem k použití technologie Redux je využití pro FP23, jelikož vyjadřuje akce pomocí plain Javascript objektů, které jdou jednoduše serializovat. To je důležité pro zaznamenávání interakce uživatele se systémem do sémantické databáze. Očekává se, že bude možné tyto data využít i pro analýzu průběhu studie.

• **M NFP4** Použití ES6 standardu jazyka JavaScript

- ES6 je 6. verze standardu ECMAScript z roku 2015. Oproti používané 5. verzi ES5 je poté kód mnohem přehlednější a jednodušeji se píše. Umožňuje například vytváření proměnných pomocí „const“, díky kterým můžeme jednoduše ošetřit, že proměnná nepůjde změnit.

  S cílem zjednodušit implementování mnoha nových funkcionalit a udržet kód dle NFP6 snadno rozšiřitelný je updatování z ES5 na ES6 žádoucí.

• **M NFP5** Systém bude fungovat na nejnovějších prohlížečích

• **M NFP6** Rozšiřitelnost

### 4.1.3 Požadavky na zavedené řešení

• **M PZ1** Systém bude otestovaný

### 4.1.4 Analýza randomizace

Jedním z cílů bakalářské práce je naimplementovat jednoduchou randomizaci. Pro implementaci jsem si vybral metodu z kategorie adaptivních randomizací pro postupné přidávání subjektů do studie, a to metodu osudí od autorů Wei a Lachin, kterou více rozebírám v sekci 1.2.1.4. Algoritmus na rozřazení subjektů do skupin bude implementován na backendu.

Pro pozdější doimplementování vytvořím rozhraní, které bude rozšiřitelné o metodu minimalizace, která by mohla být dle pánů T. Treasure a K. D. MacRae platinovým standardem randomizačních metod [38]. Jak tato metoda funguje, je možné zjistit v sekci 1.2.1.4.

#### 4.1.4.1 Vstupy a výstupy implementované randomizační metody

Návrh vstupních objektů je možné nalézt v příloze D-1.

Vstupy randomizační metody (Vstupy a atributy označené * jsou nutné pro minimalizaci)

- Stratifikační kritéria*
- Ramena, do kterých je možné subjekty přiřadit spolu s počtem již přiřazených subjektů (* navíc s počtem subjektů splňující daná stratifikační kritéria)
- Subjekt spolu s prognostickými faktory*

Výstupy randomizační metody

- Rameno, do něhož má být subjekt přiřazen

#### 4.1.4.2 Návrh randomizační metody

Jak jsem již avizoval, k implementaci využiji randomizační metodu osudí. Jelikož v naší studii mohou být více než 2 ramena, rozšířím tuto metodu ze sekce 1.2.1.4 na n ramen. Návrh implementace je možné vidět v příloze D-2.

$$P_{r(i)} = \frac{(max_{n_i} + 1) - n_i}{\sum_{i=1}^{n}(max_{n_i} + 1) - n_i}$$

kde:

$n$ je počet ramen

$P_{r(i)}$ je pravděpodobnost přiřazení subjektu do ramene i, kde i ∈ {1,...,n}

$n_i$ je počet již přiřazených subjektů do ramene i, kde i ∈ {1,...,n}

$max_{n_i}$ je nejvyšší počet již přiřazených subjektů z všech ramen

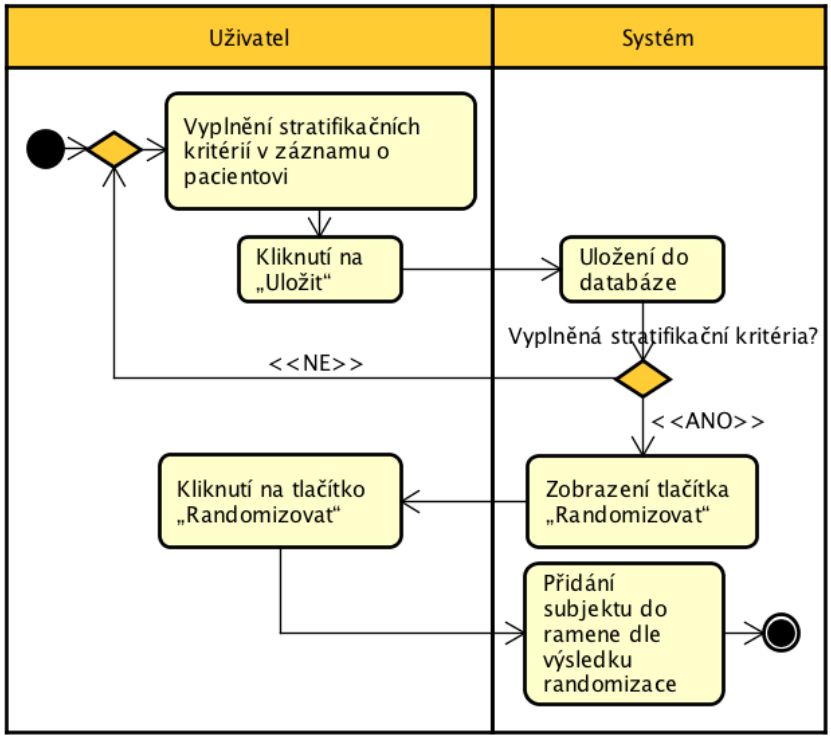
#### 4.1.4.3 Konfigurace stratifikačních kritérii a skupin

Pro metodu osudí nyní není nutné do aplikace přidávat stratifikační kritéria, pouze výčet ramen studie, které momentálně přidám do konfiguračního souboru. Pro budoucí rozšíření metodou minimalizace bude však nutné do dynamicky generovaného formuláře přidat políčka, do kterých se budou vyplňovat prognostické faktory. Dále do repozitáře dynamické části (viz zdrojový kód D.1) bude nutné přidat stratifikační kritéria. Každé stratifikační kritérium má question origin, což je URI otázky v dynamicky generovaném formuláři, do které se vyplňuje prognostický faktor.

#### 4.1.4.4 Diagram aktivit

Na obrázku 4.7 je možné vidět návrh randomizace z pohledu uživatele. Stratifikační kritéria jsou určená pro randomizační metodu minimalizace, pro implementovanou metodu osudí se nyní v průchodu vždy v rozhodovacím uzlu „Stratifikační kritéria?“ zvolí cesta „«NE»“.

Obrázek 4.7: Průběh randomizace z pohledu uživatele




UML 2.0 - Diagram aktivit [14]

## 4.2 Návrh rozšíření aplikace

V této sekci v tabulkách 4.1, 4.2 a 4.3 je možné vidět funkce jednotlivých uživatelských rolí. Ty, které jsou označené zelenou hvězdičkou budou naimplementovány na základě požadavků označených prioritou M a S. V příloze B je v tabulce B.1 srovnání důležitých vlastností existujících řešení s aplikací StudyManager.

Tabulka 4.1: Zobrazení funkcí uživatelské role Doktor po rozšíření StudyManager

| Uživatelská role | Kategorie | Funkce |
|---|---|---|
| Doktor | Instituce | Zobrazit jakoukoli instituci |
| | | Zobrazit všechny uživatele své instituce |
| | | Zobrazit všechny záznamy o pacientech své instituce |
| | Pacienti | Zobrazit všechny záznamy o pacientech své instituce |
| | | Vytvořit / zobrazit / upravit / smazat záznam o pacientovi |
| | | Randomizovat záznam o pacientovi * |
| | Ostatní | Zobrazit hlavní menu |
| | | Zobrazit a upravit svůj profil |
| | | Změnit své heslo * |
| | | Odhlásit se |

Tabulka 4.2: Zobrazení funkcí uživatelské role Administátor po rozšíření StudyManager

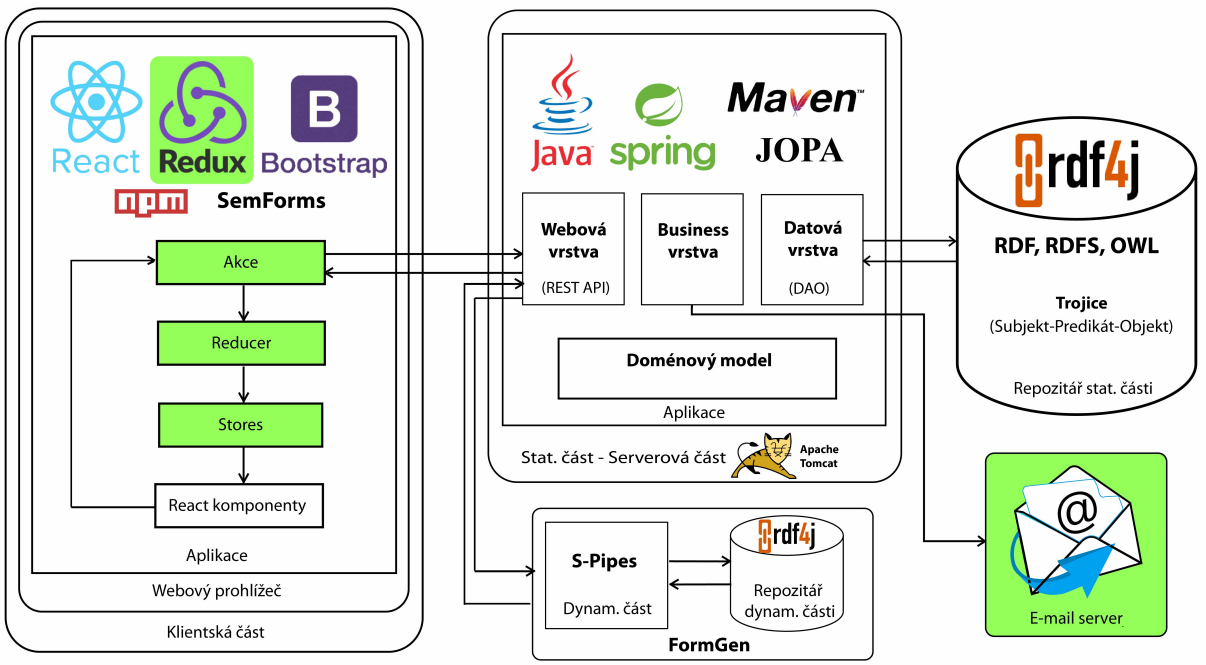
| Uživatelská role | Kategorie | Funkce |
|---|---|---|
| Administrátor | Uživatelé | Zobrazit / upravit / smazat profil uživatele |
| | | Vytvořit nového uživatele jakékoli role |
| | | Pozvat uživatele do studie * |
| | | Změnit heslo jakéhokoli uživatele * |
| | | Impersonovat uživatele * |
| | | Vygenerovat uživatelské jméno * |
| | Instituce | Vytvořit / zobrazit / upravit / smazat instituci |
| | | Zobrazit všechny uživatele jakékoli instituce |
| | | Zobrazit všechny záznamy o pacientech jakékoli instituce |
| | Pacienti | Zobrazit všechny záznamy o pacientech |
| | | Vytvořit / zobrazit / upravit / smazat záznam o pacientovi |
| | | Randomizovat záznam o pacientovi * |
| | Statistiky | Zobrazit statistiky * |
| | Historie | Zobrazit historické akce * |
| | | Filtrovat historické akce * |
| | Ostatní | Zobrazit hlavní menu |
| | | Zobrazit a upravit svůj profil |
| | | Změnit své heslo * |
| | | Odhlásit se |

Tabulka 4.3: Zobrazení funkcí uživatelské role Nepřihlášený uživatel po rozšíření StudyManager

| Uživatelská role | Kategorie | Funkce |
|---|---|---|
| Nepřihlášený uživatel | Ostatní | Resetovat zapomenuté heslo * |
| | | Nastavit nové heslo * |
| | | Přihlásit se |

### 4.2.1 Architektura aplikace

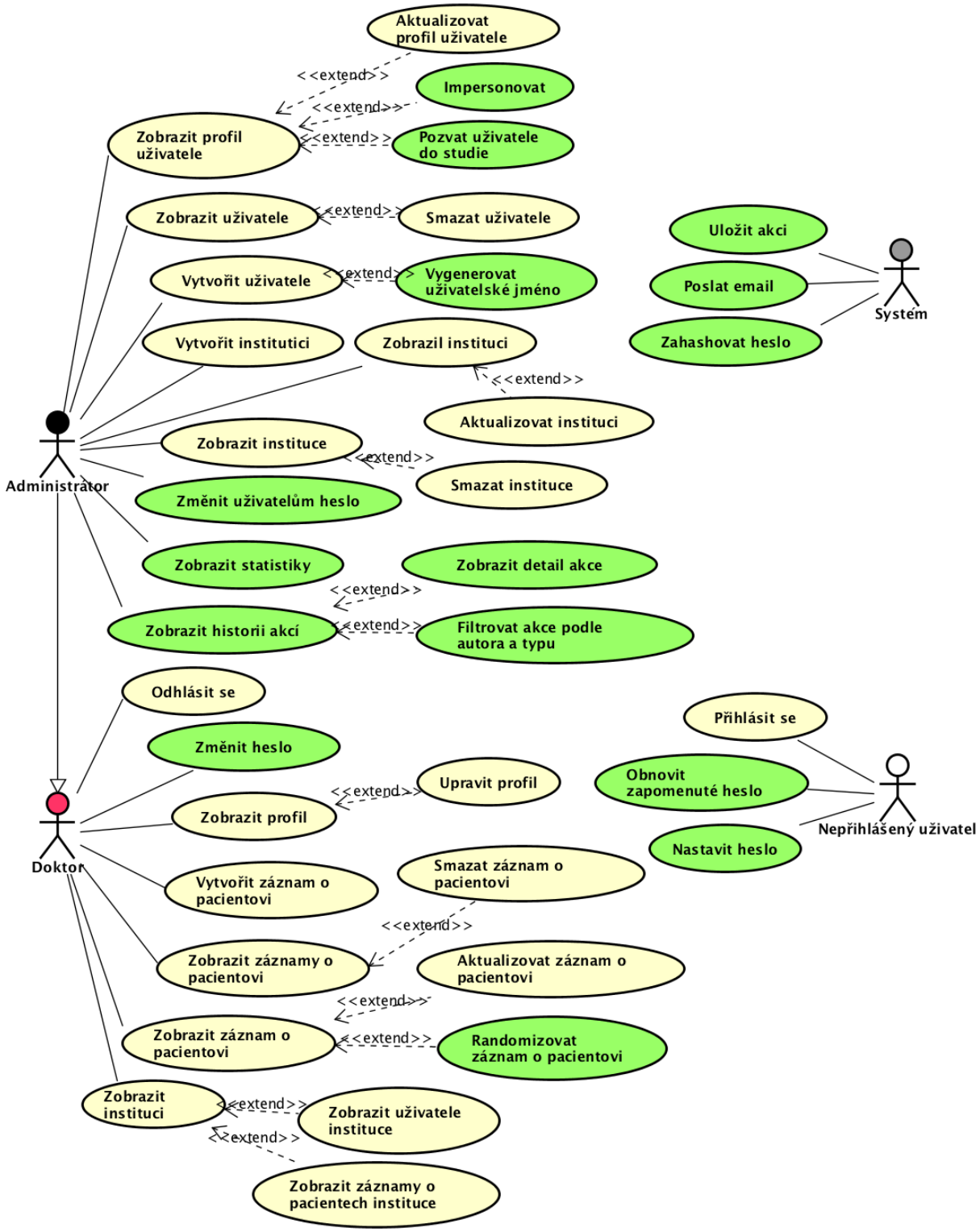
Obrázek 4.8: Architektura aplikace po rozšíření




Části označené zelenou barvou budou v rámci implementace změněny/přidány

### 4.2.2 Případy užití

Obrázek 4.9: Případy užití po rozšíření aplikace




UML 2.0 - Diagram případů užití [14]; funkce označené zelenou barvou jsou naimplementovány

# Kapitola 5

# Implementace

V této kapitole více rozeberu funkční a nefunkční požadavky, které mají prioritu M nebo S a jsou naimplementované v rámci této bakalářské práce. Nejdříve uvedu nefunkční požadavky, jelikož některé zásadně ovlivnily implementaci určitých funkčních požadavků.

## 5.1 Nefunkční požadavky

### 5.1.1 NFP1 - Hashování hesel

K hashování a solení hesel jsem použil implementaci BCryptPasswordEncoder[1] z Spring Security frameworku, která k heslu nejdříve přidá sůl[2] a poté ho zahashuje. Sůl poté přidá nakonec výsledného hashe.

### 5.1.2 NFP2 - Responzivní design

K responzivnímu designu jsem využil CSS media queries a framework Bootstrap, který usnadňuje tvorbu responzivních webů.

### 5.1.3 NFP3 - Použití technologie Redux

V předchozí verzi nástroje se používala knihovna Reflux (viz 3.3.2.2), tu jsem nahradil právě knihovnou Redux, jelikož jsem ji využil k implementaci FP8 a FP23 a celkově k zefektivnění implementace nových funkcionalit. K nahrazení technologie Reflux na Redux jsem chtěl využít postupu od Rufa Raghunatha[3], ale tento způsob se mi zdál zbytečně zdlouhavý.

Postupoval jsem tedy tak, že jsem nejdříve vytvořil akce, poté k nim vhodné reducery, napojil komponenty na Redux store a vytvořil logiku v komponentách pro volání Redux akcí. Na závěr jsem smazal pozůstatky z Reflux technologie. Komponenty, které využívají knihovnu SForms, jsem musel nechat v technologii Reflux, jelikož nepodporuje zatím Redux.

[1] BCryptPasswordEncoder - více informací na: `https://docs.spring.io/spring-security/site/docs/4.2.4.RELEASE/apidocs/org/springframework/security/crypto/bcrypt/BCryptPasswordEncoder.html` k 17.5.2018

[2] Náhodný řetězec, který zaručí, že dvakrát zahashované stejné heslo s jinou solí, nemá stejný hash

[3] Postup k nahrazení technologie Reflux na Redux dostupný na: `https://rufusraghunath.com/2017/05/05/migrating-from-reflux-to-redux.html` k 7.5.2018

### 5.1.4 NFP4 - Použití ES6 standardu jazyka Javascript

Soubory jsem přepisoval do ES6 průběžně, když jsem potřeboval něco v kódu v daném souboru změnit. Vždy když jsem přepsal určitou část aplikace do ES6, musel jsem projít zbytek aplikace a opravit rozbité importy. Kód starající se o knihovnu SForms jsem nechal v standardu ES5, jelikož se bude v budoucnu měnit z důvodu přepisu do technologie Redux.

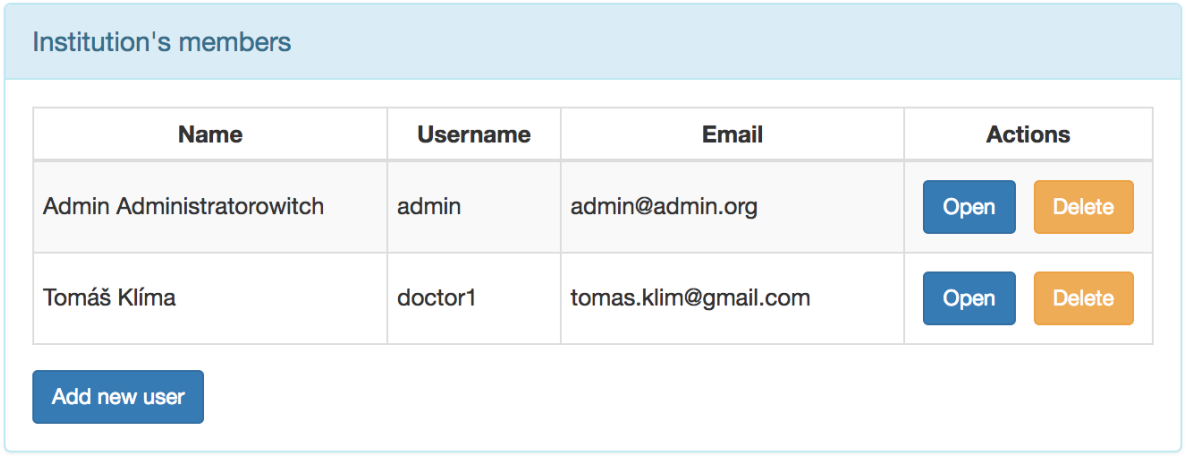
## 5.2 Funkční požadavky

V této sekci rozeberu všechny funkční požadavky označené prioritou M a S a jsou implementovány.

### 5.2.1 FP3 - Vytvoření, zobrazení a smazání uživatele přímo v instituci

V původní verzi v detailu instituce byl seznam uživatelů, kteří patří do dané instituce bez možnosti zobrazení či smazání uživatele. Jelikož tato funkcionalita byla již naimplementována v aplikaci v seznamu všech uživatelů, použil jsem ji v rámci tohoto požadavku. Vytvoření nového uživatele jsem vyřešil přidáním tlačítka „Přidat nového uživatele“ (viz obrázek 5.1), které uživatele přesměruje na formulář pro vytvoření nového uživatele spolu s již vyplněnou institucí, kterou při kliknutí na tlačítko ukládám do Redux store.

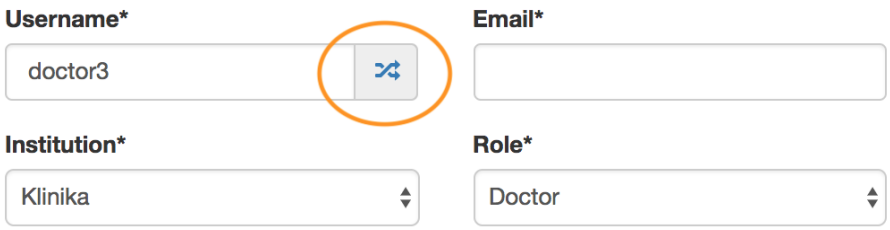
Obrázek 5.1: Část detailu instituce




### 5.2.2 FP4 - Vygenerovaní uživatelského jména

Klientská část aplikace po kliknutí na tlačítko (obrázek 5.2) pošle na server aktuálně zvolenou roli. Server přidá k roli číslo, které je o jedna větší než již existující uživatelské jméno s názvem dané role (viz zdrojový kód 5.1), a pošle ho zpět do klientské části, kde se automaticky vyplní do pole „Uživatelské jméno“.

Obrázek 5.2: Naimplementované tlačítko pro vygenerování uživatelského jména




Zdrojový kód 5.1: Nalezení volného uživatelského jména

```
public String generateUsername(String usernamePrefix) {
   return usernamePrefix + (userDao.findAll().stream()
      .filter(u -> u.getUsername().startsWith(usernamePrefix))
      .map(u -> u.getUsername().replaceFirst(usernamePrefix, ""))
      .filter(s -> StringUtils.isNotBlank(s) && StringUtils.isNumeric(s))
      .map(s -> Integer.parseInt(s))
      .max(Comparator.naturalOrder()).orElse(0) + 1);
}
```

### 5.2.3 FP5 - Pozvání nového uživatele do studie

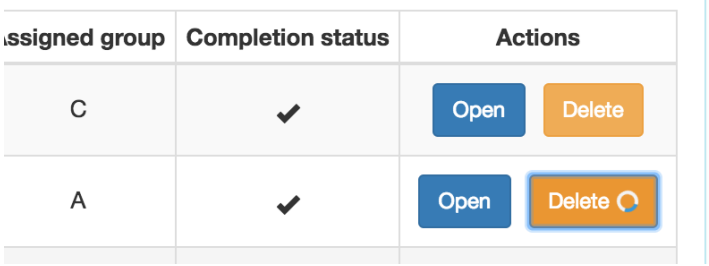
Vytvořil jsem tlačítko „Poslat pozvánku nyní“, které zobrazuji administrátorovi v profilu uživatele, jenž ještě nebyl pozván do studie. Informaci, zda byl již uživatel do studie pozván, jsem zajistil pomocí přidání atributu „isInvited“ do modelu uživatele. Atribut se při vytvoření nového uživatele nastaví na „false“.

Po kliknutí na tlačítko „Poslat pozvánku nyní“, nastavím atribut „isInvited“ na „true“, vygeneruji token pro nastavení nového hesla pomocí funkcionality z FP10 a pošlu přivítací e-mail díky funkcionalitě FP12 spolu s odkazem na nastavení nového hesla.

### 5.2.4 FP8 - Indikátory stavu

Tento požadavek jsem vyřešil díky NFP3. Když klientská část aplikace komunikuje se serverem, ukládám aktuální stav komunikace do Redux store a na základě toho zobrazuji uživateli aktuální stav (viz obrázek 5.3).

Obrázek 5.3: Probíhající odstraňování záznamu o pacientovi

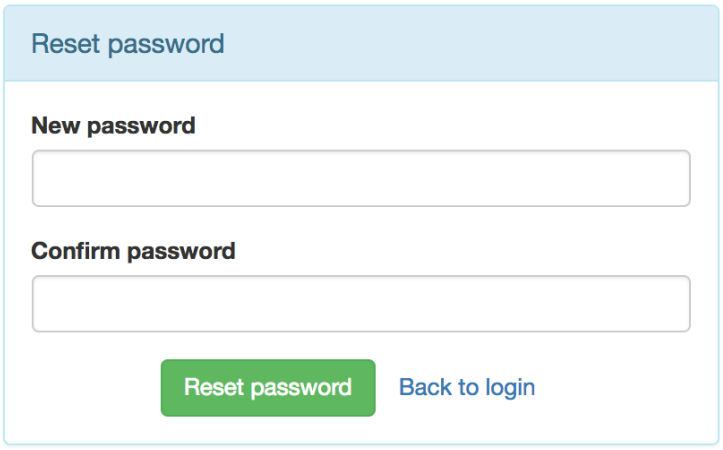


### 5.2.5 FP10 - Zapomenuté heslo

Při implementaci tohoto požadavku jsem vytvořil v klientské části aplikace formulář pro nepřihlášené uživatele, do kterého je možné vyplnit e-mail od účtu, od nějž uživatel zapomněl heslo. Serverová část na základě požadavku vygeneruje pro účet s daným e-mailem token, který uloží do databáze a odešle e-mail na resetování hesla spolu s odkazem obsahujícím právě vygenerovaný token.

V klientské části jsem dále vytvořil formulář k zadání nového hesla, který se zobrazí, pokud uživatel klikne na odkaz s platným tokenem ve své e-mailové schránce (viz obrázek 5.4). Pokud token platný není, formulář se nezobrazí. Spolu s uložením nového hesla do databáze mažu token, aby nemohl být použit dvakrát.

Obrázek 5.4: Formulář pro nastavení hesla pomocí tokenu nepřihlášeným uživatelem




### 5.2.6 FP11 - Změna hesla

V klientské části aplikace jsem vytvořil nový formulář pro změnu hesla s poli: současné heslo, nové heslo a nové heslo znovu. Uživatel v roli doktor může měnit pouze své heslo, administrátor může změnit heslo všem uživatelům v aplikaci a nemusí vyplňovat formulářové pole „Současné heslo“. Po správném vyplnění a odeslání formuláře se v serverové části aplikace nové heslo zahashuje a uloží do databáze. Pokud heslo mění uživatel v roli doktor, zkontroluje se před uložením do databáze, zda vyplněné současné heslo souhlasí s současným heslem uloženým v databázi.

### 5.2.7 FP12 - Notifikace přes e-mail

K implementaci jsem využil knihovnu JavaMail[4], která umožnuje připojení se do většiny e-mailových systémů používající standard MIME[5], SMTP[6], POP3[7] nebo IMAP[8]. Jelikož e-maily posílám, využil jsem standard SMTP pro odchozí poštu. Do konfiguračního souboru jsem přidal proměnné, díky kterým je možné nastavit adresu a port serveru pro posílání e-mailů a autentizační údaje pro přihlášení. Co se týče obsahu e-mailů, je možné v konfiguračním souboru upravit znění předmětu a znění e-mailu.

[4]Dostupné na adrese: `https://docs.oracle.com/javaee/7/api/javax/mail/package-summary.html` k 8.5.2018
[5]MIME - Multipurpose Internet Mail Extensions
[6]SMTP - Simple Mail Transfer Protocol
[7]POP3 - Post Office Protocol
[8]IMAP - Internet Message Access Protocol

Notifikace se nyní využívají při změně v uživatelském profilu a pro FP3, FP10 a FP11. Při případném budoucím rozšíření stačí vytvořit novou e-mailovou šablonu dědící z třídy „BaseEmailTemplate“ a do konfiguračního souboru přidat předmět a obsah tohoto nového e-mailu. Část konfiguračního souboru je možné vidět v kódu 5.2 pod tímto textem.

Zdrojový kód 5.2: Ukázka části konfiguračního souboru pro emaily

```
appContext=http://localhost:8080/
#SMTP host
smtp.host=smtp.gmail.com
#SMTP port
smtp.port=587
#SMTP user
smtp.user=studymanagercvut@gmail.com
#SMTP password
smtp.password=Klinika321
#You can use variables in email contents by using variable, available variables are
    listed before email content property
#Password Reset email subject
email.passwordResetSubject=Password Reset
#PasswordReset email html content, variables: username, link, appContext
email.passwordResetContent=<div><p>Dear user {{username}}, </p><p>please set your new
    password here: {{link}} </p><p>Best regards, <br>StudyManager</p></div>
```

### 5.2.8 FP13 - Impersonace

Při implementaci této funkce jsem vytvořil nové tlačítko „Impersonovat“ v profilu uživatele, které v systému po kliknutí změní právě přihlášeného uživatele na uživatele nového (viz zdrojový kód 5.3). Pokud se chce nyní uživatel vrátit na svůj účet, je nutné odhlásit se a přihlásit na svůj účet pomocí uživatelské jména a hesla. Při větším množství času na implementaci, by bylo vhodné, aby se uživateli v pravém horním rohu zobrazilo, že impersonuje daného uživatele, a měl možnost vrátit se zpět na svůj účet bez odhlášení a znovu přihlášení.

Tato funkce je dostupná pouze administrátorovi a ten se může přihlásit jen na uživatele v roli doktor.

Zdrojový kód 5.3: Změna přihlášeného uživatele

```
public void impersonate(@RequestBody String username) {
   User user = getByUsername(username);
   // test if user to impersonate is not admin
   if (user.getTypes().contains(Vocabulary.s_c_administrator)) {
      throw new BadRequestException("Cannot impersonate admin.");
   }
   // create new object UserDetails with user we want to impersonate
   final SecurityContext context = SecurityContextHolder.getContext();
   UserDetails ud = new UserDetails(user);
   // authenticate user we want to impersonate
   UsernamePasswordAuthenticationToken auth = new
       UsernamePasswordAuthenticationToken(ud, null, ud.getAuthorities());
   context.setAuthentication(auth); // change logged user
}
```

### 5.2.9 FP22 - Statistiky

Implementace statistik zahrnovala vytvoření nové položky v menu, která přesměruje na statistiky. Bylo nutné vytvořit React komponentu, která zobrazuje tabulku se statistikami, dále Redux reducer a akce. Na backendu bylo nutné naimplementovat controller, servisu a dao. Momentálně statistika zobrazuje počet záznamů o pacientech a počet zkoušejících ve studii. K spočítaní počtu zkoušejících jsem využil parametrizovaný SPARQL dotaz, který je možné vidět v kódu 5.4 pod tímto textem.

Zdrojový kód 5.4: Dotaz do databáze na získaní počtu zkoušejících v studii

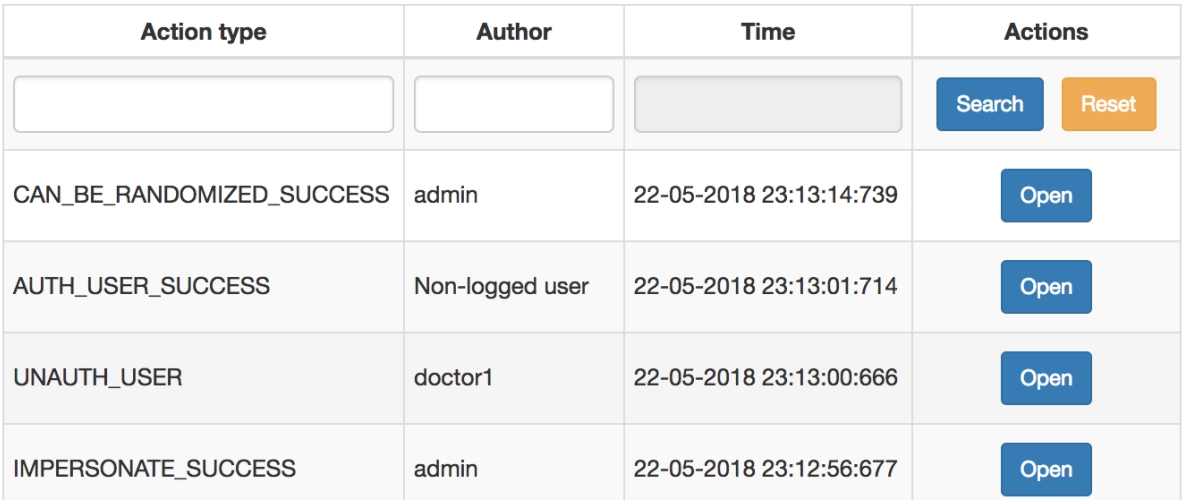
```
public int getNumberOfInvestigators() {
   final EntityManager em = entityManager();
   try {
      return (int) em.createNativeQuery(
      "SELECT (count(?p) as ?investigatorCount) WHERE { ?p a ?typeDoctor . MINUS {?p a
          ?typeAdmin}}")
         .setParameter("typeDoctor", URI.create(Vocabulary.s_c_doctor))
         .setParameter("typeAdmin",
            URI.create(Vocabulary.s_c_administrator)).getSingleResult();
   } finally { em.close(); }
}
```

### 5.2.10 FP23 - Historie změn a interakce se systémem

Chyby, které nastanou na serveru, jsou již logovány do souboru v předchozí verzi aplikace. Rozšířil jsem tedy toto logovaní i do klientské části. Ukládám do databáze chyby a důležité akce, které uživatel vykoná, včetně data, právě přihlášeného uživatele, používaného prohlížeče a obsahu akce či chyby. Jelikož jsem změnil technologii stavu aplikace Reflux na Redux, ukládám do databáze zásadní Redux akce. K ukládání chyb používám stejné rozhraní, jen chyby zachytávám pomocí znázorněného zdrojového kódu 5.5.

Uložené akce a chyby zobrazuji pouze administrátorovi. Vytvořil jsem podobné rozhraní, které je používáno v celé aplikaci (viz obrázek 5.5). Do menu jsem přidal položku „Historie“, která po kliknutí zobrazí list 25 akcí seřazených od nejnovější po nejstarší. Zobrazuji pouze 25 položek na stránku pro rychlejší načítání dat. Mezi stránkami je možné se pohybovat pomocí tlačítek „Další“ a „Předchozí“, kdy při každém pohybu ze stránky na stránku se načítají nová data z databáze. Dále jsem přidal možnost filtrování historie podle autora a názvu akce.

Obrázek 5.5: List akcí




Zdrojový kód 5.5: Zachycení chyby v klientské části

```
window.onerror = (message, source, line) => {
   errorLogger(message, line, store);
   return false; // to show error in console
};
export function errorLogger(message, line, store) {
   const height = window.innerHeight || document.documentElement.clientHeight ||
       document.body.clientHeight;
   const width = window.innerWidth || document.documentElement.clientWidth ||
       document.body.clientWidth;
   const error = {
   type: SCRIPT_ERROR, message, line, url: window.location.href,
      viewport: `${width}x${height}`,
      userAgent: window.navigator.userAgent // current browser and OS
   };
};
```

### 5.2.11 FP27 - Randomizace

Do konfiguračního souboru aplikace jsem přidal dvě proměnné. „enableRandomization“, která pokud je nastavená na boolean hodnotu „true“, povolí v aplikaci randomizaci, a „groups“, do které napíšeme jména skupin, do kterých budou subjekty randomizovány. Poté jsem v serverové části naimplementoval funkci na randomizace podle návrhu zdrojového kódu D.2. Mnou navržená metoda měla nedostatek v zaokrouhlování. Jelikož ve funkci uvažuji pouze celá čísla, nastával problém například, když vyšla pravděpodobnost přiřazení do tří skupin třikrát 1/3, součet těchto pravděpodobností nedal 100%. Dále jsem v serverové části přidal rozhraní pro komunikaci s klientskou částí. Jelikož nyní neuvažuji stratifikační kritéria, funkce, která ověřuje, zda může záznam o pacientovi být randomizován, vrací vždy boolean hodnotu „true“. K modelu záznamu o pacientovi jsem přidal atribut „assignedGroup“, v kterém bude uloženo přiřazené rameno studie z randomizační metody.

V klientské části jsem přidal do listu záznamů o pacientech další sloupec, v kterém je u každého záznamu vidět přiřazené rameno studie. Pokud záznam o pacientovi ještě nebyl randomizován, je tato

položka prázdná. Do detailu záznamu o pacientovi jsem přidal políčko zobrazující přiřazenou skupinu a tlačítko „Randomizovat“, které se zobrazí pouze pokud subjekt ještě nebyl randomizován, jelikož nyní neberu v úvahu vyplněná stratifikační kritéria.

### 5.2.12 FP29 - Zaslepení

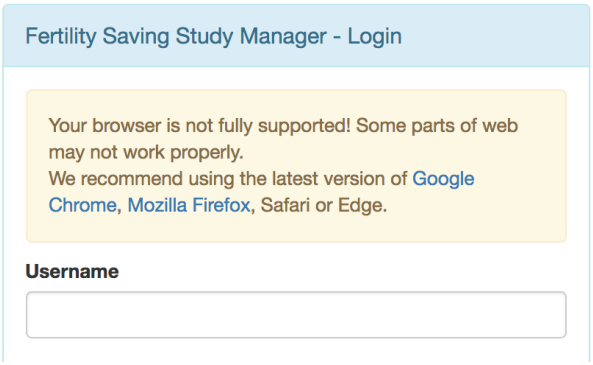
Pro tuto bakalářskou práci jsem zvolil 1. řešení z FP29 bez implementace.

### 5.2.13 FP35 - Nepodporovaný prohlížeč

K vyřešení tohoto požadavku jsem chtěl použít knihovnu react-browser-support [9], jenže se mi ji z nezjištěného důvodu nepovedlo naimportovat. Je ve verzi 0.0.2, tudíž může obsahovat spoustu chyb.

Požadavek jsem tedy splnil pomocí vlastní implementace a jiné knihovny. Kontroluji, zda aktuálně používaný prohlížeč je ve verzi vyšší nebo roven verzi, která je nejnižší podporovaná. Pokud uživatelův prohlížeč je nepodporovaný, zobrazím upozornění při přihlašování (viz obrázek 5.6), že aplikace nemusí fungovat správně a vyzvu ke stažení novější verze prohlížeče. Pro detekci používaného prohlížeče jsem využil knihovnu platform[10]. Aplikace StudyManager funguje v prohlížečích, které podporují standard ES5, proto jsem zvolil ty verze prohlížečů, které tento standard dle webu Can I use[11] podporují.

Obrázek 5.6: Upozornění o nepodporovaném prohlížeči




[9] Dostupné na adrese: `https://www.npmjs.com/package/react-browser-support` k 18.5.2018
[10] Dostupné na adrese: https://www.npmjs.com/package/platform k 12.5.2018
[11] Dostupné na adrese: `https://caniuse.com/#feat=es5` k 12.5.2018

# Kapitola 6

# Testování

Rozšířený systém jsem v souladu se zadáním bakalářské práce a PZ1 řádně otestoval pomocí jednotkových testů[1] a uživatelských testů použitelnosti. Dále jsem z důvodu NFP5 ověřil, že aplikace funguje na nejnovějších prohlížečích (viz tabulka 6.1).

Tabulka 6.1: Seznam otestovaných prohlížečů na různých zařízení

| Operační systém | Prohlížeč | Podporováno |
|---|---|---|
| macOS 10.13.4 | Google Chrome 66.0.3359.139 | Ano |
| macOS 10.13.4 | Safari 11.1 | Ano |
| macOS 10.13.4 | Opera 53.0.2907.37 | Ano |
| macOS 10.13.4 | Firefox 60.0 (64 bitů) | Ano |
| iOS 11.3.1 | Safari 11.1 | Ano |
| iOS 11.3.1 | Google Chrome 66.0.3359.122 | Ano |
| Android 8.1.0 | Google Chrome 66.0.3359.158 | Ano |
| Windows 10 | Edge 42.17134.1.0 | Ano |

## 6.1 Jednotkové testy

Pomocí jednotkových testů jsou pokryté tyto částí aplikace:

**Klientská část**

- Komponenty
- Reducery
- Akce

**Serverová část**

- Webová vrstva
- Business vrstva
- Dao vrstva

[1] Jednotkový test - automaticky ověřuje, zda části zdrojového kódu fungují, jak se předpokládá

- Model aplikace

Naopak nejsou otestované kontrolery[2], jelikož nebyl dostatek času a také protože je otestované vše ostatní v klientské části, tudíž zásadní chyba by neměla uniknout. Dále není testy pokryt uživatelský přístup k webové vrstvě aplikace z nedostatku času a Spring Security[3] vrstva aplikace, protože jsem v této části nic neimplementoval.

Jednotkové testy mi pomohly pro ověření funkčnosti jednotlivých částí aplikace po implementaci nových funkcí a také pro doladění chyb, které při implementaci vznikly. Bylo vytvořeno 112 testů v serverové části a 304 testů v části klientské.

## 6.2 Uživatelské testy použitelnosti

Testování použitelnosti slouží k odhalení zásadních chyb uživatelském prostředí aplikace. Cílovou skupinou testování byli uživatelé, kteří nemají zkušenost s používáním rozšířené aplikace StudyManager, mají znalosti z medicíny a alespoň základní znalost ovládání počítače.

Aplikace byla otestována třemi uživateli na MacBook Pro 2013 v operačním systému macOS verze 10.13.4 v prohlížeči Google Chrome ve verzi 66, pomocí 15 úkolů, které je možné najít v sekci C.1. Nálezy je možné najít v sekci C.2.

## 6.3 Zhodnocení výsledků testování

Během vytváření jednotkových testů jsem objevil několik zásadních chyb aplikace. Největší odhalený problém spočíval ve špatném ověření přístupu k funkci, která slouží k získaní záznamů o pacientovi z databáze. Uživatel, který neměl mít přístup k určitým datům, tento přístup měl. Tuto chybu jsem vyřešil změnou implementace v dané funkci. Řekl bych, že vytvoření jednotkových testů splnilo jejich cíl a budou nadále sloužit při dalším vývoji tohoto nástroje.

Zásadní chyby během testování použitelnosti uživateli nalezeny nebyly, nebudou proto opraveny v rámci této bakalářské práce. Účastnící testování měli k aplikaci pár výtek, které ovšem nezabraňují v možnosti používání aplikace. Kromě kritiky ale ocenili intuitivnost a jednoduchost používání aplikace.

[2] Kontroler - komponenta napojená na Redux store

[3] Spring Security - více informacích dostupných na `https://projects.spring.io/spring-security/` k 17.5.2018

# Závěr

Cílem této bakalářské práce bylo zanalyzovat požadavky prospektivních klinických studií a vytvořit nový, případně rozšířit existující nástroj pro klinické studie StudyManager. Byla zvolena možnost rozšíření aplikace StudyManager, která se z již zmíněných důvodů jeví jako vhodný nástroj pro klinické studie. Dále jsem dle zadání zanalyzoval a teoreticky rozebral požadavky a metody klinických studií, včetně rozboru existujících řešení. Tento teoretický rozbor mi napomohl vyvarovat se chyb a navrhnout adekvátní řešení problému při rozšiřování aplikace. Bylo navrženo 35 nových funkcionalit. Některé z nich, jako například randomizace, posílaní emailů, generování uživatelských jmen, změna hesla a další, byly do aplikace přidány. Ostatní nebyly z různých příčin naimplementovány. U většiny to však bylo z důvodu možnosti přidání pomocí dynamické části aplikace. Kromě dílčího testování při vytváření nových funkcionalit pomocí jednotkových testů, jsem provedl testování použitelnosti mého řešení na 3 nezávislých subjektech, kteří díky faktu, že se zabývají medicínou, poskytli validní názor.

V aplikaci je stále mnoho prostoru pro vylepšení. Z mého návrhu řešení zůstaly nevyužity například implementace vyhledávání, subjektové matice a mnohé další. I požadavky, které byly naplněny, je stále možné rozvinout, například indikátory stavu. Aplikace oproti existujícím řešením vyniká v použití ontologických formulářů a v historii změn, nad kterou je možné provádět různé ontologické filtrování a analýzy.

Domnívám se, že zadání jsem splnil. Rozšířenou aplikaci je možné zprovoznit dle návodu v příloze E. Tato práce mi přinesla mnoho cenných zkušeností v oblasti analýzy softwaru a programování v jazycích Java a JavaScript s knihovnou React. Nesmím opomenout nově nabyté znalosti z oblasti Sémantického Webu, ontologií a klinických studií. Jsem rád, že jsem se mohl podílet na vytváření zajímavého nástroje sloužícího pro dobrou věc.

# Literatura


[1] SÚKL (2010). *Klinické hodnocení léků.* [online] Státní ústav pro kontrolu léčiv. Dostupné z: http://www.sukl.cz/klinicke-hodnoceni-leku [cit. 29.4.2018]

[2] Bizer, C., Heath, T., Berners-Lee, T.(2009). *Linked Data - The Story So Far.* [online] Tom Heath. Dostupné z: http://tomheath.com/papers/bizer-heath-berners-lee-ijswis-linked-data.pdf [cit. 21.5.2018]

[3] Zapletalová, J. (2013) *Typy studií.* [online] MEFANET. Dostupné z: https://mefanet.upol.cz/download.php?fid=115 [cit. 29.4.2018]

[4] SÚKL *O lécích.cz.* [online] Státní ústav pro kontrolu léčiv. Dostupné z: http://www.olecich.cz/ [cit. 3.5.2018]

[5] Farlex *Dictionary, Encyclopedia and Thesaurus - The Free Dictionary* [online] The Free Dictionary. Dostupné z: https://www.thefreedictionary.com/ [cit. 3.5.2018]

[6] Pecen, L. *Praktické aspekty analýzy dat v klinickém výzkumu* [online] KSVI - Univerzita Karlova. Dostupné z: http://ksvi.mff.cuni.cz/~holan/Prednaska2.pps [cit. 27.4.2018].

[7] Nahm, M., Shepherd, J., Buzenberg, A., Rostami, R., Corcoran, A., McCall, J., Pietrobon, R. (2012). *Design and implementation of an institutional case report form library.* [online] PMC. Dostupné z: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3494996/ [cit. 12.11. 2017].

[8] Ústav zdravotnických informací a statistiky ČR (2018). *MKN-10.* [online] Dostupné z: http://www.uzis.cz/cz/mkn/index.html [cit. 7.5.2018]

[9] Röhrig, B., du Prel, J.B., Wachtlin, D.,Blettner, M. (2009). *Types of Study in Medical Research* [online] PMC. Dostupné z: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2689572/ [cit. 29.4.2018]

[10] Pavlíková, M. (2014)*Všespasná randomizace?* [online] Biostatistička. Dostupné z: http://www.biostatisticka.cz/vsespasna-randomizace/ [cit. 26.4.2018].

[11] Lamorte, W.W. (2016). *Prospective Versus Retrospective Cohort Studies.* [online] SPH | Boston University. Dostupné z: http://sphweb.bumc.bu.edu/otlt/MPH-Modules/EP/EP713_CohortStudies/EP713_CohortStudies2.html [cit. 29.4.2018]

[12] Horová, R. (2006). *Co to jsou klinické studie?* [online] Mladá fronta. Dostupné z: https://zdravi.euro.cz/clanek/sestra/co-to-jsou-klinicke-studie-277585 [cit. 29.4.2018]

[13] Mulimani, P.S. (2017). *Evidence-based practice and the evidence pyramid: A 21st century orthodontic odyssey.* [online] PubMed. Dostupné z: https://www.ncbi.nlm.nih.gov/pubmed/28651753 [cit. 29.4.2018]

[14] Tutorials Point *Python Data Science, Java i18n, GitLab, TestRail, VersionOne, DBUtils, Common CLI, Seaborn, Ansible, LOLCODE, Current Affairs 2018, Apache Commons Collections* [online] Dostupné z: https://www.tutorialspoint.com/ [cit: 5.5.2018]

[15] Schulz, K.F., Grimes, D.A. (2002) *Blinding in randomised trials: hiding who got what* [online] The Lancet. Dostupné z: https://pdfs.semanticscholar.org/2198/5326a49a7e6d9d7bc6a938bf032adc4166fb.pdf [cit. 27.5.2018].

[16] Suresh K. (2011) *An overview of randomization techniques: An unbiased assessment of outcome in clinical research.* [online] PMC. Dostupné z: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3136079/ [cit. 26.4.2018].

[17] Stark, N. (2011). *Data Management in Device Studies.* [online] First Clinical Research. Dostupné z: http://www.firstclinical.com/journal/2011/1110_Data_Management.pdf [cit. 12.11.2017].

[18] Latha, M., Bellary, S., Krishnankutty, B. (2017). *Basics of case report form designing in clinical research.* [online] PMC. Dostupné z: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4170533/ [cit. 12.11.2017].

[19] Le Jeannic, A., Quelen, C., Alberti, C., Durand-Zaleski, I. (2014). *Comparison of two data collection processes in clinical studies: electronic and paper case report forms.* [online] PMC. Dostupné z: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3909932/ [cit. 12.11.2017].

[20] OpenClinica. *[Users] clincapture looks familiar.* [online] Dostupné z: https://mailman.openclinica.com/pipermail/users/2013-June/008433.html [cit. 10.12.2017].

[21] Reactjs.org *React - A javascript library for building user interfaces* [online] Dostupné z: https://reactjs.org/ [cit. 30.12.2017].

[22] GitHub. *reflux/refluxjs* [online] Dostupné z: https://github.com/reflux/refluxjs [cit. 30.12.2017].

[23] Bootstrap *Bootstrap is the most popular HTML, CSS and JS library in the world* [online] Dostupné z: https://getbootstrap.com/ [cit. 30.12.2017].

[24] Mička, P. *Další vlastnosti Springu Moduly Springu. Spring Framework.* [online] Dostupné z: http://docplayer.cz/33429371-Dalsi-vlastnosti-springu-moduly-springu-spring-framework-pavel html [cit. 31.12.2017].

[25] Porter, B., Zyl, J., Lamy, O. *Maven – Welcome to Apache Maven.* [online] Maven.apache.org. Dostupné z: https://maven.apache.org [cit. 31.12.2017].

[26] Obitko, M. (2007). *Semantic Web - Introduction to ontologies and semantic web - tutorial.* [online] Obitko.com. Dostupné z: http://obitko.com/tutorials/ontologies-semantic-web/semantic-web.html [cit. 4.1.2018].

[27] Procházka, J. (2009). *Úvod do Sémantického Webu - Zdroják.* [online] Obitko.com. Dostupné z: https://www.zdrojak.cz/clanky/uvod-do-semantickeho-webu/ [cit. 4.1.2018].

[28] Štencek, J. (2009). *Užití sémantických technologií ve značkovacích jazycích.* [online] Vse.stencek.com. Dostupné z: http://vse.stencek.com/semanticky-web/ [cit. 4.1.2018].

[29] RDF Working Group (2014). *RDF - Semantic Web Standards.* [online] W3C. Dostupné z : https://www.w3.org/RDF/ [cit. 17.5.2018]

[30] Schreiber, G., Raimond, Y. (2014). *RDF 1.1 Primer.* [online] W3C. Dostupné z: https://www.w3.org/TR/rdf11-primer/ [cit. 19.5.2018]

[31] JSON-LD. *JSON-LD - JSON for Linking Data.* [online] Dostupné z: https://json-ld.org/ [cit. 6.5.2018].

[32] Petr Křemen, Miroslav Blaško. *s-pipes - KBSS.* [online] Knowledge-based and Software Systems Group. Dostupné z: https://kbss.felk.cvut.cz/web/kbss/s-pipes [cit. 6.5.2018].

[33] Holger Knublauch. *SPARQLMotion.* [online] SPARQLMotion. Dostupné z: http://sparqlmotion.org/ [cit. 6.5.2018].

[34] Agile Business Consortium (2008) *MoSCoW Prioritisation.* [online] Maven.apache.org. Dostupné z: https://www.agilebusiness.org/content/moscow-prioritisation-0 [cit. 31.12.2017].

[35] GO-Globe (2016). *Mobile Vs Desktop Internet Usage – Statistics and Trends [Infographic] | Hongkong Web Design.* [online] Dostupné z: https://www.go-globe.hk/blog/mobile-vs-desktop-internet-usage/ [cit. 7.5.2018]

[36] Redux. *Introduction - Redux.* [online] Dostupné z: https://redux.js.org/introduction [cit. 7.5.2018]

[37] Lichnovská, P., Karberová, E. (2010). *Heuristická analýza :: Testování a hodnocení rozhraní* [online] Dostupné z: https://human-computer-interaction.webnode.cz/testovani-a-hodnoceni-rozhrani/metody-testovani/heuristicka-analyza/ [cit. 8.5.2018]

[38] Treasure, T., MacRae, K.D. (1998) *Minimisation: the platinum standard for trials?* [online] The BMJ. Dostupné z: https://www.bmj.com/content/317/7155/362.full [cit. 27.4.2018].

# Seznam obrázků

# Seznam tabulek

# Zdrojové kódy

# Přílohy

# Příloha A

# Chyby v aplikaci

Chyby v aplikaci, které byly nalezené při procházení aplikace a byly opraveny.

- Doktor si může měnit svou vlastní instituci.

  Pouze administrátor by měl být schopen změnit doktorovi instituci.

- Pokud doktor změní url adresu, může si tím zobrazit jakéhokoli uživatele a instituci v systému.

  Doktor by měl vidět pouze svou instituci a uživatele v ní.

- Doktor si může zobrazit list se všemi institucemi.

  Doktor by měl vidět pouze svou instituci.

- UI aplikace dovoluje změnit svou přezdívku, server ovšem změnu odmítne.

  Nemělo by být možné změnit svou přezdívku.

- Doktor může upravit atribut zaškrtávacího políčka a změnit tím svou roli na administrátora.

  Nemělo by být možné uložit změnu role doktorem.

- Nelze vytvořit dva uživatele se stejným křestním jménem a příjmením.

  Systém by měl umožnit uložit 2 uživatele se shodnými jmény.

- Pokud se z důvodu validace nepovede vytvořit uživatele/instituci na první pokus, není možné bez opětovného načtení aplikace uživatele/instituci vytvořit.

  Systém by měl umožnit vytvořit uživatele/instituci po validační chybě.

- Po vytvoření nového uživatele/instituce není možné uživatele/instituci upravit bez opětovného načtení detailu uživatele/instituce.

  Systém by měl umožnit upravit uživatele/instituci ihned po vytvoření.

- Do systému je možné uložit e-mail uživatele/instituce, který nemá formát e-mailu.

  Systém by měl umožnit ukládat pouze validní e-mail.

- Do systému je možné uložit jméno se speciálními znaky. Detail uživatele s tímto jménem ale nelze otevřít.

  Systém by měl umožnit ukládat uživatelské jméno obsahující pouze znaky a-z, A-Z a 0-9.

- Nepřihlášený uživatel si může zobrazit v aplikaci hlavní menu, vytváření instituce a záznamu o pacientovi.

  Systém by měl nepřihlášenému uživateli zobrazit pouze přihlašovací okno.

- UI aplikace při neúspěšném načtení dat přestane reagovat, dokud uživatel neaktualizuje okno prohlížeče.

  Systém by měl zobrazit chybovou hlášku a umožnit uživateli výkon dalších akcí v aplikaci.

- UI aplikace při načítání dat zamrzne, dokud data nejsou úspěšně načtena.

  Systém by měl umožnit uživateli přejít na jinou část webu při načítání dat.

- Pouze jedna instituce je zobrazena v listu institucí, pokud mají stejný název.

  Instituce se stejným názvem by měly být zobrazeny všechny.

- Po automatickém odhlášení zůstane na přihlašovací stránce navigace přihlášeného uživatele.

  Navigace by měla být zobrazena jen přihlášenému uživateli.

- Při zobrazeném profilu jiného uživatele než svého a následném otevření uživatelova profilu přes jméno v pravém horním rohu se uživatelův profil nenačte.

  Při otevření uživatelova profilu při zobrazeném jiném profilu by se měl zobrazit uživatelům profil.

- Při smazání instituce s uživateli, mají uživatelé stále v profilu tuto instituci.

  Instituce s uživateli by neměla jít smazat

- Lze vytvořit instituci bez jména a e-mailu.

  Instituce by měla vždy mít jméno a e-mail.

- Lze vytvořit doktora bez instituce.

  Každý doktor by měl být přiřazen do nějaké instituce.

# Příloha B

# Porovnání důležitých vlastností aplikace s existujícím řešením

Tabulka B.1: Srovnání důležitých vlastností aplikace StudyManager s ostatními aplikacemi

| | REDCap | OpenClinica | ClinCapture | CastorEDC | OpenMRS | **StudyManager** |
|---|---|---|---|---|---|---|
| Platforma | Web, Mobilní aplikace | Web | Web | Web | Web | **Web** |
| Responzivnost | Ne | Ne | Ne | Ne | Ano | **Ano** |
| Vytvoření eCRF | V grafickém rozhraní | V Excelu | V grafickém rozhraní | V grafickém rozhraní | Naprogra-váním, v exter-ní aplikaci | **Naprogramováním** |
| Uživatelské role | Možné vytvo-ření vlastních | Běžné pro studie | Běžné pro studie | Možné vytvo-ření vlastních | Možné vytvo-ření vlastních | **Administrátor, doktor** |
| Import / export dat | Veškerá data | Veškerá data | eCRF, záznamy | Veškerá data | Pouze import záznamů* | **Ne** |
| Historie změn | Ano | Ano | Ano | Ano | Ano | **Ano** |
| Randomizace | Ano* | Ano** | Ano** | Ano** | Ano* | **Ano** |

* - pomocí přídavného modulu, ** - placená funkcionalita

# Příloha C

# Testování

## C.1 Zadání úkolů testování

**1.** Přihlaste se do aplikace pomocí uživatelského jména „admin“ a hesla „admin“.

**2.** Vytvořte novou instituci.

**3.** Vytvořte nového uživatele v roli doktor s e-mailem „tomas.klim@icloud.com“. Využijte vygenerování uživatelského jména.

**4.** Pozvěte nového uživatele do studie a odhlaste se.

**5.** Přihlaste se na nového uživatele.

**6.** Vytvořte nový záznam o pacientovi.

**7.** Randomizujte vytvořený záznam o pacientovi a zjistěte, do jaké skupiny byl přiřazen.

**8.** Přihlaste se zpět na uživatele „admin“, ale stalo se Vám, že jste zapomněl/a heslo.

**9.** Přihlaste se na vámi vytvořeného uživatele bez znalosti hesla. Pro přihlášení zůstaňte na uživateli „admin“.

**10.** Smažte vámi vytvořený záznam o pacientovi.

**11.** Přihlaste se zpět na uživatele „admin“ a zjistěte, kolik je v aplikaci zkoušejících.

**12.** Zobrazte všechny uživatele a záznamy o pacientech instituce „Ordinace Carolans“.

**13.** Změňte své heslo na „admin“.

**14.** Odstraňte vámi vytvořenou instituci a uživatele.

**15.** Odhlaste se.

## C.2 Nálezy

Závažnost nálezů rozděluji do tří kategorií:

**Závažnost**

- Vysoká
  - Nález, který zásadně ovlivňuje či přímo zabraňuje používaní aplikace. Měl by být ihned opraven.
- Střední
  - Nález, který nemá zásadní vliv na fungování aplikace, ale měl by být co nejdříve opraven.
- Nízká
  - Nález, který nemá téměř vliv na používání aplikace, může být opraven později.

**Nález 1 - Tlačítko vygenerování uživatelského jména**

- Závažnost
  - Nízká (Uživatelské jméno je možné vytvořit i bez vygenerování)
- Nalezené účastníkem
  - 1, 3
- Nalezené u úkolu
  - 3
- Popis
  - Účastníka nenapadlo, že tlačítko u políčka uživatelské jméno vygeneruje uživatelské jméno

**Nález 2 - Heslo při vytváření uživatele**

- Závažnost
  - Nízká (Heslo je uvedeno z důvodu přihlášení na uživatele bez impersonace předtím, než je pozván do studie. Například někým, kdo není administrátor)
- Nalezené účastníkem
  - 1
- Nalezené u úkolu
  - 5
- Popis
  - Účastník byl zmatený, proč si má volit nové heslo, když bylo zobrazené při vytváření nového uživatele a zná ho.

**Nález 3 - Zpět na přihlášení**

- Závažnost
  - Nízká (Na přihlašovací stránku je možné se dostat i bez kliknutí na tlačítko „Zpět na přihlášení“)

- Nalezené účastníkem
  - 1, 2
- Nalezené u úkolu
  - 5
- Popis
  - Účastník si nevšiml po nastavení hesla tlačítka „Zpět na přihlášení“. K zobrazení přihlašovacího okna využil změnu url adresy v adresním řádku.

**Nález 4 - Indikátor stavu impersonace**

- Závažnost
  - Střední (Uživateli nemusí být jasné, zda už impersonace proběhla)
- Nalezené účastníkem
  - 1, 3
- Nalezené u úkolu
  - 9
- Popis
  - Účastník nevěděl, zda impersonace proběhla úspěšně a zda je přihlášen na jiném uživateli, jelikož očekával, že se zobrazí informace o úspěšné akci.

**Nález 5 - Tlačítko uložit a poslat e-mail**

- Závažnost
  - Střední (Uživatel může být zmaten, proč tam jsou dvě tlačítka, když mu stejně přijde email při zvolení jakéhokoli z nich)
- Nalezené účastníkem
  - 3
- Nalezené u úkolu
  - 13
- Popis
  - Účastník si všiml, že po změně svého hesla, mu přišel e-mail, i když nevybral tlačítko uložit a poslat e-mail.

**Nález 6 - Instituce s uživateli nemůže být smazána**

- Závažnost
  - Nízká (Systém jasně upozorňuje, kde nastala chyba)
- Nalezené účastníkem
  - 2
- Nalezené u úkolu
  - 14
- Popis
  - Účastník nevěděl, jak odstranit instituci, jelikož mu systém napsal „Institution with members or patient records cannot be deleted“. Čekal, že mu systém dá nápovědu, aby nejdříve odstranil uživatele přidané do instituce.

# Příloha D

# Zdrojové kódy

## D.1 Návrh vstupních objektů randomizační metody ve formátu JSON

Zdrojový kód D.1: Návrh vstupních objektů randomizační metody ve formátu JSON

```
// * Stratifikační kritéria - pole s objekty Criteria
[
  {
    qo: "<....age>", //URI Question origin
    type: "RANGE",
    options: ["<=20", "<=50", ">50"]
  },
  {
    qo: "<....age>",
    type: "OPTIONS",
    options: ["Female", "Male"]
  }
]

// * Subjekt - objekt Subject
{
  key: "12345678"
  factors: {age: 20, gender: "Female"}
}

// Ramena - pole s objekty Arm
[
  {
    name: "A",
    total: 24,
    numOfParticipantsWithFactor*: [
      [5, 10, 9], // odpovídá počtu subjektů s daným faktorem z objektu Criteria podle
          indexů pole
      [10, 14], // např. 10 subjektů v rameni A jsou ženy, 14 jsou muži
    ]
  },
  {...}
]
```

## D.2 Návrh randomizační metody v jazyce Java

Zdrojový kód D.2: Návrh randomizační metody v jazyce Java

```
private static int getRandomNumberBetween(int low, int high) {
   SecureRandom r = new SecureRandom();
   return r.nextInt(high+1-low) + low;
}

public static String randomize(List<Arm> arms, Subject subject, List<Criterion>
    criteria) {
   // objekt subjekt a criteria se v osudí nepoužívá, může být nu
   if (arms == null) {
      throw new BadRequestException("Object arms should not be null!");
   }

   int max = getMaximumNumberOfSubjectsInArms(arms);
   int numberOfArms = arms.size();
   int[] probabilitiesOfAssignment = new int[numberOfArms];

   int denominator = getDenominatorForCounting(arms, max);
   int probabilitiesTotal = 0;
   // spočítám pravděpodobnosti přiřazení subjektu do každého ramene
   // sečtu všechny pravděpodobnosti, jelikož kvůli zaokrouhlování může vyjít součet 99
   for(int i = 0; i < arms.size(); i++) {
      int probability = (((max + 1) - arms.get(i).getTotal())*100) / denominator;
      probabilitiesOfAssignment[i] = probability;
      probabilitiesTotal += probability;
   }

   int randomNumber = getRandomNumberBetween(1, probabilitiesTotal);
   int index = 0;
   int sumOfProbabilities = 0;
   String armToAssign = null;

   // mám pomyslnou osu od 1 do 100 s body
   // vytvářím si na ní body v vzdálenosti pravděpodobnosti přiřazení do ramene krát
       100
   // pokud náhodně vygenerované číslo, patří do daného úseku mezi body
   // vrátím úsek do kterého subjekt patří, což je rameno
   while (armToAssign == null) {
      sumOfProbabilities += probabilitiesOfAssignment[index];
      if (randomNumber <= sumOfProbabilities) {
         armToAssign = arms.get(index).getName();
         return armToAssign;
      }
      index++;
   }
   throw new BadRequestException("Something went wrong during randomization!");
}
```

## D.3 Ukázka dynamicky generovaného formuláře

Zdrojový kód D.3: Ukázka dynamicky generovaného formuláře

```
{
  "@context": { //
    "doc": "http://onto.fel.cvut.cz/ontologies/documentation/",
    "x": "https://x.com/x/",
    "rdfs": "http://www.w3.org/2000/01/rdf-schema#",
    "has-layout-class": { // typ políčka
      "@id": "http://onto.fel.cvut.cz/ontologies/form-layout/has-layout-class"
    },
    "has_related_question": { // má podotázky
      "@id":
         "http://onto.fel.cvut.cz/ontologies/documentation/has_related_question",
      "@type": "@id"
    },...
    // a mnohé další např: has-question-origin (identifikátor), is-relevant-if
        (validační pravidlo), requires-answer (validační pravidlo),
        has-tested-question,...
  },
},
  "@graph": [
    { // validační pravidlo - když checkbox má hodnotu true, tak...
      "@id": "_:information-available",
      "@type": "http://onto.fel.cvut.cz/ontologies/form/condition",
      "http://onto.fel.cvut.cz/ontologies/form/accepts-answer-value": true,
      "has-tested-question": "rdfs:checkbox-show-q"
    },
    { // checkbox, který obsahuje podotázku
      "@id": "rdfs:checkbox-show-q", "@type": "doc:question",
      "has-layout-class": ["section", "checkbox", "answerable"], // typ
      "has_related_question": ["rdfs:collapsed-q" ], // podotázky
      "has-question-origin": "@id",
      "rdfs:label": "Information of the outcome available?*" // text k checkboxu
    },
    {
      "@id": "rdfs:collapsed-q", "@type": "doc:question",
      "has-question-origin": "@id",
      "requires-answer": true, // povinné políčko
      "rdfs:label": "First name*", // text k políčku
      "is-relevant-if": "_:information-available" // zobrazí se, když podmínka platí
    },
    {
      "@id": "x:form-1-default-q", "@type": "doc:question",
      "has_related_question": "rdfs:checkbox-show-q", // má podotázku checkbox
      "has-layout-class": ["section","wizard-step"], // typ (sekce, wizard)
      "rdfs:label": "Example form"
    },
    {
      "@id": "x:form-1-q", "@type": "doc:question",
      "has_related_question": "x:form-1-default-q", // má sekce
      "has-layout-class": "form" // formulář - kořen stromu formuláře
}]}
```

# Příloha E

# Návod k instalaci StudyManager

### Potřebné soubory

- **StudyManager** - viz přiložený kompaktní disk dle přílohy F
- **SPipes Mock** - viz přiložený kompaktní disk
- **RDF4J 2.2.*** - ke stažení na adrese `http://rdf4j.org/download/`
- **Apache Tomcat 8.5.*** - ke stažení na adrese `https://tomcat.apache.org/download-80.cgi`
- **JDK 8** - ke stažení na adrese `http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html`
- **Maven 3.5.*** - ke stažení na adrese `https://maven.apache.org/download.cgi#`
- **Node.js 8.9.4** - ke stažení na adrese `https://nodejs.org/dist/v8.9.4/`

### Instalace

1. Přesuňte a rozbalte study_manager.zip a spipes_mock.zip z kompaktního disku na svůj disk
2. Stáhněte potřebné soubory
3. Nainstalujte JDK, Maven a Node.js
4. Rozbalte Apache Tomcat a vytvořte kopii, kterou použijete pro databázi (budete mít 2x Apache Tomcat)
5. Přepněte se do složky „conf“ v druhém Apache Tomcat, otevřete soubor „server.xml“ a změňte na řádku s „<Connector port="8080"protocol="HTTP/1.1"connectionTimeout="20000" redirectPort="8443"/> „port“ z 8080 na 18080 a „redirectPort“ na 18443. Uložte a zavřete.
6. Rozbalte RDF4J, vyjměte z „war“ složky obsah a vložte ho do složky „webapps“ v druhém Apache Tomcat.
7. Otevřete příkazový řádek, přepněte se do složky „SPipes Mock“ a napište „mvn clean install“
8. Vyčkejte na dokončení, přepněte se do složky „target“, odkud zkopírujte soubor „spipes-mock.war“ do „webapps“ v druhém Apache Tomcat
9. Otevřete příkazový řádek, nasměrujte se do podsložky „bin“ v druhém Apache Tomcat (pomocí příkazu „cd“). Přidejte práva souboru „catalina.sh („chmod 755 catalina.sh“) a spusťte „catalina.sh“ („./catalina.sh start“).

10. Nyní opět server vypněte(„./catalina.sh stop“), přemístěte obsah „repositories.zip“ do „%appdata/RDF4J“ a znovu spusťte server („./catalina.sh start“)
11. Otevřete příkazový řádek, nasměrujte se do podsložky „bin“ v druhém Apache Tomcat. Přidejte práva souboru „catalina.sh“ („chmod 755 catalina.sh“) a spusťte „catalina.sh“ („./catalina.sh start“).
12. Otevřete příkazový řádek, nasměrujte se do složky „StudyManager“ a napište „mvn clean install -Pproduction“.
13. Vyčkejte na dokončení a běžte do složky „target“ odkud zkopírujte soubor „study-manager.war“ do „webapps“ v prvním Apache Tomcat.
14. Otevřete příkazový řádek, nasměrujte se do podsložky „bin“ v prvním Apache Tomcat. Přidejte práva souboru „catalina.sh“ („chmod 755 catalina.sh“) a spusťte „catalina.sh“ („./catalina.sh start“).
15. Nyní byste měl nalézt funkční aplikaci na adrese `http://localhost:8080/study-manager`.
16. Můžete se přihlásit pomocí přihlašovacího jména „admin“ a hesla „5y5t3mAdm1n.“.

# Příloha F

# Obsah přiloženého kompaktního disku

- **study_manager.zip** - rozšířená statická část aplikace
    - **src** - zdrojové kódy aplikace
- **spipes_mock.zip** - dynamická část aplikace
    - **src** - zdrojové kódy aplikace
- **repositories.zip** - data pro RDF4J Server
- **klimato2_2018bach.pdf** - zpráva bakalářské práce
- **bachelor_thesis_source.zip** - zdrojové kódy zprávy bakalářské práce
- **README.txt** - návod na zprovoznění aplikace