ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

FAKULTA BIOMEDICÍNSKÉHO INŽENÝRSTVÍ

Katedra biomedicínské informatiky

Systém pro ukládání a přístup k datům z geriatrických vyšetření

System for geriatric examinations data storing and accessing

Bakalářská práce

Studijní program:	Biomedicínská a klinická technika
Studijní obor:	Biomedicínská informatika

Autor bakalářské práce:	Karolína Straková
Vedoucí bakalářské práce:	Mgr. Radim Krupička, Ph.D.

Kladno 2019

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení:	**Straková**	Jméno:	**Karolína**	Osobní číslo:	**456602**
Fakulta:	**Fakulta biomedicínského inženýrství**				
Garantující katedra:	**Katedra biomedicínské informatiky**				
Studijní program:	**Biomedicínská a klinická technika**				
Studijní obor:	**Biomedicínská informatika**				

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Systém pro ukládání a přístup k datům z geriatrických vyšetření

Název bakalářské práce anglicky:

System for geriatric examinations data storing and accessing

Pokyny pro vypracování:

Cílem bakalářské práce je navrhnout a implementovat způsob uložení a přístupu k datům z funkčních geriatrických vyšetření. V práci se seznamte se způsobem sběru dat, který provádí Centrum pro studium dlouhověkosti a dlouhodobé péče na Fakultě humanitních studií UK. Na základě požadavků navrhněte systém pro jejich správu a ukládání, ten nainstalujte, upravte pro potřebu importu dat a otestujte. Vytvořte aplikaci pro import již naměřených dat ve formátu XML, pomocí aplikace data importujte.

Seznam doporučené literatury:

[1] Tomáš Janků, GDiag Webová aplikace pro správu a export funkčních vyšetření , 2013
[2] Paul A. Harris, Robert Taylor, Robert Thielke, Jonathon Payne, Nathaniel Gonzalez, Jose G. Conde,, Research electronic data capture (REDCap)—A metadata-driven methodology and workflow process for providing translational research informatics support, Journal of Biomedical Informatics, ročník 42, číslo 2, 2009

Jméno a příjmení vedoucí(ho) bakalářské práce:

Mgr. Radim Krupička, Ph.D.

Jméno a příjmení konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **19.02.2019**
Platnost zadání bakalářské práce: **20.09.2020**

doc. Ing. Zoltán Szabó Ph.D.	prof. MUDr. Ivan Dylevský, DrSc.
podpis vedoucí(ho) katedry	podpis děkana(ky)

PROHLÁŠENÍ

Prohlašuji, že jsem bakalářskou práci s názvem „Systém pro ukládání a přístup k datům z geriatrických vyšetření“ vypracovala samostatně a použila k tomu úplný výčet citací použitých pramenů, které uvádím v seznamu přiloženém k diplomové práci.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu § 60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů.

V Kladně 15.5.2019 …..……..……...

Karolína Straková

PODĚKOVÁNÍ

Mé poděkování patří Mgr. Radimu Krupičkovi, Ph.D. za odborné vedení, trpělivost a ochotu, kterou mi v průběhu zpracování bakalářské práce věnoval.

ABSTRAKT

Systém pro ukládání a přístup k datům z geriatrických vyšetření

Se zvyšujícím se počtem starších lidí v populaci je potřeba zefektivnit a zkvalitnit péči a hodnocení celkového stavu osob ve vyšším věku. Posuzování zdravotního stavu starších osob je momentálně prováděno pomocí funkčních vyšetření. Jedná se o řadu specializovaných testů, jejichž výsledky jsou zaznamenávány do papírových formulářů. Záměrem této práce je modernizovat ukládání dat z funkčních vyšetření a zlepšit dostupnost těchto dat pro zdravotnický personál. Hlavním cílem této práce je výběr databázového systému pro ukládání dat z geriatrických vyšetření a tvorba aplikace, která umožní přenos již nasbíraných dat do vybrané databáze. Data z papírových dotazníků uložená v XML (Extensible Markup Language) formátu jsou pomocí aplikace získána, upravena do tvaru požadovaného pro import a následně přenesena do systému REDCap (Research Electronic Data Capture). Přes webovou aplikaci systému REDCap je možné přistupovat k nasbíraným datům, které jsou zde přehledně uložené v předpřipravených webových dotaznících. K vývoji aplikace i ukládání dat je využito nejnovějších technologií a samotná implementace aplikace je provedena v programovacím jazyce C#.

Klíčová slova

EDC systémy, funkční geriatrická vyšetření, XML, REDCap

ABSTRACT

System for geriatric examinations data storing and accessing

As the numbers of older people within the population increase, it is necessary to make the nursing care more effective and provide it in higher quality, which also applies to the assessment of the overall condition of people at their higher age. Currently, the assessment of health conditions of older people is executed in the form of functional examinations. These consist of a series of specialized tests, whose results are recoded to paper-based forms. The goal of this thesis is to modernize the storage of data obtained through the functional examinations and to improve the accessibility of such data for medical staff. The main aim of this thesis is to choose a database system for the storage of data arising from geriatric examinations and to create an application which would enable the collected data to be transferred to the selected database. The data included in the paper-based questionnaires stored in the XML (Extensible Markup Language) format are then acquired through an application, modified in a form required for the import and then transferred to the REDCap (Research Electronic Data Capture) system. By using the website application of the REDCap system, the collected data may be accessed, while the data are well arranged and stored in already prepared website questionnaires. The development of the application and the storage of data is executed by using the latest technology. Moreover, the C# programming language was used to implement the application.

Keywords

EDC systems, functional examinations, REDCap, XML

Obsah

Seznam zkratek

Zkratka	Význam
REDCap	Research Electronic Data Capture
XML	eXtensible Markup Language (rozšiřitelný značkovací jazyk)
FGA	Funkční geriatrická vyšetření
ECD	Electronic Data Capture (elektronický sběr dat)
API	Application Programming Interface (rozhraní pro programování aplikací)

Seznam obrázků

1 Úvod

V dnešní době dochází ke stárnutí populace. Z tohoto důvodu by se měla zaměřit pozornost více na péči a vyšetření starých lidí. Jedním ze způsobů zkvalitnění péče o seniory jsou funkční geriatrická vyšetření (FGV), která poskytují širší náhled na celkovou kvalitu a úroveň života seniorů. Zároveň tato vyšetření mohou sloužit k zefektivnění péče, díky dotazníkům je totiž možné sledovat průběžně stav pacienta a současně vyhodnocovat jeho potřeby.

Právě dotazníky k funkčním geriatrickým vyšetřením se zabýval i projekt Fakulty humanitních studií UK a FBMI ČVUT a také závěrečné práce několika absolventů. V rámci těchto projektů byly vytvořeny nástroje na tvorbu dotazníků pro funkční geriatrická vyšetření, jejich digitalizaci a následné uložení sesbíraných dat do předem určené XML (eXtensible Markup Language) šablony. Do dnešní doby je sesbíráno kolem 8000 vyplněných dotazníků, a to ze dvou poboček Gerontocentra.

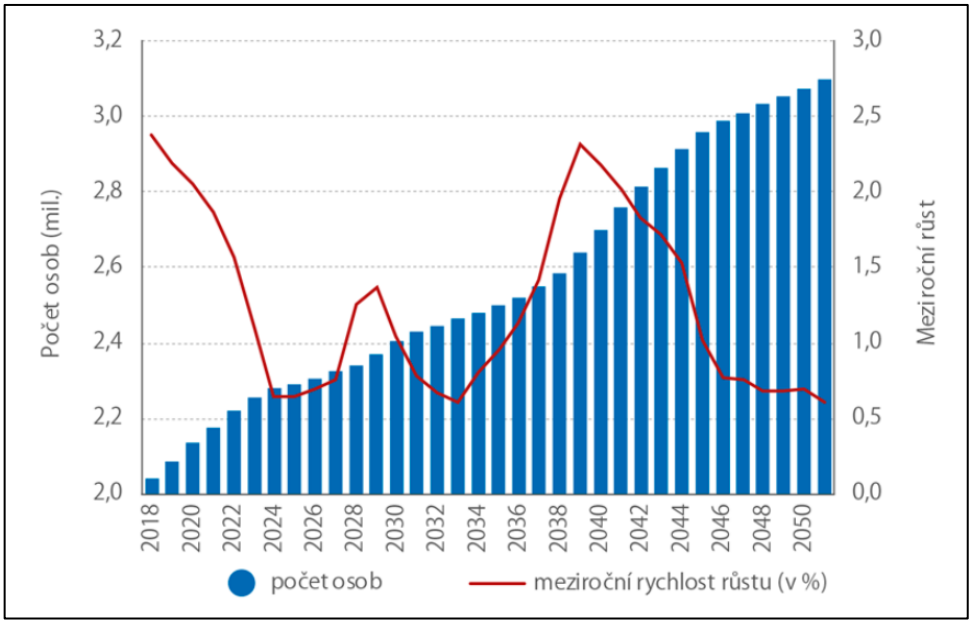
Cílem této bakalářské práce je navrhnout a implementovat způsob uložení dat z funkčních geriatrických vyšetření. Dále je potřeba na základě požadavků navrhnout systém pro jejich ukládání, ten nainstalovat a otestovat, a následně navrhnout a implementovat způsob migrace již uložených dat do databáze.

Pro volbu tématu mne motivovala především skutečnost, že může dojít ke zjednodušení hned v několika oblastech. Především při sběru dotazníků, poněvadž bude umožněn i sběr dat do online dotazníků, což mnohonásobně urychlí následnou práci s dotazníky, a též odboural veškerou zdlouhavou manipulaci s jejich papírovou podobou. Uložení dat v online dostupné databází zároveň ulehčí přístup všem, kteří kvůli péči o pacienta potřebují s danými daty manipulovat.

2 Přehled současného stavu

V této kapitole je popsána oblast, kterou se bakalářská práce zabývá. Též zde bude vysvětlen význam této bakalářské práce, vysvětlena aktuální situace v této oblasti a analyzován současný stav řešení podobné problematiky.

Podle nedávno publikovaného článku v měsíčníku českého statistického úřadu měla Česká republika 2,040 milionů obyvatel starších 65 let, což bylo 19,2 % z celého obyvatelstva ČR. Podle předpokladů ČSÚ z roku 2018 se bude počet obyvatel ve věku nad 65 let každoročně zvyšovat. Např. v roce 2058 by měl počet starých lidí představovat až 30 % celé populace ČR. Předpokládaný vývoj je zobrazen na Graf 2.1. Tato skladba je ovlivněna jednak nižší úmrtností (díky prodlužování délky života) a nízkou porodností. Z důvodu uvedeného demografického vývoje se stává péče o seniory důležitým ekonomickým i zdravotním problémem. Jednou z možností, jak zkvalitnit a zefektivnit péči o seniory je analýza dat z funkčních geriatrických vyšetření. [1]



Graf 2.1: Očekávaný vývoj počtu obyvatel ve věku nad 65 let, zdroj [1]

2.1 Funkční geriatrická vyšetření

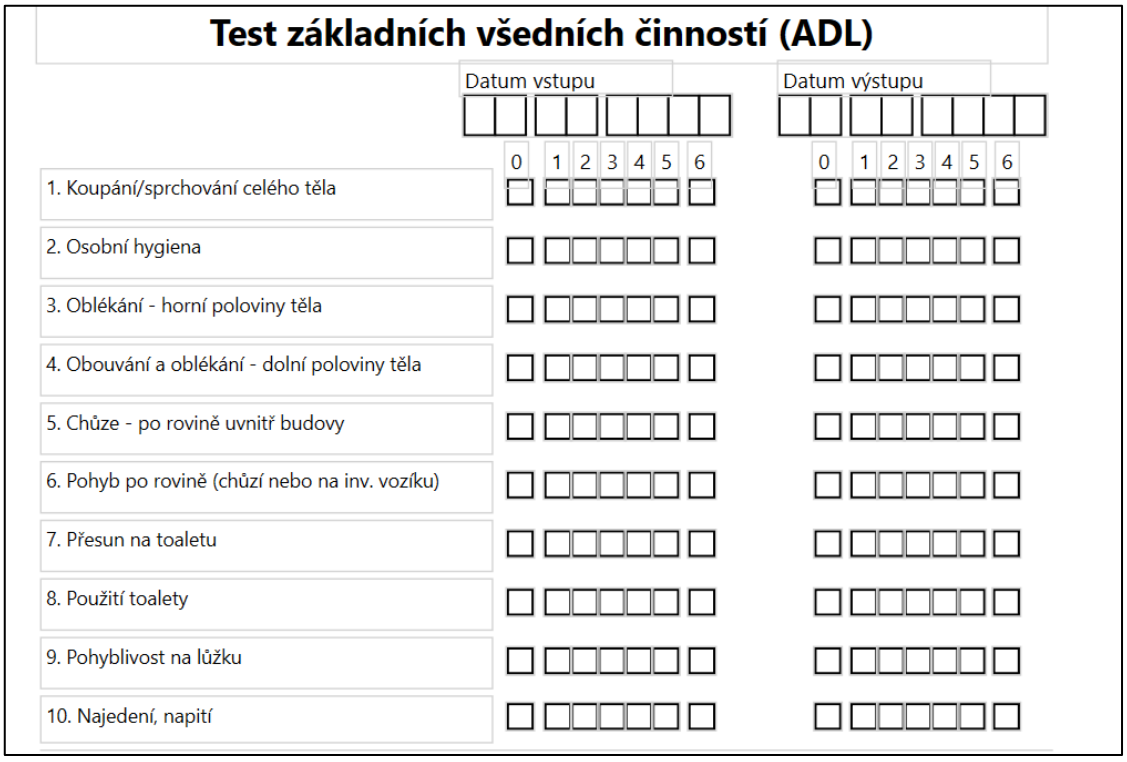
Funkční geriatrické vyšetření (zkráceně FGA) je multidisciplinární diagnostický a léčebný proces, který hodnotí zdravotní, psychický, sociální i funkční stav starší osoby. Cílem FGA je vytvořit plán s cílem zkvalitnit celkový zdravotní stav. Zdravotní péče o staršího člověka vyžaduje hodnocení více oblastí včetně fyzických, kognitivních, sociálních a psychických složek, které ovlivňují zdraví a kvalitu života starého člověka. FGA jsou založeny na předpokladu, že systematické hodnocení starších osob týmem

zdravotníků může lépe identifikovat zdravotní problémy a vést ke zlepšení kvality života a zdravotního stavu. [2]

- **Příklady funkčních geriatrických vyšetření**

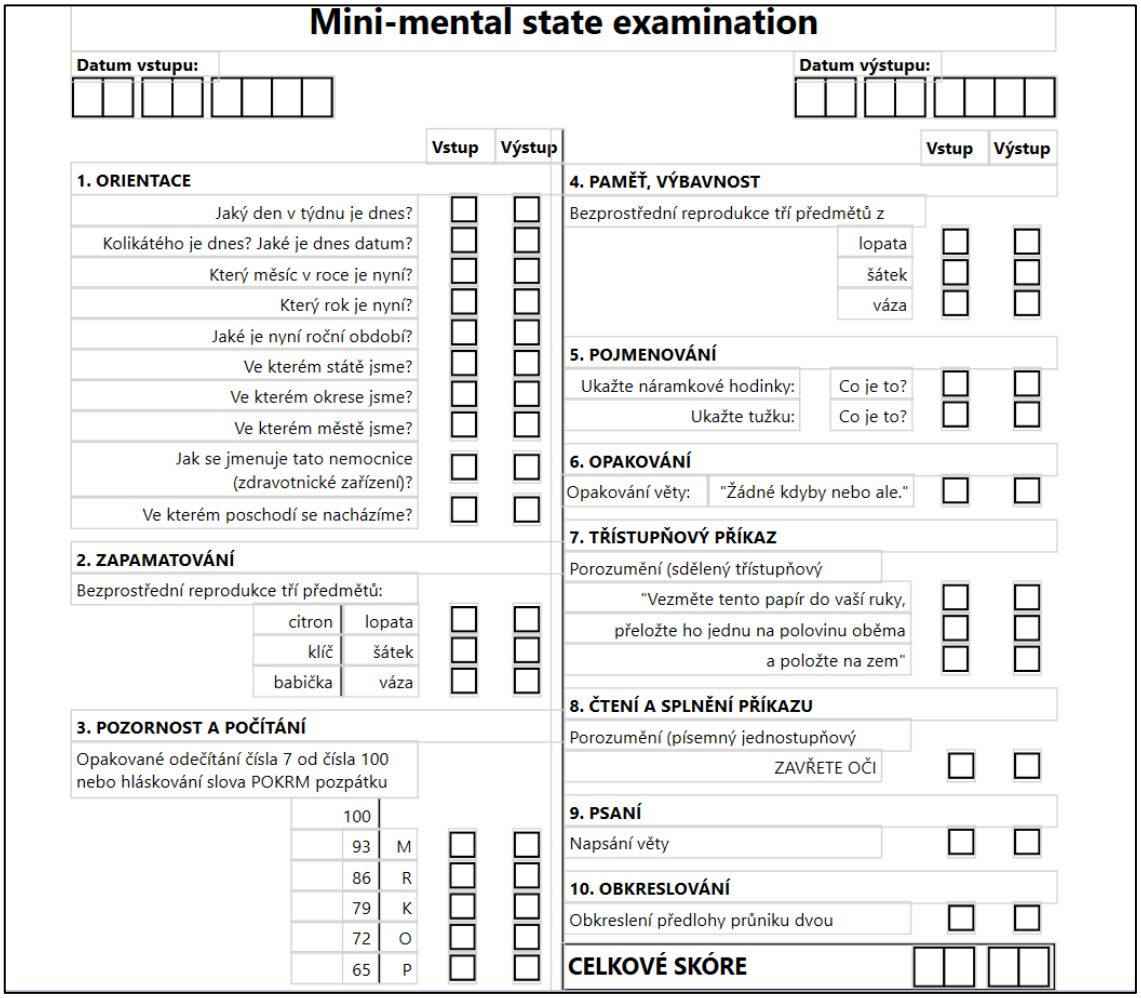
Existuje rozsáhlé množství funkčních geriatrických vyšetření (zkráceně FGA) pro všechny klíčové oblasti (zdravotní stav, duševní zdraví, sociální situaci, fyzickou zdatnost). Jejich využití v běžné praxi např. u praktického lékaře však brání jejich časová náročnost případně potřeba účasti dalších odborníků. Některá vyšetření se musí sbírat opakovaně, sběr dat u některých vyšetření tak trvá např. měsíc (hodnocení bolesti). FGA jsou tak využívaná spíše v geriatrických centrech, pro potřeby ústavní péče nebo výzkumné účely. [3]

Fyzická výkonost se hodnotí např. pomocí testu všedních činností (ADL). Pomocí tohoto testu lze vyhodnotiti, jak pacient zvládá běžné denní činnosti jako např. najíst se, obléct nebo využít WC. Ukázku tohoto testu můžete vidět na Obrázek 2.1.Pomocí testu se dá zjistit, zda je pacient soběstačný nebo lehce, středně či těžce závislý na pomoci další osoby/osob. [3]



Obrázek 2.1: ADL test

Pro hodnocení duševního zdraví je velmi rozšířen test MMSE(Mini Mental State Exam), který hodnotí paměť, orientaci, pozornost, řeč i konstrukční schopnost. Ukázku můžete vidět na Obrázek 2.2 .[3]



Mini-mental state examination

Datum vstupu: ☐☐.☐☐.☐☐☐☐ **Datum výstupu:** ☐☐.☐☐.☐☐☐☐

	Vstup	Výstup
1. ORIENTACE		
Jaký den v týdnu je dnes?	☐	☐
Kolikátého je dnes? Jaké je dnes datum?	☐	☐
Který měsíc v roce je nyní?	☐	☐
Který rok je nyní?	☐	☐
Jaké je nyní roční období?	☐	☐
Ve kterém státě jsme?	☐	☐
Ve kterém okrese jsme?	☐	☐
Ve kterém městě jsme?	☐	☐
Jak se jmenuje tato nemocnice (zdravotnické zařízení)?	☐	☐
Ve kterém poschodí se nacházíme?	☐	☐
2. ZAPAMATOVÁNÍ		
Bezprostřední reprodukce tří předmětů:		
citron / lopata	☐	☐
klíč / šátek	☐	☐
babička / váza	☐	☐
3. POZORNOST A POČÍTÁNÍ		
Opakované odečítání čísla 7 od čísla 100 nebo hláskování slova POKRM pozpátku		
100		
93 M	☐	☐
86 R	☐	☐
79 K	☐	☐
72 O	☐	☐
65 P	☐	☐

	Vstup	Výstup
4. PAMĚŤ, VÝBAVNOST		
Bezprostřední reprodukce tří předmětů z		
lopata	☐	☐
šátek	☐	☐
váza	☐	☐
5. POJMENOVÁNÍ		
Ukažte náramkové hodinky: Co je to?	☐	☐
Ukažte tužku: Co je to?	☐	☐
6. OPAKOVÁNÍ		
Opakování věty: "Žádné kdyby nebo ale."	☐	☐
7. TŘÍSTUPŇOVÝ PŘÍKAZ		
Porozumění (sdělený třístupňový		
"Vezměte tento papír do vaší ruky,	☐	☐
přeložte ho jednu na polovinu oběma	☐	☐
a položte na zem"	☐	☐
8. ČTENÍ A SPLNĚNÍ PŘÍKAZU		
Porozumění (písemný jednostupňový		
ZAVŘETE OČI	☐	☐
9. PSANÍ		
Napsání věty	☐	☐
10. OBKRESLOVÁNÍ		
Obkreslení předlohy průniku dvou	☐	☐
CELKOVÉ SKÓRE	☐☐	☐☐

Obrázek 2.2: Test MMSE

- **Sběr dat z funkčních geriatrických vyšetření**

Aktuálně jsou vyšetření sbírána ve formě papírových dotazníků. Následně jsou dotazníky skenovány a poté jsou v programu FormScan ručně zkontrolovány, zda jsou správně vyplněné, zda program rozpoznal všechny vybrané checkboxy a případně musí být dopsány chybějící texty. Poté jsou převedeny do formátu XML s předdefinovanou strukturou.

2.2 Předcházející závěrečné práce zabývající se podobnou problematikou

Jelikož se tato práce zabývá ukládáním a importem dat z funkčních geriatrických vyšetření do databáze a využívaná data prošla již určitým procesem (od digitalizace dotazníku a validace dat po export do definované XML šablony), bylo by dobré nejprve stručně zmínit obsah předcházejících prací, která se obdobným tématem již zabývala.

- **Diplomová práce Petr Šlajchrt**

V této diplomové práci se autor zabýval tvorbou pracovního prostředí umožňujícího vytváření, správu, vyplňování a zobrazování formulářů, především pro hodnocení rizika pádů lidí ve vyšším věku. Zároveň se práce zabývala i tvorbou laboratoře pro 3D záznam stereotypu chůze pacientů. V rámci diplomové práce byly také vytvořeny XML šablony a byla navržena aplikace v jazyce C# pro konverzi dat mezi touto XML šablonou a relační databází. Poslední částí práce byl import již existujících dat z gerontologického centra do databáze. [4]

- **Bakalářská práce Tomáš Janků**

V bakalářské práci řešil autor tvorbu komplexní webové aplikace pro práci s daty získanými z webových dotazníků. Vytvořená aplikace umožňuje import, export dat z elektronických formulářů, úpravy a vyplňování formulářů přímo ve vyvíjené aplikaci, dále umožňuje správu uživatelských účtů a přiřazení uživatelských rolí. Aplikace je schopná komunikovat s databází, na kterou jsou ukládána data z elektronických dotazníků. Aplikace dále umožňuje správu XML šablon, které mají předdefinované pojmenování (AAABCD.XML, kde AAA je identifikátor dotazníku, B je číslo, C verze dat a D udává verzi vzhledu). Do aplikace je možné importovat již vyplněné formuláře, a to buď ve formátu XML nebo v datových souborech. Dále má aplikace možnost exportu dat do MS EXCEL a jiných datových formátů. Data jsou ukládána do databáze. [5]

- **Bakalářská práce Pavel Šustek**

Bakalářská práce Pavla Šustka se zabývala vývojem webové aplikace pro digitalizaci papírových dotazníků, především v nich obsažených dat a jejich následným exportem do předdefinovaných XML šablon. V době vzniku práce existovala již desktopová aplikace FormScan, vyvinutá v rámci projektu GDiag, která umožňovala digitalizaci dotazníků i jejich následný převod do formátu XML. Desktopový program, ze kterého tato aplikace vychází, měl velkou nevýhodu v náročnosti ovládání, dále v nutnosti instalace na každou pracovní stanici zvlášť a ve špatné dostupnosti při aktualizaci. Avšak hlavní nevýhodou a tím pádem důvodem pro tvorbu webové aplikace bylo nepříliš intuitivní ovládání, nutnost instalace na každou jednotlivou stanici a složitá aktualizace (kvůli špatné dostupnosti). Aplikace vyvinutá v bakalářské práci Šustka měla tak za úkol především umožnit práci online, přijímat ofocené dotazníky přímo ze scanneru bez nutnosti do programu je ručně importovat, následně možnost přijaté dotazníky

zobrazit. Funkčnosti aplikace byly inspirovány již existujícím programem FormScan, případně byly doplněny nebo upraveny. Mezi tyto funkčnosti patří možnost ořezat digitalizovaný dotazník podle předdefinovaných rohů (jelikož systém OMR při objevování zaškrtnutých polí nepočítá s okraji), úprava kontrastu a barvy pro lepší čitelnost, možnost ručně opravit špatně určené rohy, přiřadit XML šablonu podle čárového kódu k dotazníku, převést XML šablony do HTML, automaticky číst zaškrtnutá pole (toho bylo docíleno aplikací OMR algoritmu, který čte zaškrtávací pole na předem definovaných souřadnicích). Zaškrtnutá pole jsou označena červeně, modře nenalezené nebo nevyplněné. Možná je i ruční oprava nenalezených checkboxů. Dalšími funkčnostmi jsou možnost ručního vkládání textových částí do HTML formuláře (např. komentáře, bodové ohodnocení apod.), validace vyplněných dat (kontrola, zda jsou klíčové části formuláře vyplněny, v případě rodného čísla se kontroluje i správný formát), převod zpět na vyplněnou XML šablonu. [6]

2.3 XML

Podstatnou částí této práce je získání dat z vyplněných XML šablon a jelikož struktura a některé elementy a atributy hrají velmi důležitou roli při výběru dat, bylo dobré nyní stručně uvést XML a následně popsat využívanou XML šablonu.

XML *(Extensible Markup Language)* neboli rozšiřitelný značkovací jazyk, který je softwarově a hardwarově nezávislým nástrojem pro ukládání a přenos dat. Stejně jako HTML bylo XML vytvořeno k popisu a uchování dat. Na rozdíl od HTML nemá XML tagy předdefinované, díky čemuž je možné si tagy přizpůsobit a uchovávat různorodé informace. Základními částmi XML jsou elementy, které se ve struktuře označují pomocí tagů, většinou párových (např. <formulář></formulář>). [7]

První řádka v XML dokumentu většinou obsahuje verzi a informaci o kódování (např. <?xml version="1.0" encoding="UTF-8"?>). Každý XML dokument musí obsahovat kořenový element, ve kterém jsou všechny ostatní elementy uzavřené. Elementy mohou dále obsahovat různé atributy, které se využívají pro uchování dalších informací, upřesnění významu apod. Atributy jsou uvedeny uvnitř tagů jednotlivých elementů a mají přiřazené hodnoty a název (např. <formulář název="form">, kde název je jméno atributu a form je hodnota atributu). Hodnoty atributů musí být vždy uzavřené v uvozovkách nebo apostrofech. [7]

- **XML schéma**

XML schéma popisuje strukturu XML dokumentu. Po ověření dokumentu s jeho schématem je dokument validní a tzv. dobře formátovaný.

Jelikož celá XML šablona pro geriatrické vyšetření je velmi obsáhlá, bude zde uvedeno jen zkrácené schéma s prvky důležitými pro tuto práci.

Elementy dotazníků mají určené pro sebe specifické parametry a také globální atributy, které jsou pro všechny stejné. Každý z dotazníků může obsahovat libovolný počet elementů různých typů, předdefinované atributy může i nemusí obsahovat. [5]

Prvním důležitým elementem je *<formset>*, který je komplexního typu a obsahuje jednotlivé formuláře. Každý jednotlivý formulář je definován tagem *<form>* a obsahuje další elementy a atributy, které jsou pro tuto práci důležité. Element *form* obsahuje ještě důležitý atribut *image*, ve kterém je uložen název obrázku s oskenovaným vyplněným dotazníkem ve formátu .tiff.

- **Základní elementy formuláře**

Uvnitř tagu *<form>* nalezneme sekvenci elementů, ze kterých může být vybrán libovolný počet. Tyto elementy jsou znázorněny na Obrázek 2.3. Elementy *label* a *title* slouží pouze k popisu, další jako např. checkbox, textfield, texarea nebo img již nesou data. Z dalších elementů jsou pro tuto práci velmi důležité právě elementy *rc* a *barcode*.

Element *rc* má v hotovém XML jako hodnotu rodné číslo pacienta a zároveň obsahuje atribut *hashcode* s hashem rodného čísla.

Element *barcode* označuje specifický čárový kód pro daný typ formuláře.

V elementu *date* je uloženo datum, nejčastěji datum vyplnění, datum vstupní nebo výstupní kontroly. Element má rovněž atribut *name,* podle jehož hodnoty se dá poznat o jaké datum jde (např. date je pro vstupní kontroly, outDate pro výstupní). Základní formát datumu je DDMMRRR.

```
−<xs:element name="form">
  −<xs:complexType>
    −<xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element ref="label"/>
        <xs:element ref="title"/>
        <xs:element ref="div"/>
        <xs:element ref="barcode"/>
        <xs:element ref="rc"/>
        <xs:element ref="date"/>
        <xs:element ref="checkbox"/>
        <xs:element ref="textfield"/>
        <xs:element ref="textarea"/>
        <xs:element ref="img"/>
      </xs:choice>
      <xs:attribute name="margin" type="defaultUnits" default="60px"/>
      <xs:attribute name="format" type="pageFormats" default="A4"/>
      <xs:attribute name="image" type="xs:string"/>
      <xs:attribute name="id" type="xs:string"/>
    </xs:complexType>
  </xs:element>
```

Obrázek 2.3: Elementy uvnitř tagu form

- **Specifické atributy**

Každý z elementů dotazníku obsahuje sadu atributů, opět volitelných. Příklad atributů pro element *checkbox* je znázorněn na Obrázek 2.4. Z uvedených atributů je důležitý *checked*, který pomocí hodnot true/false označuje, zda byl daný checkbox zvolen.

```
−<xs:element name="checkbox">
  −<xs:complexType>
      <xs:attributeGroup ref="SharedControlAttributes"/>
      <xs:attributeGroup ref="ControlShareAttributes"/>
      <xs:attribute name="size" type="defaultUnits"/>
      <xs:attribute name="checked" type="trueType"/>
    </xs:complexType>
  </xs:element>
```

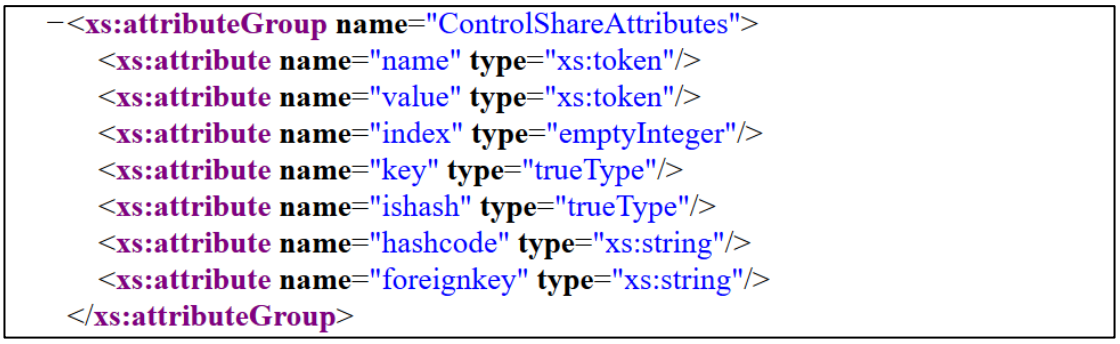
Obrázek 2.4: Atributy pro element checkbox

- **Globální atributy**

Elementy zároveň mohou obsahovat globální atributy ze skupin SharedTextAttributes, ControlShareAttributes a SharedControlAttributes.

První z uvedených skupin obsahuje některé atributy používané dále v této práci, a to především atributy nesoucí informaci o pozici elementů (např. checkboxu). Je to především atribut *top* udávající y-ové souřadnice prvku vzhledem k nadřazenému prvku a pak dále atribut *left* udávající x-ovou souřadnici vzhledem k nadřazenému prvku [5]. Vyplnění obou zmíněných atributů je zároveň povinné. Dalšími atributy této skupiny jsou například *width* a *height*, kterou jsou vyplňovány automaticky. Všechny čtyři zmíněné atributy akceptují hodnoty v px, pt, mm, cm, in, %, nejčastěji jsou ve vyplněných XML šablonách hodnoty uvedeny v px.

Atributy z druhé sady jsou znázorněny na Obrázek 2.5. Dále v této práci budou důležité atributy *name*, určující jméno elementua *value*, který udává hodnotu prvku. Atribut key (true/false) označuje, zda jsou tyto atributy klíčové, resp. zda hodnota atributu je ID dotazníku [5]. Dále při validaci vyplněných XML šablon se kontroluje, zda byly elementy s tímto atributem vyplněny, případně se kontroluje jejich správný tvar. Klíčové elementy jsou např. *rc, date* nebo *textfield*, který v atributu *department* nese název oddělení, ve kterém byl dotazník vyplněn. Atribut *foreignkey* definuje, zda hodnota elementu je cizí klíč z jiného dotazníku [5]. Ve třetí skupině jsou opět atributy udávající pozici elementu *(top, left, width, height* apod.)



Obrázek 2.5: Elementy ze skupiny ControlShareAttributes

2.4 EDC systémy

EDC neboli Electronic Data Capture systémy, jsou počítačové systémy vytvořené pro elektronický sběr klinických dat, především v oblasti výzkumu. EDC systém většinou umožňuje tvorbu reportů, validaci dat a sběr dat v uživatelsky přívětivém prostředí.

Jelikož pro ukládání dat bude využit právě některý z EDC systémů, bude nyní následovat popis několika vybraných nejpoužívanějších systémů. REDCap, který byl pro využití v této práci zvolen je popsán samostatně a podrobněji hned v následující kapitole.

- **Castor ECD**

Castor ECD je založen na cloudu a jde o komplexní systém pro řízení klinických studií a sběr dat. Castor je známý především díky jednoduchosti ovládání. Castor umožňuje tvorbu formulářů, sdílení průzkumů i tvorbu výpočtů a export i import dat. Mimo jiné umožňuje i kontroly v reálném čase, omezení viditelnosti polí dle různých podmínek. Zahrnuje ještě mnoho dalších funkcionalit, jako např. manažery kroků výzkumu nebo přizpůsobení pro eCRF. Jazykem je stejně jako u většiny ostatních ECD angličtina. [12]

- **OpenClinica**

OpenClinica je velmi rozšířený open source systém pro klinický výzkum. Software je opět webově založený a optimalizovaný pro cloud. Umožňuje vytvářet studie a projekty, sbírat a spravovat klinická data včetně jejich validace a normalizace, spravovat dotazy do databáze, správu rolí a uživatelských práv, vytvářet reporty a přístup k datům přes API (Application Programming Interface). OpenClinica umožňuje také rozsáhlou kontrolu postupu studie včetně upozornění na nutnost provedení dalších kroků, zvýraznění chyb a sporných informací. I tento systém nabízí přívětivé uživatelské prostředí, včetně možnosti zasílání dat přímo pacienty přes systém BYOD. Jazykových verzí existuje více (angličtina, francouzština, němčina…). [13]

- **Clinical Studio**

Clinical Studio je komplexní a robustní nástroj pro sběr dat a plánování studií, který je založen na cloudu. Tento systém umožňuje sběr dat, tvorbu formulářů, vytváření plánu sběru dat, kontrolu integrity dat a řízení přístupu k datům pomocí rolí. Přístup k datům je umožněn mimo webovou aplikaci také přes aplikaci mobilní a prostřednictvím API. Jazykem systému je opět angličtina. [14]

2.5 REDCap

REDCap neboli Research Electronic Data Capture je zabezpečená webová aplikace pro správu online výzkumů a databází a sběr medicínských dat. REDCap byl vyvinut na Vanderbilt Univerzity v Nashville a poprvé zveřejněn v roce 2004 [8]. REDCap byl vytvořen ve skriptovacím jazyce PHPa JavaScriptu a k ukládání dat využívá databázi MySQL.[9]

Systém může běžet jak na Windowstak na Linux (Apache) serverech. Standardní databáze pro REDCap vyžaduje tvorbu pěti odlišných tabulek:

- METADATA, kde jsou uloženy všechny údaje týkající se ukládání do databáze (např. typy datových polí),
- LOGGING, kam se ukládají všechny informace o změně dat,
- DOCS, sloužící k ukládání nahraných souborů nebo vygenerovaných exportů,
- PRÁVA, obsahující specifická práva pro řešitele a nastavení expirace,
- DATA, kam se ukládají všechna shromážděná data (jeden řádek odpovídá jednomu subjektu). [9]

REDCap není přímo open-source program, jelikož existují určitá omezení, např. je možné jej využívat jen pro nekomerční účely výzkumu a k distribuci je oprávněna pouze Vanderbilt Univerzity – Vanderbilt poskytuje software členům consortia, kteří ho dále poskytují výzkumným týmům. Zároveň licence zahrnuje kontrolu nad všemi publikacemi vytvořenými členy consortia zabývajícími se tímto softwarem a jeho metodikami. [9]

REDCap umožňuje všechny základní služby od nastavení práv uživatelů, importu dat a přístupu k nim, sběru dat přímo od účastníků, zobrazení základních statistik ve webové aplikaci nebo export dat či vytvořených formulářů do různých formátů. Nevýhodou REDCapu je nemožnost upravovat velké množství položek najednou, což se dá řešit exportem struktury dat do MS EXCEL, naplněním tohoto souboru a zpětným importem. [10]

Existují také rozšíření napsaná v PHP, která umožňují změnu vzhledu stránky (hooks) nebo programovanou práci s daty (hooks). Hooks jsou zároveň možností, jak vyřešit fakt, že základním jazykem REDCapu je angličtina. [10]

- **Tvorba projektů**

Při vytváření projektu i dodatečně lze nastavit přístupová práva uživatelů, typ projektu a další nastavení. Mezi tato nastavení patří např. možnost nastavit typ projektu na longitudinální, kdy je možné, aby jeden formulář byl vyplněn pro jedno ID vícekrát a zároveň se dá nastavit, které formuláře se budou vyplňovat při které návštěvě. Definování, které dotazníky budou vyplňovány, při které návštěvě probíhá pomocí definování tzv. events (událostí). Dále je možné větvení projektu (vytvoření více větví např. pro různé pobočky). Nastavit je také možné vytváření automatických ID účastníků.

- **Tvorba dotazníku/formuláře**

Základním prvkem práce s REDCapem je tvorba dotazníku/formuláře. Ten je možné přímo ve webové aplikaci vytvořit pomocí nástroje Online Designer nebo vytvořením tzv. datového slovníku v MSEXCEL a následným nahráním do REDCapu. Jednotlivé dotazníky se v aplikaci vytvářejí pomocí nástroje zvaného instrument. Důležitou součástí tvorby je pak výběr správného typu polí pro sběr dat (pro formát v jakém budou data ukládána). Zde jsou uvedeny některé typy polí, především ty, které jsou dále využívané v této práci:

  - Textfield, s možností validace dat jako datum, číslo, telefonní číslo a další,
  - multiple-choice question, kde se po zvolení odpovědi uloží této odpovědi definovaná číselná hodnota,
  - calculated field, kde je možné po zadání rovnice provádět výpočty třeba i s hodnotami doplněnými do jiných polí,
  - upload-file,který umožňuje nahrávat soubory,

U většiny typů polí je zároveň možné nastavit vyžadování vyplnění (atribut *required*), zda je položka osobní údaj (může být nastaveno, že při export bude vynechána), poznámku nebo další speciální nastavení. Jedním z dalších nastavení je tzv. branching logic, která umožňuje skrývat nebo zobrazovat otázky za předem definovaných podmínek (např. pokud je na otázku, zda má pacient proleženiny odpovězeno ano, zobrazí se otázka na počet). Dalším z těchto nastavení jsou tzv. Action tags, které umožňují nastavit defaultní hodnotu pole, pole jen pro čtení a mnoho dalšího.

- **Import dat**

Získávat data lze více způsoby. Nejsnadnějším způsobem je vyplňovat data přímo do dotazníků ve webové aplikaci nebo v mobilní aplikaci. Dalším způsobem je příprava dat v MS EXCEL s předem definovanou strukturou a následný import do REDCapu. Importovat se dají i data ve formátu CSV či XML. Data lze importovat i z již exitující databáze, pomocí dynamických SQL příkazů [10]. Posledním způsobem a zároveň způsobem, který je využívaný i v této práci je nahrání dat pomocí API.

- **REDCap API**

Pro možnost importovat data pomocí API musí mít uživatel příslušná práva a vygenerovaný token, který slouží jako prostředek pro ověřování všech přijatých API požadavků. Token je specifický pro každého uživatele a pro každý projekt. Stejně jako při zadávání dat přes webové rozhraní, dochází i při importu dat pomocí API k validaci dat. API mají zveřejněné ukázky ve více jazycích např. (PHP,Perl,Python,Ruby,cURL), ale na internetu je možné najít knihovny i pro jiné programovací jazyky. API je vestavěná funkce REDCap, takže není nutná žádná doinstalace. [11]

- **Statistiky a export dat**

U každého projektu je možné kontrolovat kvalitu dat – počet chybějících položek, počet chybějících povinných položek, nesprávný formát zadaných dat. Tuto kontrolu lze provést pomocí přehledových statistik přímo v REDCapu, kde je možné u každé otázky zobrazit počet vyplnění, počet chybějících dat, průměr, maximální i minimální hodnotu a další.

Data je zároveň možné exportovat v různých formátech: CSV, R, SAS, XML (CDICS ODM). Pro export je možné zvolit různá nastavení (vynechat osobní data, hashování ID, vynechání nevyplněných dat apod.). Také je možné exportovat pouze vybrané podmnožiny a jen některé události/návštěvy.

3 Cíle práce

Tato bakalářská práce má za hlavní cíl navrhnout a implementovat systém pro migraci již uložených dat z funkčních geriatrických vyšetření.

Během postupu byly stanoveny následující podcíle:

- Výběr vhodného databázového systému a jeho zprovoznění
- Přepis všech šablon dotazníků do webové aplikace REDCap
- Seznámení se se způsoby importu dat do REDCapu a využitím REDCap API
- Předzpracování dat ve formátu XML (doplnění dat důležitých pro import apod.)
- Tvorba aplikace na vynětí potřebných dat z formátu XML
- Rozšíření aplikace o část pro migraci dat do databáze
- Ověření funkčnosti na testovacích datech
- Přenos doposud sesbíraných dat do databáze
- Doplnění aplikace o uživatelský vstup (pouze konzolová podoba)

4 Návrh software

4.1 Požadavky na databázový systém

V závislosti na oblasti použití (využití Gerontologickým centrem) byly stanoveny následující požadavky na databázový systém:

- Možnost nainstalovat na vlastní server
- Stále vyvíjený software
- Technická podpora/manuály
- Možnost nastavení uživatelských práv
- Validní software
- Zabezpečení

4.2 Výběr databázového systému

Databázový systém byl vybírán na základě požadavků uvedených výše. Zde bude v rychlosti shrnuto, jaké mají systémy, ze kterých je vybíráno, požadavky na server nebo databázi, jak zajišťují bezpečnost apod.

REDCap je možné instalovat jak na Windows servery, tak i na Linux servery. Instalace na vlastní server je umožněna. REDCap vyžaduje databázový server s MySQL a nástroj pro správu databáze např. phpMyAdmin. Veškerá data uložená do REDCapu zůstávají uložená v MySQL databázi organizace, která si REDCap nainstalovala. Data tak nikdy nejsou přenášena nebo ukládána do organizace. Některá zabezpečení implementuje již rovnou samotný REDCap. Patří sem ověřování identity koncových uživatelů pomocí různých metod autentizace: LDAP, interní metoda autentizace založená na tabulkách, Shibboleth nebo kombinace LDAP a tabulkové metody. Zároveň jsou zde různá nastavení jako např. automatické odhlášení po stanovené době nebo odhlášení do delší době nečinnosti. Také se dá nastavit expirace uživatelských hesel, počet přípustných http požadavků za minutu, počet pokusů na přihlášení apod. REDCap také uschovává záznamy o veškeré aktivitě uživatele, včetně navštívených záznamů nebo projektů. Tyto záznamy je pak možné s příslušným oprávněním zobrazit na stránce Logging. Použití UTF-8 kódování navíc umožňuje využití češtiny.

Aby bylo zajištěno, že uživatelé REDCapu se dostanou pouze k datům, se kterými pracují nebo mají oprávnění s nimi pracovat, využívá REDCap uživatelská oprávnění. Každý uživatel v REDCapu má vlastní účet a má přístup k vlastním projektům nebo projektům, ke kterým mu ostatní uživatelé udělili přístup. Nastavení mohou být správci upravena, stejně tak existují obdobná nastavení i v rámci určitého projektu. Uživateli mohou být přidělena různá oprávnění, jako možnost zadávat data, exportovat data,

přidávat formuláře, zobrazovat záznamy o aktivitě apod. REDCap využívá mnoho metod k zabezpečení dat uložených v databázi. Využívá různé způsoby ochrany před SQL Injection, XSS (Cross-site scripting) nebo CSRF (Cross-site Request Forgery).

Castor EDC běží na virtuálních privátních serverech ve třech různých zemích na světě (Holandsko, Velká Británie, USA). Všechny využívané platformy jsou certifikované a vyhovují certifikacím ISO27001 a ISO9001 a dalším národním/nadnárodním standardům jako např.: HIPAA nebo NENE7510. Tyto servery jsou Linuxové, pravidelně aktualizované a uplatňují nejmodernější bezpečnostní principy. Neoprávněný přístup do datových center není možný (fyzické zabezpečení, kamerový systém). Zálohování probíhá 4x denně a zálohy jsou ukládány na různých geografických místech tak, aby se docílilo co největší bezpečnosti. Databázové servery nejsou přístupné z internetu. Samotná aplikace Castor ECD nabízí široké spektrum zabezpečení. Mezi tato zabezpečení patří požadavek na silné heslo, individuální účet pro každého uživatele, může být nastaveno i dvou faktorové ověření nebo pravidelné střídání hesel. Existuje zde možnost nastavení uživatelský práv. V aplikaci jsou využívány techniky k zamezení SQL injection, XSS a dalších běžných útoků. Citlivá data mohou být šifrována a klíče spravované mimo tento systém v aplikacích třetích stran. Castor je oficiálně certifikovaný v oblasti bezpečnosti informací (ISO 27001 – standardy pro zabezpečení informační bezpečnosti). Umožněn je i import pomocí API. [15]

OpenClinica vyžaduje Linux server nebo Windows server a databázový server s PostgreSQL. Mimo instalaci na vlastní server je možné využívat i úložiště v cloudu. Stejně jako ostatní systémy umožňuje i validaci importovaných dat, import pomocí Data Import File nebo API (REST nebo SOAP). Tato aplikace také umožňuje nastavení uživatelských práv formou rolí. [16].

Clinical Studio jako jeden ze způsobů zabezpečení nabízí opět nastavení uživatelských práv a validaci dat. Umožňuje mimo jiné využití API pro práci s daty. Clinical Studio taktéž ukládá kompletní auditní záznamy. Clinical Studio je založeno na cloudu, je možné si ale vybrat buďto společný nebo privátní cloud. Clinical Studio nabízí také snadnou možnost získat informace o jeho validitě. [17]

Aby bylo možné z výše vypsaných systému snadněji vybrat, byla vytvořena tabulka. Na Tabulka 4.1 můžete přehledně vidět, která kritéria splňují nebo nesplňují jednotlivé systémy.

Tabulka 4.1: Splnění požadavků jednotlivými systémy

	REDCap	***Castor ECD***	***Clinical Studio***	***OpenClinica***
Vyvíjeno kým	Vanderbilt University	Castor EDC	Crucial Data Solutions, Inc.	OpenClinica, LLC
Vlastní server	ANO	NE	NE	ANO
Databáze	MySQL	Postgree	-	-
Licence	akademická licence Vannderbilt University	podle využití (komerční, nekomerční)	různé podle využití	GNU LGPL
Cena	bezplatné pro členy consortia	od 2250€/rok	od $1875 měsíčně	dle verze (zdarma nebo placené)
Zabezpečení	ANO	ANO	ANO	ANO
Validní	ANO	ANO	ANO	ANO
Podpora	ANO	ANO	ANO	ANO
Uživ. práva	ANO	ANO	ANO	ANO

Po pečlivém posouzení všech kritérií a jejich splnění jednotlivými systémy byl vybrán systém REDCap jelikož splňuje všechny požadavky. Zároveň nabízí asi největší podporu formou obsáhlých uživatelských příruček, videí a existenci consortia, kam je možné obrátit se s problémy. Gerontologické centrum navíc již využívá databáze MySQL a získalo licenci pro REDCap.

4.3 Výběr způsobu importu dat

Jak již bylo zmíněno při popisu REDCapu, REDCap umožňuje více způsobů zadávání dat. Ruční zadávání by pro tolik dat nepřipadalo v úvahu, další způsoby (import dat pomocí excelovského dokumentu nebo pomocí API) ale mohou být pro import již sesbíraných dat uložených v XML využity.

- **Import dat pomocí API**

Import pomocí API vyžaduje vytvoření programu, který bude s REDCapem komunikovat podle předem definovaného způsobu. Existují ukázky metody při importu ve více programovacích jazycích, zveřejněné přímo v dokumentaci REDCapu, případně je možné stáhnout různé knihovny pro (Python, C# a další). Tento import vyžaduje vždy nejprve vygenerování tokenu (lze jednoduše v uživatelském účtu). Jelikož zde se již stejně tvoří program, není nutné vytvářet ještě další, speciální, který bude data z XML získávat, je možné programy spojit. Právě tento způsob byl po konzultaci s vedoucím práce zvolen. Důvodů pro zvolení této metody bylo několik: mimo import dat je možné využít i export, je možné rovnou importovat i obrázky.

- **Import dat pomocí Data Dictionary**

Tento typ importu vyžaduje nejprve vytvoření online dotazníků (resp. vytvořením těchto dotazníků se vytvoří i názvy proměnných pro data – názvy sloupců v databázi). Poté je potřeba stáhnout Data Dictionary (datový slovník) z REDCapu, což jde velmi jednoduše pro přihlášení do uživatelského účtu a následně už se jen dokument plní požadovanými daty. Jak takovýto dokument vypadá je možné vidět na Obrázek 4.1. Zda se budou psát hodnoty pro jednotlivé záznamy do řádků nebo sloupců, je možné zvolit. Na vzorovém obrázku se vždy text na řádce nahradí hodnotou pro záznam (hodnoty pro jednotlivé záznamy jsou oddělené čárkou). Ruční přepis dat do dokumentu by ale byl stejně náročný, jako přepis rovnou do webových dotazníků. Navíc se pomocí tohoto způsobu nedají vkládat obrázky (naskenované dotazníky). Proto by bylo potřeba vytvořit program, který by vybral potřebná data z XML a přenesl je do excelovského dokumentu.

	A
1	Variable / Field Name,Record,Record,Record,Record,Record,Record,Record,Record,Record,Record
2	record_id
3	redcap_event_name
4	rc
5	pacient_complete
6	department_adb000
7	date_adb000
8	in1_adb000
9	in2_adb000
10	in3_adb000
11	in4_adb000
12	in5_adb000
13	in6_adb000
14	inpmk_adb000
15	in7_adb000
16	in8_adb000
17	in9_adb000
18	in10_adb000
19	intotal_adb000
20	adb000_complete
21	department_adl000
22	date_adl000
23	adl1_2_adl000
24	adl2_2_adl000
25	adl3_2_adl000
26	adl4_2_adl000

Gerontocentrum_ImportTemplate_2

Obrázek 4.1: Struktura dat v Excelu pro import do REDCapu

4.4 Požadavky na aplikaci

Jelikož mimo výběr databázového systému je v této práci navrhován i způsob přenosu již sesbíraných dat do databáze (tvorba aplikace), byly stanoveny tyto požadavky, které by měla aplikace splňovat. Aplikace by měla umožňovat:

- Získat z XML souboru potřebná data
- Uložit data do databáze
- Uložit do databáze naskenované dotazníky
- Vytvořit nové ID záznamu v REDCapu, pokud pacient ještě neexistuje
- Přiřadit dotazník podle RČ k již existujícímu ID (pacientovi)
- Najít další volnou událost
- Označit ty dotazníky, které u kterých se vyskytla chyba, ale přesto byly nahrané
- Uložit seznam těch dotazníků, které nebylo do REDCapu možné importovat

4.5 Využité technologie

Pro vývoj aplikace byl využit programovací jazyk C#. Jako vývojové prostředí pro tento projekt bylo využito Microsoft Visual Studio Community 2017. Visual studio obsahuje editor kódu i integrovaný debugger. Nabízí také možnosti pro tvorbu aplikací s GUI, tvorbu databázových schémat apod. Mezi vestavěné jazyky patří C, C++, C# a VB.NET.

4.6 Knihovna RedcapApi

Pro komunikaci aplikace s REDCapem byla použita externí knihovna REDCapApi ve verzi 1.0.7 dostupná přes NuGet nebo GitHub [https://github.com/cctrbic/redcap-api]. Tato knihovna obsahuje předdefinované metody pro import i export záznamů, import souborů, export metadat a mnoho dalších. Všechny metody této knihovny jsou asynchronní. Obrázek 4.2 ukazuje jednu z těchto metod, metodu pro export záznamů.

```
Task<string> ExportRecordAsync(string token, Content content,[Return
Format format= ReturnFormat.json],[RedcapDataType redcapDataType =
RedcapDataType.flat],[string[] records = null],[string[] fieleds =
null],[string [] forms = null],[string[] events = null],[RawOrLabel
rawOrLabel= RawOrLabel.Law],[RawOrLabelHeaders rawOrLabelHeaders =
RawOrLabelHeaders.raw],[bool exportCheckboxLabel= false],[OnErrorFormat
onErrorFormat = OnErrorFormat.json],[bool exportSurveyFields =
false],[bool exportDataAccessGroups = false],[string filterLogic =
null] ) (+3 overloads)
```

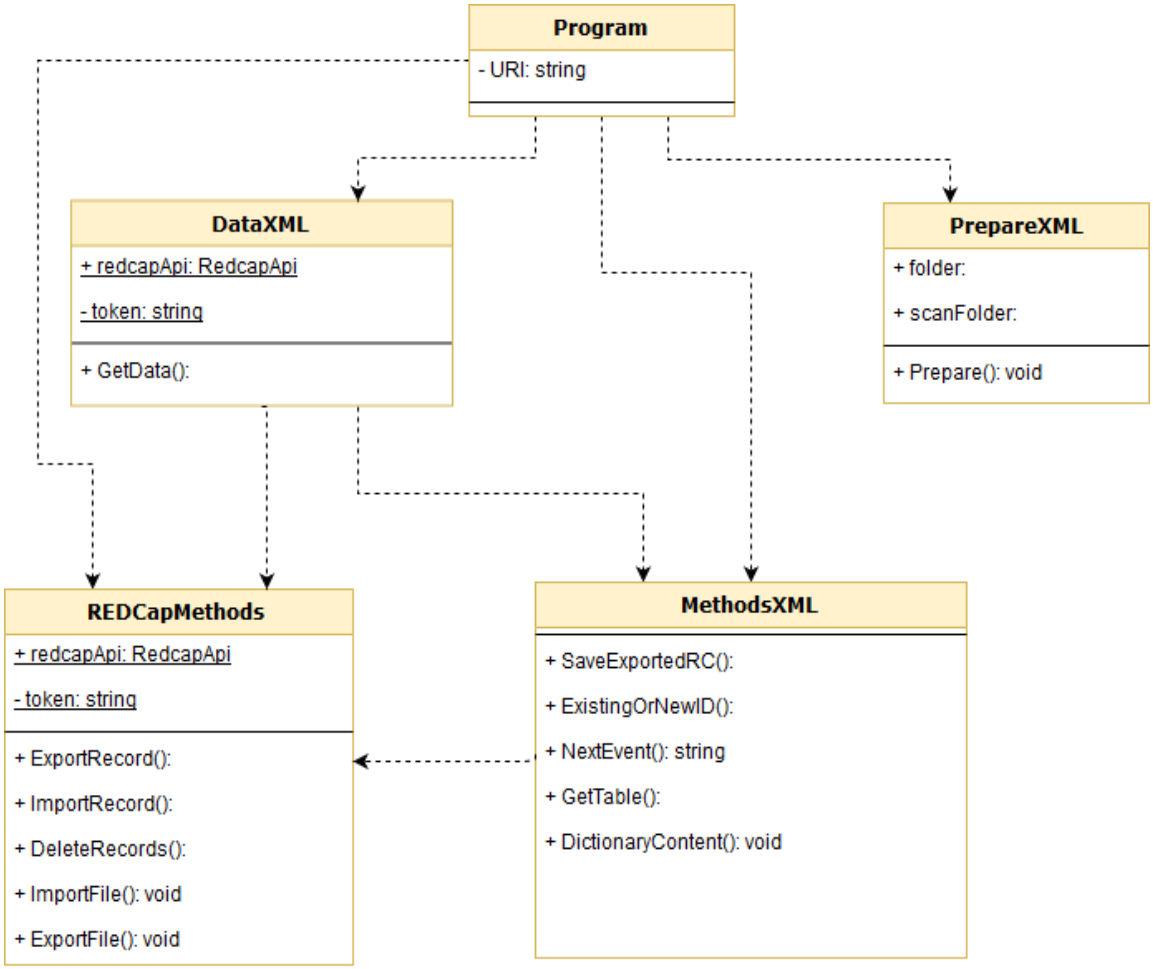
Obrázek 4.2: Metoda pro export záznamů

Pro každou metodu existuje více tvarů. Například pro metodu ukázanou výše je možné zvolit, v jakém formátu budou data vrácena a v jakém formátu budou chybové hlášky. Je možné vybírat z formátů (JSON,XML,CSV). Další z nastavitelných možností je Content, kde je možné nastavit, jaký obsah chceme vracet (např. pokud chceme exportovat záznam, je nejlepší zvolit možnost Record; pokud potřebujeme zjistit názvy polí v REDCapu pro daný instrument, existuje možnost ExportFieldNames atd.). Nejdůležitějšími parametry jsou ale token, fields (název pole/í v REDCapu), events (událost), forms (instrument resp. název dotazníku/ů) a records (ID záznamu/ů). Token je nutné zadat vždy, nemá přednastavenou žádnou defaultní hodnotu. Ostatní ze jmenovaných mají defaultní hodnotu, nicméně pokud tuto metodu využíváme, většinou je alespoň některý z parametrů doplněn.

Ostatní metody mají podobné parametry. Za zmínku stojí ještě metoda ImportRecordsAsync. Zde existuje opět více tvarů a je možné zvolit, v jaké formě budou data importována. Zda budou metodě předány ve tvaru Dictionary<string,string> nebo ve formě listu obsahujícím jednotlivé dictionary nebo jako objekt.

4.7 Diagram tříd

Pro lepší dokumentaci byl vytvořen UML diagram tříd, který můžete vidět na Obrázek 4.3. Všechny třídy včetně metod budou popsány v následující kapitole.



Obrázek 4.3: Diagram tříd

5 Implementace

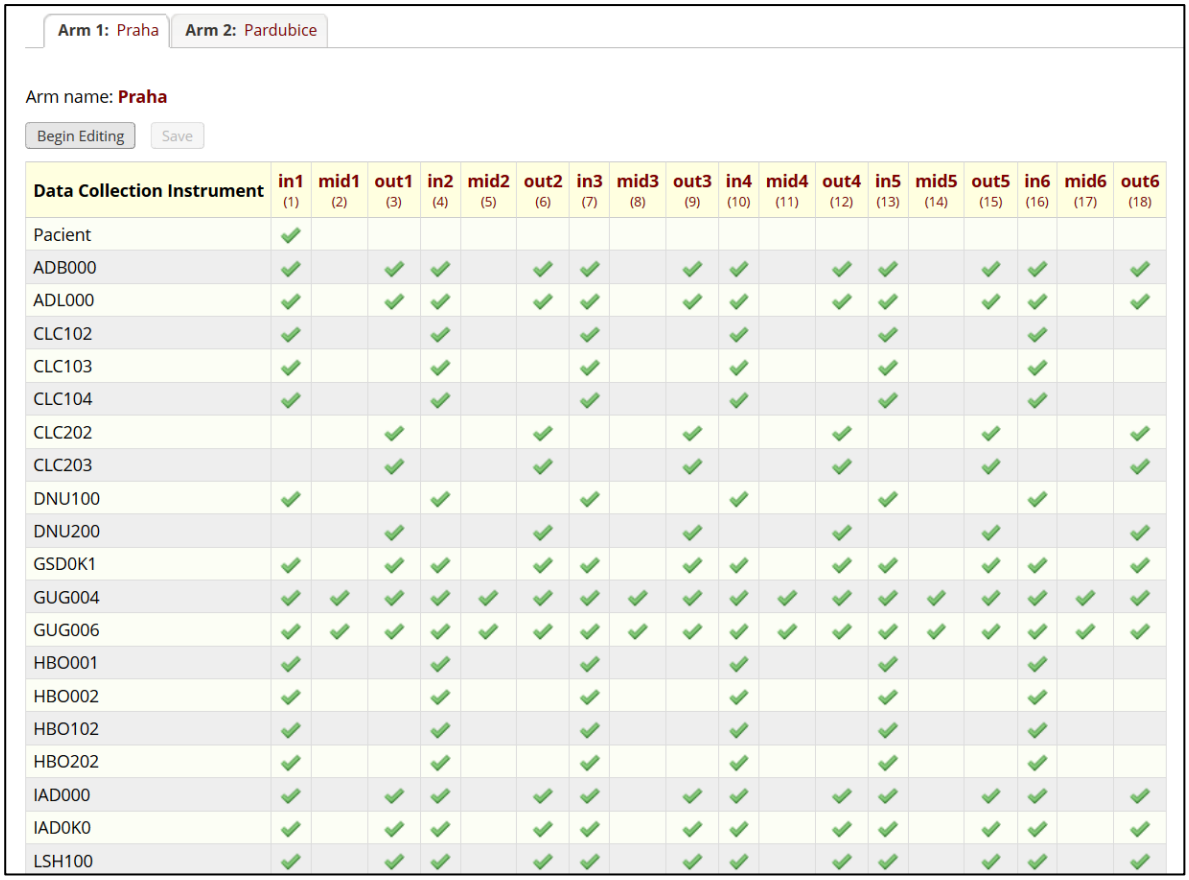
V této kapitole je popsán vytvořený projekt v REDCapu, včetně událostí a rozdělení pro dvě pobočky Gerontocentra. Další část této kapitoly tvoří podrobný popis aplikace (tříd a metod).

5.1 Redcap

V REDCapu byl vytvořen projekt nazvaný Gerontocentrum, který obsahuje formuláře, resp. jednotlivé funkční testy. Projekt je typu longitudinal (dlouhodobý), což znamená, že jeden dotazník může být sbírán vícekrát (pod jedním ID může být ten samý dotazník uložený vícekrát s rozdílnými daty). V tomto projektu odpovídá ID záznamu jednomu pacientovi. Pod jedním ID záznamu jsou tak uloženy všechny dotazníky, které byly v souvislosti s tímto pacientem vyplněny a další dotazník nazvaný Pacient, který obsahuje rodné číslo pacienta.

Jelikož sběr dat probíhal na dvou různých pobočkách Gerontocentra – v Praze a Pardubicích, jsou v REDCapu pomocí tzv. Arm odlišena i tato dvě centra (Obrázek 5.1).

Některé papírové dotazníky byly rozděleny na část pro vstupní, výstupní případně další data. Obdobně je tomu i v REDCapu. Jsou vytvořeny 3typy událostí, *in* pro vstupní data (dotazník se vstupními daty), *out* pro data výstupní (dotazník s výstupními daty) a *mid* pro data prostřední (data sbíraná mezi vstupem a výstupem). Jelikož se jedná o projekt typu longitudinal, může být dotazník vyplněn pro každý typ události vícekrát. Proto je definováno více událostí *in, out* i *mid* (Obrázek 5.1). Zelené značky u událostí znamenají, že daný dotazník byl pro tuto událost definovaný (např. CLC202 je tedy dotazník obsahující pouze data z výstupní kontroly, a naopak ADB000 obsahuje data jak ze vstupní, tak i z výstupní kontroly).

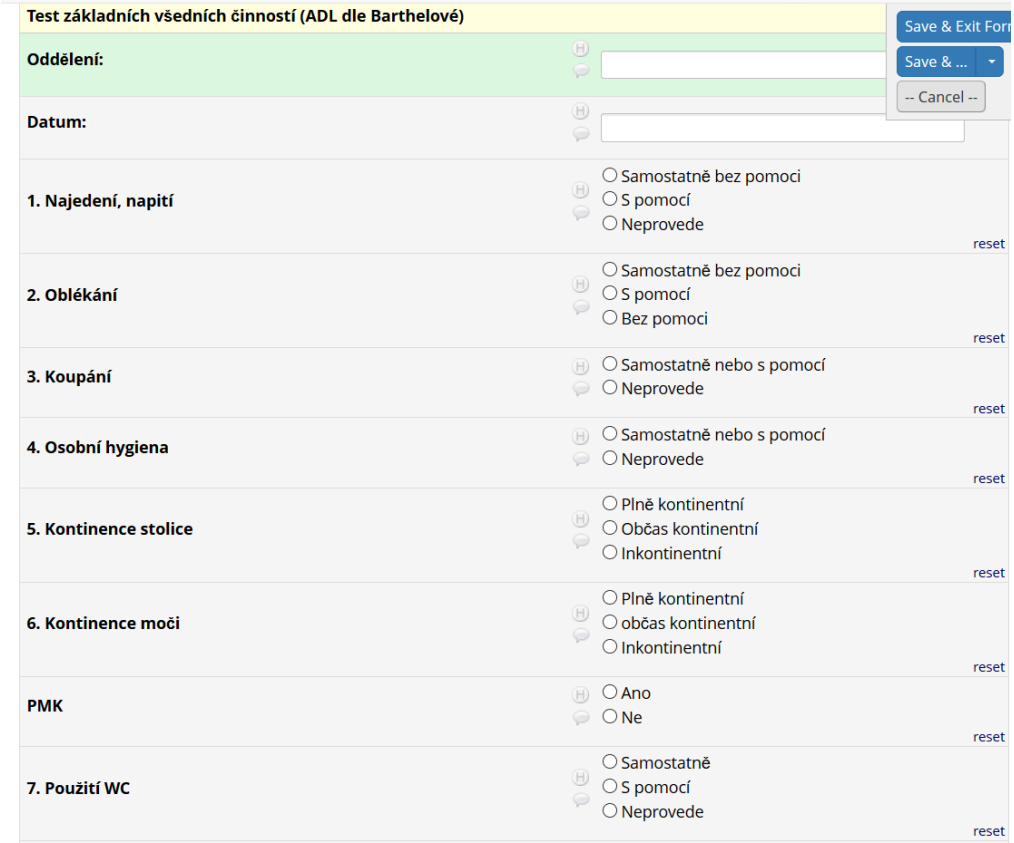


Data Collection Instrument	in1 (1)	mid1 (2)	out1 (3)	in2 (4)	mid2 (5)	out2 (6)	in3 (7)	mid3 (8)	out3 (9)	in4 (10)	mid4 (11)	out4 (12)	in5 (13)	mid5 (14)	out5 (15)	in6 (16)	mid6 (17)	out6 (18)
Pacient	✓																	
ADB000	✓		✓	✓		✓	✓		✓	✓		✓	✓		✓	✓		✓
ADL000	✓		✓	✓		✓	✓		✓	✓		✓	✓		✓	✓		✓
CLC102	✓			✓			✓			✓			✓			✓		
CLC103	✓			✓			✓			✓			✓			✓		
CLC104	✓			✓			✓			✓			✓			✓		
CLC202			✓			✓			✓			✓			✓			✓
CLC203			✓			✓			✓			✓			✓			✓
DNU100	✓			✓			✓			✓			✓			✓		
DNU200			✓			✓			✓			✓			✓			✓
GSD0K1	✓		✓	✓		✓	✓		✓	✓		✓	✓		✓	✓		✓
GUG004	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
GUG006	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
HBO001	✓			✓			✓			✓			✓			✓		
HBO002	✓			✓			✓			✓			✓			✓		
HBO102	✓			✓			✓			✓			✓			✓		
HBO202	✓			✓			✓			✓			✓			✓		
IAD000	✓		✓	✓		✓	✓		✓	✓		✓	✓		✓	✓		✓
IAD0K0	✓		✓	✓		✓	✓		✓	✓		✓	✓		✓	✓		✓
LSH100	✓		✓	✓		✓	✓		✓	✓		✓	✓		✓	✓		✓

Obrázek 5.1: Události a dotazníky pro ně definované

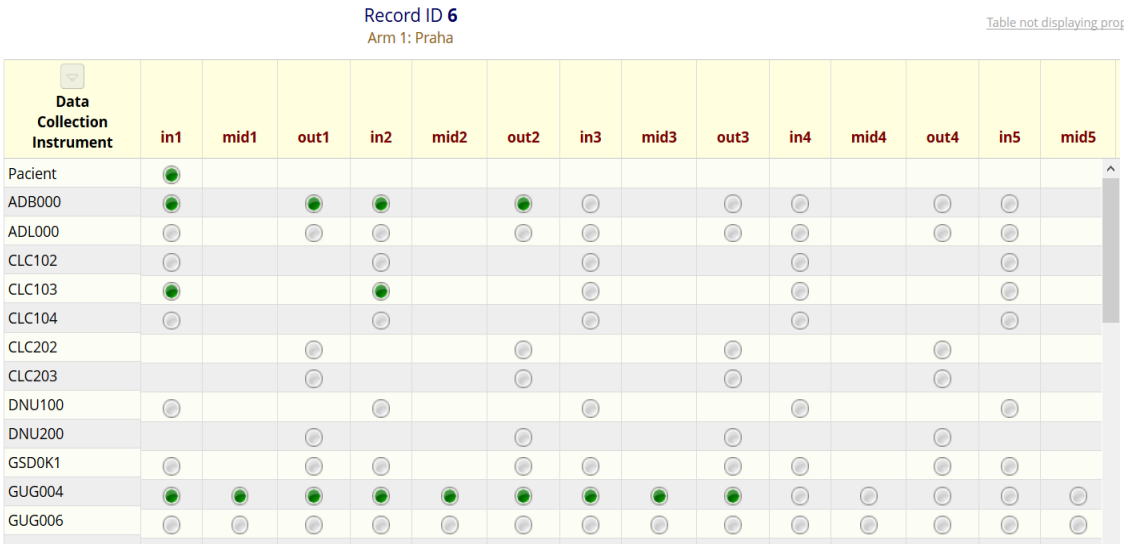
5.2 Popis struktury dotazníku – pole:

Každý dotazník obsahuje pole, která mohou být různého typu (textfield, checkbox, radio button, calculated field apod.) a z nichž každé má jiné možnosti např. výběru odpovědi apod. Velkým problémem se během řešení projektu ukázalo to, že názvy proměnných musí být pro každý dotazník unikátní. Například pokud dotazník „A“ obsahuje pole do kterého se vkládá datum a ukládá se do proměnné *date* nemůže se v dotazníku „B“ ukládat datum také do proměnné *date*. Vzorový dotazník pro jeden z funkčních testů ukazuje Obrázek 5.2. Pokud to bylo možné, byly proměnné v REDCapu pojmenovány stejně jako jsou pojmenovány atributy *name* u elementů v XML souboru, opakující se názvy proměnných byly povětšinou vyřešeny přidáním názvu dotazníku za název proměnné (in1_ADL000). Během tvorby dotazníků bylo zjištěno ještě několik požadavků REDCapu na názvy proměnných/polí: všechny názvy proměnných/polí musí být malými písmeny (pokud tomu tak není, REDCap si je na malá písmena stejně převede), názvy polí nesmí začínat číslem, názvy polí nesmí obsahovat pomlčku. Tato omezení bylo pro následnou implementaci velmi důležitá, jelikož těmito pravidly se názvy elementů v XML neřídily a musely být, proto zavedena některá opatření (nahrazení pomlčky znakem _, převedení názvů na malá písmenka nebo úplné přejmenování).



Obrázek 5.2: Dotazník ADB000 vytvořený v REDCapu

Hlavní pole v dotazníku je Record ID. Zde je do proměnné *record_id* uložen jedinečný identifikátor záznamu, v tomto případě odpovídá jedno ID jednomu pacientovi. Pro každé takovéto ID je ale u definováno více dotazníků. Právě díky tomuto je možné, aby byly všechny dotazníky patřící jednomu pacientovi pohromadě, tedy pod jedním ID. K lepšímu pochopení může sloužit Obrázek 5.3. U projektu typu longitudinal se toto pole vyplňuje pouze u prvního dotazníku v pořadí, v tomto případě tedy u dotazníku Pacient.



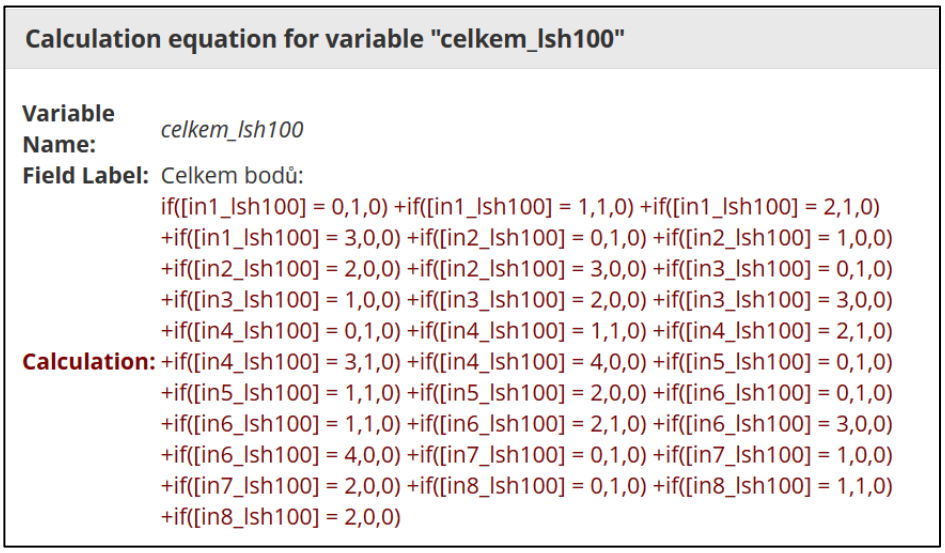
Obrázek 5.3: Vyplněné dotazníky pro jedno vzorové ID

Dalším polem obsaženým v každém dotazníku (mimo dotazník Pacient) je datum. Pole datum je pouze textové pole, kde se do proměnné ukládá datum sběru dat. Pole datum je důležité především dál v importu dat, kde se právě pomocí data (pomocí hodnoty atributu *name* u elementu *date*) rozlišují vstupní a výstupní data. U pole *date* je možné nastavit validaci, bohužel se ale žádný s přednastavených formátů nehodí na formát datumu, se kterým se pracuje. Proto validace nebyla nastavena. K názvu pole je vždy na konec přidán název dotazníku (vysvětleno výše, z důvodu opakujících se názvů polí).

Další pole, která dotazníky většinou obsahují jsou checkboxy s možností pouze jedné odpovědi, což je vlastně obsah testu – otázky a odpovědi. Jak bylo uvedeno dříve v práci, každá z odpovědí má přiřazenou hodnotu, která je poté uložena do odpovídající proměnné.

Většina dotazníků obsahuje také pole pro sčítání získaných bodů. Toto pole je vytvořeno pomocí tzv. calculated field, což je pole v REDCapu, které umožňuje pomocí rovnic sčítat hodnoty proměnných. Např. pokud máme otázku „Jaký je rok ?“ a možnosti „2017, 2018, 2019“ v REDCapu je možné každé odpovědi přiřadit hodnotu pro případ, že by byla zvolena. Pokud pacient vybere např. 2019 u kterého byla hodnota 10, uloží se do proměnné k této otázce hodnota 10. Calculated field pak pomocí předem definované rovnice hodnoty sečte a vrátí výsledný součet.

Velkým problémem při řešení práce se ukázal fakt, že každá odpověď pro multiplechoice otázky musí mít originální hodnotu. Tedy nelze, aby u jedné otázky byly dvě odpovědi, které by byly ohodnoceny 5 body. Ovšem právě tohoto jevu se v některých papírových dotaznících využívalo. Bylo proto navrhnuto řešení využívající právě calculated field a podmínkovou logiku. Odpovědi u otázky byly sice označeny různými hodnotami, ale pokud je vybrána odpověď s hodnotu např. 3, bude v dotazníku zvýrazněna správná odpověď, ale podle podmínky bude calculated field počítat s tím, že vybraná je odpověď s hodnotou 1. Obecný tvar vzorce je: *if([název_pole]číslo1,číslo2,číslo3),* kdy první číslo označuje, hodnotu odpovědi, další označuje zda byla odpověď vybrána(true=1,false=0) a poslední číslo je hodnota, kterou chceme vrátit. (viz. **Chyba! Nenalezen zdroj odkazů.**)



Obrázek 5.4: Příklad rovnice s podmínkami

Aby bylo možné ověřit, zda byly odpovědi správně bodově vyhodnoceny, obsahuje dotazník v REDCapu i proměnnou, kde je uložen ruční součet (viz. Obrázek 5.5).

Obrázek 5.5: Automatický součet a součet převzatý z vyplněného dotazníku

Poslední pole, které obsahuje každý dotazník (opět kromě dotazníku pacient), je pole, kam se nahrává naskenovaný dotazník (typ upload-file). Obrázek 5.5ukazuje mimo jiné pole pro nahrávání dotazníku už s nahraným dotazníkem.

Jelikož zde byl několikrát zmiňovaný dotazník Pacient, bylo by dobré zmínit, že jde o dotazník, který obsahuje pouze dvě pole, a to pole pro uložení rodného čísla pacienta (do jiného dotazník se neukládá) a pole Record ID (vysvětleno výše).

5.3 Převodní tabulky

Jelikož na papírovém dotazníku jsou data pro více kontrol (např. vstupní a výstupní), ale v REDCapu je každý dotazník pouze pro jednu návštěvu/jedno vyplnění dotazníku, byly vytvořeny převodní tabulky. V papírovém dotazníku jsou data pro vstupní i výstupní

kontrolu pojmenována odlišně (např. in1 a out1), ovšem v REDCapu se využívá stále jeden a ten samý totožný dotazník, tedy má pole pro ukládání dat stále stejně pojmenované. Důvodem, proč nebyl vytvořen dotazník naprosto přesně kopírující dotazník papírový, byla například špatná přehlednost takovéhoto dotazníku v REDCapu. V papírovém dotazníku je vždy jedna otázka a odpovědi pro *in* a *out* vedle sebe, což ale v REDCapu takto přehledně vytvořit nelze.

Převodní tabulka tak slouží k rozpoznání, která data jsou pro kterou událost (*in, out, mid*). Strukturu této tabulky viz Obrázek 5.6. V zásadě se jedná o jednoduchý textový dokument, kde jsou názvy proměnných odděleny čárkou. Tabulka se skládá ze 4 nebo 6 řádků, podle toho, zda byl dotazník definovaný i pro událost *mid*. Liché řádky obsahují názvy proměnných (resp. hodnoty atributů *name* v XML dokumentu), sudé řádky jim odpovídající názvy polí v REDCapu. Původně vytvořené převodní tabulky obsahovali pouze 3 řádky, každý řádek pro jednu událost, ale jelikož bylo často nutné nalézt v XML jiné názvy, než jaké jsou názvy proměnných v REDCapu, byla zvolena tabulka s 6 řádky.

```
2,inA,inB,inC,inD,inE,inF,inSum,height,weight,age
2,inA,inB,inC,inD,inE,inF,inSum,height,weight,age
outA,outB,outC,outD,outE,outF,outSum
inA,inB,inC,inD,inE,inF,inSum
```

Obrázek 5.6: Příklad převodní tabulky

5.4 Implementace v C#

Program pro import obsahuje celkem 5 tříd (i s třídou Program.cs) a lze ho pomyslně rozdělit do tří částí podle funkce (1. část pro úpravu XML souboru, 2. pro získání dat z XML souboru a 3. část pro komunikaci s REDCapem). Části nicméně nejsou striktně oddělené a prolínají se, proto bude program popsán po třídách.

Ve třídě program dochází pouze k načtení potřebných informací z textového souboru, který vytvořit uživatel a po vyzvání programem zadal cestu k němu. Jinak jsou v této třídě volány metody ostatních tříd, ale vlastní metody tato třída neobsahuje. Ještě by bylo dobré zmínit, že některé z uvedených metod (např. metoda pro export souborů) byly využívány během vývoje aplikace, ale pro import dat se nevyužívají. Byly zde ponechány z důvodu možných budoucích úprav a plánovaného využití částí zdrojového kódu ve webové aplikaci

5.4.1 Úprava XML souboru před importem

Úpravu XML před samotným výběrem dat původně nebylo zamýšleno provádět. V průběhu realizace práce se však tato úprava ukázala jako nezbytná, a to z mnoha důvodů. Prvním důvodem byly některé elementy a jejich atribut *name*, jehož hodnotu

REDCap neakceptoval. Byly to například názvy začínající číslem (1_1_a) nebo názvy obsahující pomlčku (in1-a). Dalším důvodem byly chybějící hodnoty (atribut *value*) u některých checkboxů. Jelikož dále v práci bude vysvětleno, že hodnota polí se u checkboxů získává právě na základě jejich atributu *value*, byla tato úprava nezbytná. Stejně tak pro import do REDCapu byly problémem zdvojené hodnoty pro jednu otázku, tedy např. byla otázka typu multiple-choice, kde všechny checkboxy byly pojmenovány „outA“. Tato otázka měla 3 odpovědi, kdy dvěma z těchto odpovědí byla přiřazena hodnota 1 a poslední hodnota 0. U REDCapu však není povoleno, aby v rámci jedné otázky měly dvě odpovědi stejnou hodnotu. Většinou byl tento problém vyřešen pomocí změny atributu *value* v XML souboru, tam, kde toto nebylo možné (např. kvůli následným sčítáním hodnot pro vyhodnocení dotazníku), bylo použito řešení pomocí calculated field přímo v REDCapu (viz. podkapitola Popis struktury dotazníku – pole:).

Úprava XML souboru před získáním potřebných dat z něj a importem je realizována pomocí třídy **PrepareXML.** Tato třída obsahuje jedinou metodu a to **Prepare()**, která bere jako parametry cestu k XML souboru a cestu ke složce s uloženými naskenovanými formuláři. Umístění složky s naskenovanými formuláři je potřeba proto, že výsledný upravený XML soubor bude uložen do nově vytvořeného souboru, který se uloží právě do složky k naskenovaným formulářům. K přepsání původního souboru nedochází z více důvodů, jedním z nich je možnost, že uživatel nebude mít tento původní soubor zálohovaný a tím pádem by mohl v případě chyby v programu přijít o data. V původním souboru odpovídají názvy elementů (i jejich hodnoty) názvům polí ve FormScanu. Po průchodu touto metodou však dojde u některých elementů k přejmenování, případně i ke změně nebo doplnění hodnoty. Z toho důvodu by pak při procházení XML souboru bylo obtížné (bez znalosti, které elementy byly změněny a jak) přiřadit, který element v nové XML odpovídá poli v původním formuláři.

Zpět k popisu metody. Metoda je typu void, tudíž nic nevrací, pouze před ukončení dojde k uložení upraveného XML souboru. Tento upravený XML soubor má již předdefinované jméno ReadyXML, cesta k umístění souboru se může pro každého uživatele lišit, avšak ukládá se do složky s naskenovanými dotazníky.

Myšlenka této metody je taková, že dochází k procházení XML souboru formulář po formuláři a v rámci formuláře element po elementu (viz. Obrázek 5.7). Kdykoliv je nalezen element *barcode* s vyplněným atributem *value,* je hodnota tohoto atributu uložena do proměnné. Element *barcode* totiž v atributu *value* obsahuje název formuláře (např. ADL000). Následně je právě podle názvu formuláře určováno, zda vůbec a případně co se bude v XML souboru upravovat. Nebylo totiž nutné upravovat všechny formuláře. Navíc u každého formuláře bylo nutné upravit něco jiného (nebylo možné zobecnit). U některých formulářů tak stačilo například doplnit všem checkboxům nový atribut *value* s hodnotou 1 a naopak u jiných formulářů muselo být přepsáno třeba i 15 atributů *name*. Obrázek 5.7 ukazuje příklad úprav pro formulář DNU100 v metodě **Prepare().**

```
public void Prepare(string folder,string @formScan)
{
    XDocument doc = XDocument.Load(folder);
    foreach (var el in doc.Root.Elements())
    {
        var name = "";
        foreach (var element in el.Elements())
        {
            if (element.Name == "barcode" && element.Value != null)
            { name = (element.Attribute("value").Value).ToLower();}

            mna000

            #region dnu100
            else if (name == "dnu100")
            {
                if (element.Attributes("name").Count() != 0 && element.Attribute("name").Value == "inSize1")
                {
                    if (element.Attribute("left").Value == "275px")
                    { element.Attribute("name").Value = "insirka"; }
                    else if (element.Attribute("left").Value == "409.5px")
                    { element.Attribute("name").Value = "invyska"; }
                    else if (element.Attribute("left").Value == "544px")
                    { element.Attribute("name").Value = "inhloubka"; }
                }
            }
            #endregion
```

Obrázek 5.7: Ukázka metody Prepare() a úprav pro dotazník DNU100

Další, co by bylo dobré vysvětlit je přejmenování elementů na základě atributů *left* nebo *top*. Vybírání elementů na základě těchto atributů je zavedeno z důvodu opakujících se jmen elementů. Např. u formuláře z Obrázek 5.7 je původním element pojmenován (má atrbitu *name* s hodnotou) „inSize1“ a element s tímto atributem *name* se vyskytuje v celém formuláři ještě 3x a pokaždé do něj má být vepsána jiná hodnota. Jelikož v celém formuláři se nevyskytuje další prvek, který by měl stejnou hodnotu atributu *left*, nemusí být v podmínce uvedeno ještě omezení jen na atribut *name=“inSize1“*. U některých formulářů toto ale neplatí.

Podmínka *atributes(„name“).Count !=0* slouží jako opatření před chybami, ke kterým dochází, pokud u elementu tento atribut vůbec není. U některých jiných dotazníků je obdobnou podmínkou ošetřen i atribut *left* nebo *top*.

5.4.2 Třída MethodsDataXML

Tato třída obsahuje hned několik metod využívaných třídou DataXML (ta bude popsána v následujícím bodu).

První z těchto metod je **SaveExportedRC(),** která bere jako argument exportovaná data z REDCapu a následně v nich najde rodná čísla a k nim odpovídající *record_id*. Hledání rodných čísel a ID probíhá následovně: z REDCapu byla pomocí metody **ExportRecords()** exportována data, která obsahují všechna ID záznamů, všechna rodná čísla a ještě dvě další pole, která ale nejsou pro vysvětlení důležitá. Do této metody již vstupují data ve formátu string a jelikož jednotlivé dotazníky jsou uzavřeny ve složených závorkách, jsou všechna data pomocí metody Split() rozdělena na formuláře a uložena.

V každém formuláři jsou názvy polí a data oddělena uvozovkami (např.:“record_id“:“4“), proto je znovu využita metoda Split () a jednotlivé formuláře jsou rozděleny na názvy polí a hodnoty. Jelikož bylo pokusy s exportem zjištěno, že po rozdělení formulářů bude vždy na 3. místě ID záznamu a na 11. místě rodné číslo, jsou z těchto pozic pro každý formulář uloženy dvojice RČ – ID. Tyto dvojice jsou přidány do Dictionary<string,string>, kde klíčem je rodné číslo a hodnota odpovídá ID. Metoda vrací právě dictionary s těmito hodnotami.

Další metodou této třídy je **ExistingOrNewID(),** která na vstupu bere v předchozí metodě vyexportovaná RC a pak současně zkoumané RČ (RČ, které bylo nalezeno v právě prohledávaném formuláři). Zde se nejprve pomocí metody pro dictionary, ContainsKey(), zjistí, zda momentálně zkoumané rodné číslo je již v dictionary (tedy, zda je již v REDCapu a pacient s tímto rodným číslem už tam má nějaké dotazníky). Pokud tomu tak je, nalezne se pomocí klíče (RČ) odpovídající hodnota (ID) v dictionary. Pokud v dictionary rodné číslo nebylo nalezeno, znamená to, že pacient ještě v REDCapu nemá svoje ID, tedy nemá ještě žádné vyplněné dotazníky. Musí být tedy vytvořeno nové ID. Nové ID se získá pomocí získání hodnoty pro poslední klíč v dictionary a přičtení 1. Poslední klíč v dictionary a jeho hodnotu lze využít jen díky tomu, že data exportovaná z REDCapu využitá v předchozí metodě jsou seřazena vzestupně podle ID záznamu. Na posledním místě v dictionary tak je nejvyšší zatím použité ID. Původně bylo zamýšleno využít pro tvorbu nového ID metodu **NextRecordName()** knihovny RedcapApi, avšak ta z neznámého důvodu v tomto programu ne vždy fungovala. Použít tuto metodu bylo v plánu především proto, že aktuálně využitý způsob nepočítá s vynechanými ID (např. 4 5 8 9 12). Pokud tedy dojde ke smazání některého ID, které zrovna není nejvyšším, jeho místo se již nezaplní. Avšak následně bylo i v dokumentaci k metodě zjištěno, že ani metoda **NextRecordName()** by tento problém nevyřešila, jelikož také bere nejvyšší zatím použité ID a jen zvětšuje o 1.

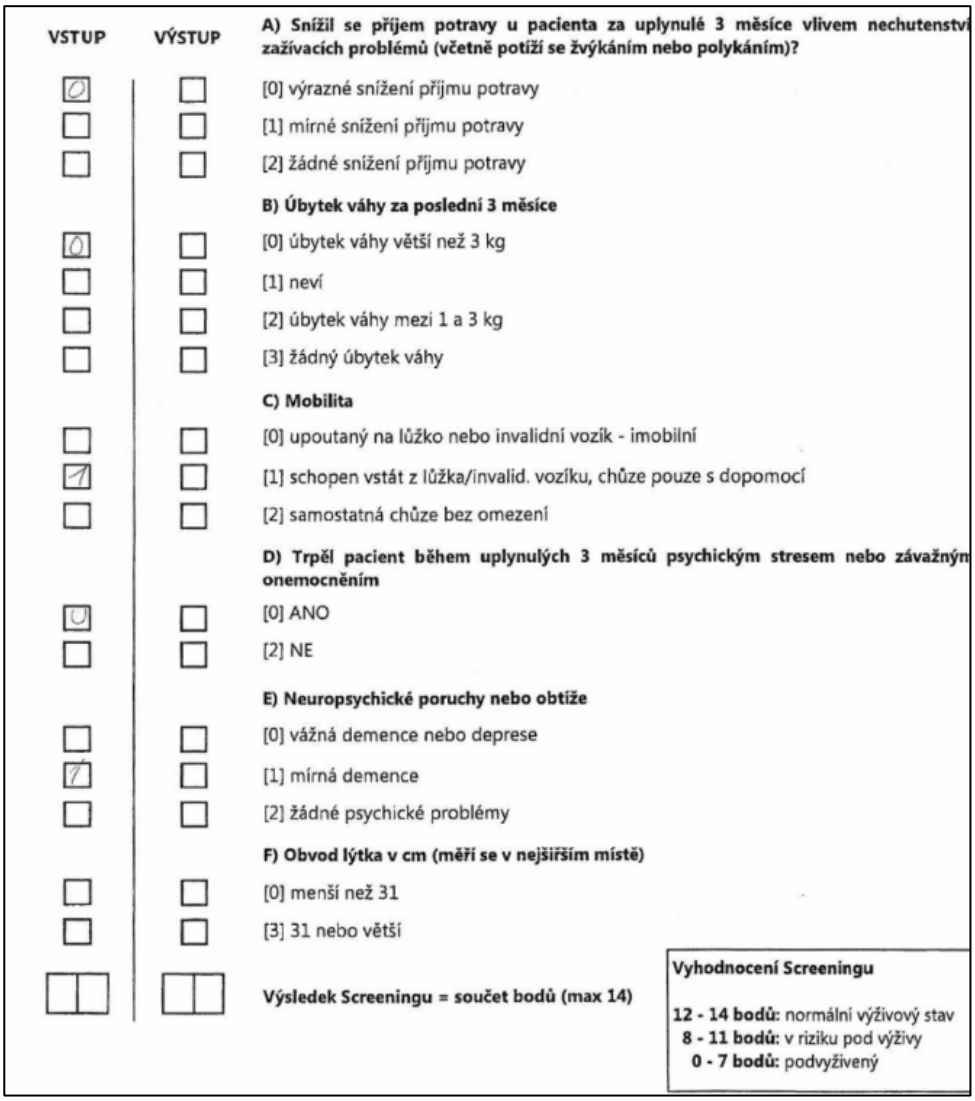
Následuje metoda **GetTable(),** která se využívá hned v začátcích prohledávání formuláře. Tato metoda bere jako argument jméno formuláře (např. ADL000). Podle tohoto jména je prohledán adresář, ve kterém jsou uložené převodní tabulky (vysvětleno výše v této kapitole). Nalezená převodní tabulka je následně pomocí StreamReader načtená, poté metodou Split() rozdělená na jednotlivé proměnné a uložená do 2D pole. Tabulka je dále využívaná třídou DataXML.

Další důležitou metodou je **NextEvent(),** což je metoda, která hledá volné události pro vložení formuláře. Jelikož projekt v REDCapu je longitudinální, může se formulář pro jednoho pacienta (pod jedním ID) vyplnit vícekrát, pokaždé pro jinou událost (event). Tyto události jsou v REDCapu již vytvořené (bylo vysvětleno výše v této kapitole) a jsou pojmenované podle toho, zda jde o vstupní, výstupní nebo prostření data (*in,out,mid*). Pokud by se nehledala volná událost a vždy by se data do REDCapu importovala do stejné události nebo případně s neudanou událostí, přepisoval by se jeden a ten samý dotazník v jedné a té samé události stále dokola. Toto bylo nutné ošetřit a k tomuto účelu vznikla

nyní popisovaná metoda. Tato metoda využívá metodu knihovny RedcapApi **ExportRecords().** Na vstupu této metody je jméno formuláře, ID záznamu, pobočka (arm), tři událostí (*in, out* a *mid*) a převodní tabulka. Jelikož metoda bere mnoho argumentů, budou nyní alespoň ty, které hrají důležitou roli popsané. Převodní tabulka je zde z toho důvodu, že na prvním místě (v prvním řádku i sloupci) obsahuje počet událostí, pro které je tento formulář v REDCapu definovaný. Počet může být 1-3, kdy 1 označuje buď událost *in* nebo *out*, 2 *in* i *out* a 3 ještě přidává událost *mid*. Na co je počet událostí využíván bude vysvětleno dále v odstavci. Ještě k událostem je nutné dodat, že na vstupu metody se zadávají 3 události, ať už je formulář definovaný pro kterýkoliv/kterékoliv z nich. Událost pojmenovaná jako *thisEvent* je událost, pro kterou se aktuálně hledá volné místo, *opositEvent* je většinou *in* nebo *out* a *opositEvent2* je téměř vždy *mid*. A nyní k popisu samotné metody. Pomocí metody **ExportRecord()** knihovny RedcapApi se vyexportuje pole XXX_complete (kde XXX je název formuláře). Jak je umožněno, že se vyexportuje pouze toto pole, pro daný formulář a danou událost bude vysvětleno při popisu třídy REDCapMethods dále v této kapitole. Toto pole má hodnotu podle toho, zda pro danou událost již dotazník je vyplněný nebo není. Může nabývat hodnot 0-2, kdy 2 znamená uložený dotazník, 1 znamená rozpracovaný dotazník (může být uložený, ale není označený jako hotový) a 0 znamená volné místo. Pokud se tedy pro událost např. in1_arm1 vyexportovalo „MKT102_complete“: „2“, po následném rozdělení na jednotlivé položky pomocí Split (˝“˝) se na třetím místě nachází 2, znamená to, že tato událost je již obsazená vyplněným dotazníkem a je potřeba najít jinou událost. Jelikož název události je složen z několika proměnných (*event+j+arm -> in1_arm1*), stačí pouze zvyšovat hodnotu proměnné j od 1 do počtu definovaných událostí v REDCapu a zjišťovat hodnotu pole XXX_complete. Pokud je nalezena volná událost (XXX_complete s hodnotou 0), jsou dvě možné varianty. Pokud je dotazník definovaný pouze pro jednu událost, průchod metodou končí a vrací se hodnota j neboli číslo události. Pokud je ale dotazník definovaný pro více událostí, začíná hledání pro další událost. Příklad může být následující: pro in3_arm1 je nalezena 0, jelikož je dotazník definovaný i pro *out*, prohledá se i událost out3_arm1. Pokud je nalezena 0, je možné importovat, pokud ne, je nutné vrátit se k hledání události zpět u *in*. Pokud by byl dotazník definovaný i pro *mid*, následovalo by po nalezení 0 u *out* ještě prohledání příslušné události i u *mid*. Jelikož v REDCapu je definovaný podle subjektivního názoru autora dostatečný počet událostí, nemělo by dojít k případu, kdy nebude volná událost nalezena. Kdyby k tomuto ale došlo, bylo by nutné postupovat podle pokynů v uživatelské dokumentaci (viz. Přidání nové události). Ještě by bylo dobré zmínit, proč program prohledává i ostatní události, pokud už nalezne volno u jiné události. Toto je z důvodu, že i když je jeden papírový dotazník rozdělený na 2 poloviny pro *in* a *out* (viz. Obrázek 5.8), ne vždy jsou obě poloviny vyplněny, a tudíž se importují data například jen do události in1. V případě, že by se následně importoval dotazník, ve kterém jsou ale obsažena data pro *in* i *out*, došlo by k roztržení dotazníku, resp. *in* by se importovalo do in2, ale *out* ještě do out1. Aby nedocházelo k roztržení dat patřících do jednoho dotazníku, bylo zavedeno opatření

zjišťující volno i v dalších událostech. Poslední, co je s touto metodou nutné zmínit je problém týkající se právě pole XXX_complete. Stav dotazníku (hodnotu 0-2) musí uživatel nastavit ručně (v případě tohoto programu jí nastaví program sám), REDCap neoznačí dotazník obsahující data automaticky 1 nebo 2. V případě, že by byl některý dotazník vyplňován ručně a vyplněný dotazník by nebyl označen jako complete, byl by v případě importu dat tímto programem s velkou pravděpodobností přepsán (resp. event, na které je dotazník uložen, by tento program bral jako prázdnou).

Poslední metoda této třídy **DictionaryContent()** už slouží jen k výpisu obsahu jednotlivých dictionary. Je typu void a vstupem je dictionary, který chceme po dvojicích (key value) vypsat.



Obrázek 5.8: Dotazník rozdělený na část pro vstupní a výstupní data

5.4.3 Třída REDCapMethods

Tato třída obsahuje metody z knihovny RedcapApi, ale jelikož se využívají vícekrát s různými parametry, bylo nakonec rozhodnuto o jejich umístění do samostatné třídy.

Nejvyužívanější metodou v programu je **ExportRecord().** Proto budou nyní popsány vstupní parametry této metody a jejich modifikace pro různá použití v tomto programu (export RČ, export volných událostí). Metoda bere jako argument ID záznamů, názvy dotazníků, polí a událostí (viz Ukázka kódu 5.1). Dále obsahuje metodu **ExportRecordsAsync()** knihovny RedcapApi.

```
publicstaticasync Task<string> ExportRecord(string token, RedcapApi
redcapApi, string[] records, string[] fields, string[] events,
string[] forms)    {
Task<string>exportRecordResult=redcapApi.ExportRecordsAsync(token,
Content.Record, ReturnFormat.json, RedcapDataType.longitudinal,
records, fields, forms5, events5, RawOrLabel.raw,
RawOrLabelHeaders.raw, false, OnErrorFormat.json, false, false,
null);

string resultString = await exportRecordResult;
return resultString; }
```

Ukázka kódu 5.1: Metoda ExportRecord()

V případě využití pro export volných událostí dotazníku, je nutné nastavit forms na jméno požadovaného dotazníku a events na jméno prohledávané události. Records musí obsahovat ID, které již existuje a pod kterým má být uložený i tento dotazník. Fields musí být nastaven na XXX_complete.

Pokud se využívá metoda pro export všech rodných čísel a ID záznamů, které REDCap zatím obsahuje, nastaví se records na defaultní hodnotu null, aby se exportovaly informace pro všechny ID, fields musí obsahovat *record_id* a *rc*, events a forms jsou taktéž nastaveny na null.

Další metodou je **ImportRecords(),** která bere na vstupu data pro import. ImportFile() obsahuje metodu **ImportRecordsAsync()** knihovny RedcapApi, která byla popsána již v části Návrh. V části Návrh bylo již nastíněno, že data mohou být v různých formách. V této práci jsou data pro metodu uložená v Dictionary<string,string> a následně více dictionary v jednom listu. Nejprve bylo v této práci využito uložení dat jen v Dictionary<string,string> a následný import. Tímto způsoben je možné importovat vždy jen jeden dotazník pro jednu událost. Díky využití List<Dictionary<string,string>> je možné importovat minimálně ten samý dotazník pro více událostí. Dle popisu metody by mělo být možné importovat více dotazníků současně, ovšem zde nastal problém

s určením volných událostí. Volné události jsou získávány pomocí exportování událostí z REDCapu, pokud by se ale nejprve třeba 100 dotazníků uložilo do listu a následně se importovaly, nebylo by možné tímto způsobem určovat volné události. V práci se tak importuje každý dotazník sám, pokud ale z jednoho dotazníku bylo možné získat data pro např. *in* a *out*, importují se naráz. Dalším problémem s listem byly stejné události, pokud byly v listu uloženy dva dictionary, které obsahovaly stejnou událost, REDCap import dat nepovolil.

Předposlední metodou je **ImportFile(),** která zajišťuje import souborů. Metoda bere jako argument ID záznamu, jméno a cestu k souboru, jméno dotazníku a událost. Importovat soubor se dá pouze do již existujícího dotazníku, není možné importem souboru vytvořit nové ID. Tato metoda musí mít při importu souboru omezen parametr field na název pole určeného pro import souborům (v REDCapu typ pole upload file).

Poslední metodou je **ExportFile(),** který exportuje soubory uložené v REDCapu. Tato metoda není při importu dat využívaná, byla využívaná během vývoje aplikace a byla zde ponechána kvůli budoucímu vývoji nebo využití.

5.4.4 Třída DataXML

Tato třída je nejdůležitější třídou celého programu, jelikož v ní dochází k získání dat z XML souboru a následně s použitím metod jiných tříd i k importu do REDCapu. Tato třída obsahuje pouze jednu, ale zato obsáhlou metodu **GetData().** Tato metoda přijímá z třídy Program jeden element form, dictionary s rodnými čísly, číslo pobočky a cestu ke složce s naskenovanými dotazníky. Jelikož zde dochází i k importu dat, jsou v této metodě vytvořeny 4 další dictionary pro import dat do události *in, out, mid* a pro dotazník Pacient. Zároveň je zde mnoho pomocných proměnných, které ale není nutné pro pochopení metody popisovat.

Prvním úkolem, který má tato metoda, je uložení názvu naskenovaného dotazníku. Ten je uložený v atributu image u elementu *form* a následně právě toto jméno slouží k prohledání složky s dotazníky a vybrání správného dotazníku.

Metoda následně prochází všechny elementy, které rodičovský element *form* obsahuje. Z těchto elementů je potřeba získat všechna data, která se mají importovat do REDCapu. Jelikož jsou tato data pro každý formulář jiná, musí se nejprve zjistit jméno aktuálně zpracovávaného formuláře.

Toto jméno se nachází v elementu *barcode,* v jeho atributu *value*. Jak již bylo zmíněno v části zabývající se REDCapem, REDCap akceptuje proměnné obsahující pouze malá písmena. Jelikož se i jméno formuláře bude využívat pro jména proměnných, je nejprve převedeno metodou ToLower() na malá písmena. Tato metoda je používaná u více názvů proměnných, právě z výše uvedeného důvodu. Jakmile je zjištěn název formuláře, je možné načíst převodní tabulku (viz. Metoda GetTable() třídy MethodsDataXML).

Dalším elementem, který by měl obsahovat každý formulář a bez kterého není možné přiřazení formuláře pacientovi je *rc* – rodné číslo. Tento element má jako hodnotu rodné číslo pacienta a následně je za použití metody NewOrExistingID() třídy MethodsDataXML zjištěno, zda pacient s tímto ID již v REDCapu existuje nebo je potřeba vytvořit nové ID. Po určení ID a rodného čísla se tyto informace uloží do dictionary Pacient, které bude sloužit k importu dat do dotazníku pacient. V předchozích odstavcích bylo zmíněno, že pro každý záznam, který se bude ukládat do REDCapu je potřeba nejprve zjistit volnou událost. U dotazníku pacient toto ale neplatí, neboť je definován a importován vždy jen do události in1_armX. Mimo RČ a ID záznamu, se do dictionary ukládá název pole *pacient_complete* s hodnotu 2, což označuje, že tento dotazník je vyplněný a v REDCapu u něj bude svítit zelené kolečko. Ještě by bylo dobré zmínit, že každé dictionary, které je v programu import dat musí obsahovat dvojici „record_id“ „HODNOTA“ a také by mělo obsahovat dvojice „redcap_event_name“ „HODNOTA“ a „XXX_complete“ „HODNOTA“.

Zpět k elementu *rc*. Po načtení příslušných dat do dictionary pacient se ještě do dictionary pro *out* a *mid* uloží dvojice „record_id“ „hodnota“. Do dictionary pro *in* se tato dvojice uloží jen v případě, že dictionary není prázdné. Je to z důvodu elementu *date,* který je v každém dotazníku ještě před elementem *rc* a pokud dotazník obsahuje data pro událost in, datum uložené v tomto elementu se do dictionary již uložilo. Element *date* bude probrán dále. Pokud není dictionary pro *in* prázdné, hledá se i volná událost a to na základě tří možností: pokud RČ nebylo nalezeno v dicitonary RC, nastaví se událost automaticky na 1 (tedy in1_arm_X), pokud RČ existuje spustí se metoda **NextEvent()** třídy MethodsDataXML a najde se vhodná volná událost. Třetí možností je, že RČ existuje, ale číslo volné události již bylo nalezeno (např. od dat pro *out* nebo *mid*), není tudíž nutné událost znovu hledat, pouze se jméno události vytvoří pomocí již nalezeného čísla (in+eventName_arm_X). Tato možnost není pravděpodobná, nicméně při odchylkách v uspořádání XML dokumentu je možná. Důvodem, proč toto hledání událostí není aplikované rovnou i pro dictionary *out* a *mid* je jednoduchý. Tyto dictionary jsou v této fázi průchodu elementy formuláře s největší pravděpodobností prázdné, jelikož jediné, co se do nich do této doby mohlo uložit byl datum, ale první element *date* obsahuje datum téměř vždy pro *in*. Ostatní elementy *date* pro *out* nebo *mid* bývají ve formulářích umístěny dále. Bylo by tak zbytečné ukládat do těchto dictionary další hodnoty nebo dokonce používat metody pro zjištění volných event (program by se zpomalil).

Dalším důležitým elementem je již zmíněný *date*, který jako hodnotu obsahuje datum vyplnění dotazníku. Zda se jedná o datum pro *in, out* nebo *mid* se rozhoduje na základě jeho atributu *name* viz.

Následně je důležitým elementem ten element, který obsahuje v atributu *name* hodnotu department. Tento element drží informaci o oddělení, na kterém byl dotazník vyplňován.

Toto byly všechny elementy, které mají formuláře společné. Další část už je pro každý formulář jiná, jelikož některý obsahuje data z checkboxů, jiný i z textboxů, další zase jen obrázek. Navíc, jak již bylo zmíněno, pro každý formulář vyžaduje REDCap jedinečné pojmenování polí. Jak je tohoto docíleno, bylo vysvětleno výše u elementu *date*. Proto v této části nastupují na řadu převodní tabulky. Prochází se dál element po elementu a hledá se, zda se hodnota atributu *name* neshoduje s 1. 3. nebo 5. řádkou v převodní tabulce. Toto jsou řádky, které postupně odpovídají jménům pro *in, out* a *mid*, které mají být nalezeny v XML dokumentu. Řádky 2, 4 a 6 jsou pak odpovídající názvy proměnných již v REDCapu. Proč byly tyto tabulky navrženy takto, bylo vysvětleno v kapitole Převodní tabulky.

Pokud se shoda nalezne, je ještě pomocí podmínek oddělena část pro checkboxy a ostatní typy elementů. Je to z toho důvodu, že checkboxy mají hodnotu uloženou v atributu *value*, kdežto například textboxy mají obsah jako hodnotu. Pro lepší představu může sloužit Ukázka kódu 5.2.

```
if(element.Attributes("name").Count()!= 0 &&
element.Attribute("name").Value == table[0, i])
{
if (element.Name=="checkbox"&&(element.Attribute("checked")!=null
  && element.Attribute("checked").Value == "true"
  && element.Attribute("value").Value != null))
        {
          var elName = table[1, i] + "_" + name;
          var elValue = element.Attribute("value").Value;
          dataIn.Add(elName.ToLower(), elValue);
        }
    if (element.Name != "checkbox"&& element.Value != "")
        {
          var elName = table[1, i] + "_" + name;
          var elValue = element.Value;
          dataIn.Add(elName.ToLower(), elValue);
         }
}
```

Ukázka kódu 5.2: Shoda s převodní tabulkou

Ukázka kódu výše ukazuje případ, kdy by se atribut *name* shodoval s 1. řádkem převodní tabulky. Pokud by se jednalo o checkbox, je ještě ověřeno, zda je vyplněn atribut *checked* a zda je true a zda je vyplněn atribut *value*. Bez těchto atributů by nebylo možné, aby program rozhodl o tom, která data má importovat a bez hodnoty by nebylo možné přiřadit checkbox určité odpovědi. Do dictionary se následně uloží jméno (odpovídající sloupec na řádku č.2) a získaná hodnota. Pokud by se jednalo o jiný element než checkbox, postup je obdobný.

Pokud jsou již všechny elementy prohledané, následuje import do REDCapu. Nejprve se importuje samostatně dotazník Pacient, jelikož v listu dohromady s jiným

dotazníkem, který by měl redcap_event_name rovný in1_arm_1 by import nefungoval. Následně je u dictionary ověřeno, že obsahují více než 3 klíče a přidá k nim ještě klíč department s hodnotou. Dictionary, které obsahují minimální počet klíčů, jsou uloženy do listu a v posledním kroku importovány do REDCapu pomocí funkce **ImportRecord()** třídy REDCapMethods.

Na úplný závěr ještě dojde k zapsání obsahu dvou listů (errors a problems) do odpovídajících souborů. Errors obsahuje název naskenovaného dotazníků a ID záznamu, pokud došlo k chybě během importu do REDCapu. Pokud došlo k chybě během importu, je pravděpodobné, že import vůbec neproběhl, proto by po otevření příslušného dotazníku pod uloženým ID, tento dotazník být neměl. K této chybě mohlo dojít například kvůli neplatné hodnotě pro pole (např. in1 může nabývat hodnot 0-4, ale v dictionary je uložena hodnota 5), neplatnému názvu pole v REDCapu, pokusu o import do neexistující události nebo do události, pro kterou není dotazník nadefinovaný, dále pokud nemá uživatel příslušná práva apod. Problémy s událostmi by měly být ošetřené v tomto programu, nicméně k importu neexistující hodnoty dojít může (viz. Testování).

Problems naopak obsahuje problémy, které se vyskytly během ukládání dat do dictionary. Tyto problémy jsou většinou způsobeny označením více checkboxů u jedné otázky hodnotou *checked=true*. Bohužel toto jsou chyby v XML souboru, se kterým si tento program neporadí. Pokud se na zdvojené true u checkboxu narazí, v dictionary zůstane první hodnota pro toto pole (ať už je správná nebo není), pouze se do problems.txt uloží jméno naskenovaného dotazníku a ID záznamu pro pozdější nalezení chyby. Chyby v XML souboru jsou dále uvedeny v kapitole Testování.

Tímto průchod metodou končí a do třídy Program je vrácen seznam (dictionary) aktuálních rodných čísel a ID.

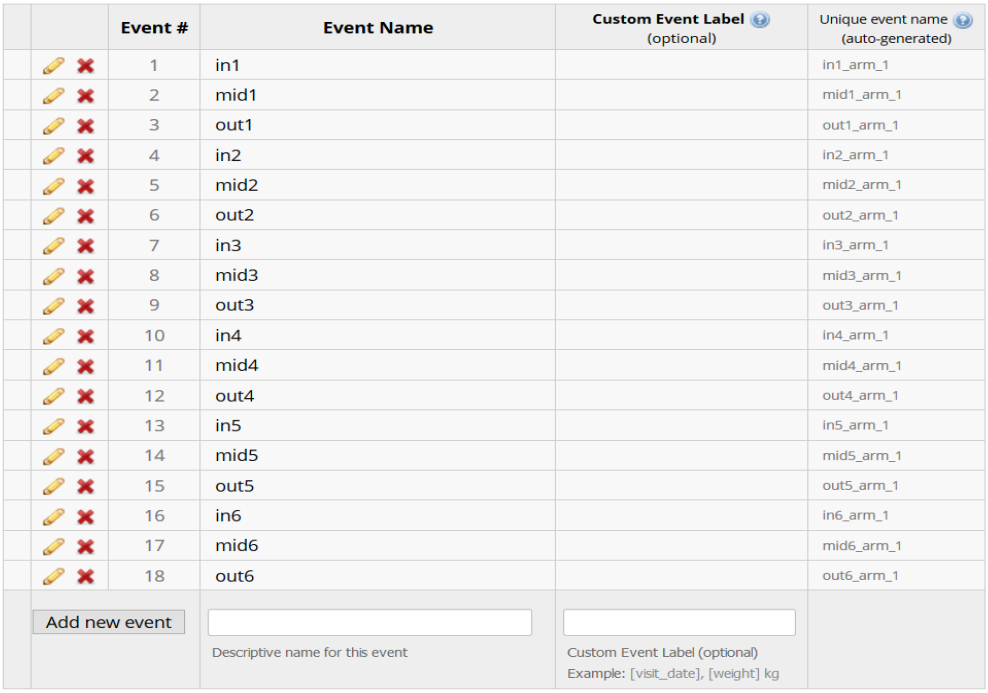
Aplikace není omezena žádnou licencí, aplikace i zdrojový kód mohou být šířeny, využívány a upravovány bez licenčních omezení, stejně tak nemusí být uváděno jméno autora.

6 Uživatelská dokumentace

Tato stručná dokumentace slouží jako návod na práci s aplikací a popisuje i některé vybrané úkony v REDCapu.

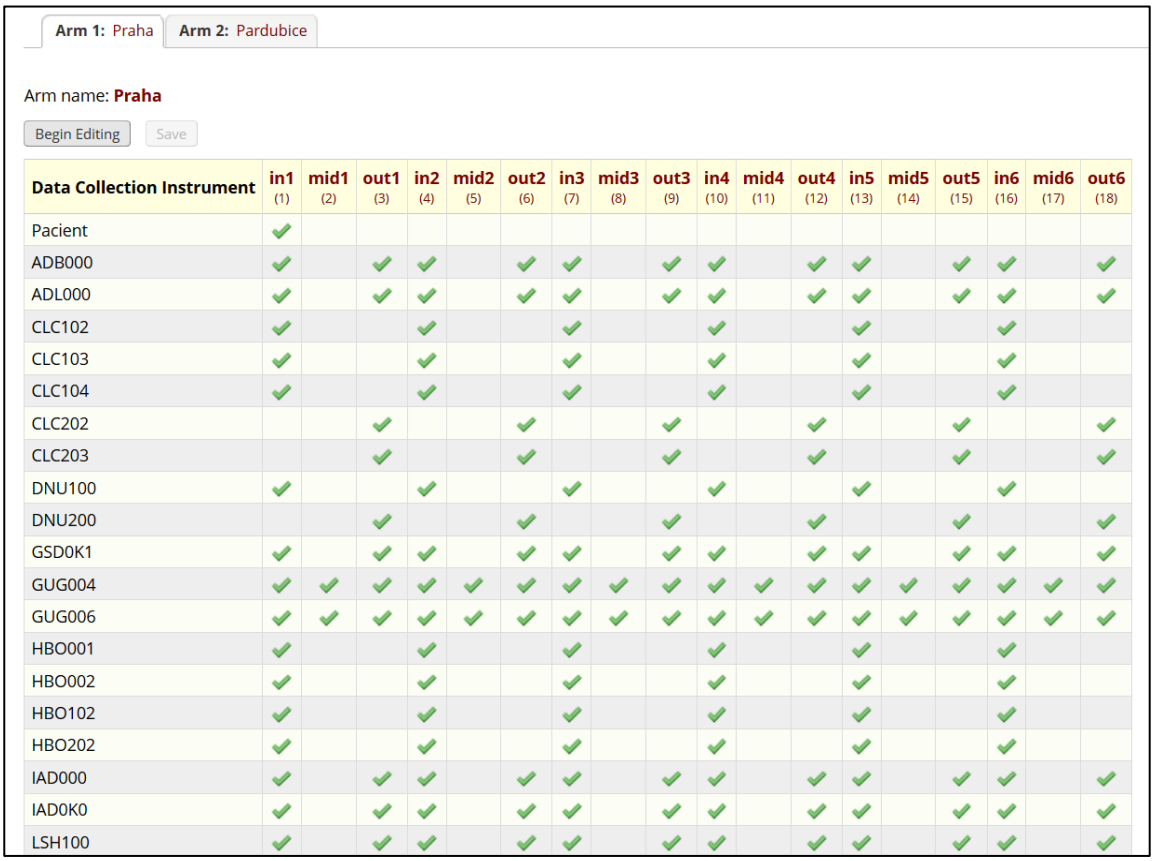
6.1 Přidání nové události

Může se stát, že bude potřeba v REDCapu vytvořit novou událost (např. u jednoho pacienta budeme potřebovat více než 8x vyplnit ten samý dotazník pro tu samou událost (*in*, *out* nebo *mid*). Vytvoření nové události je jednoduše možné po přihlášení do uživatelského účtu a následném vybrání příslušného projektu. Po otevření stránky s nastavením projektu je potřeba najít tlačítko *Define My Events*. Po kliknutí na tento odkaz budete přesměrování na další stránku, kde vidíte výpis aktuálně vytvořených událostí (Obrázek 6.1). Na poslední řádku je vždy možné další událost připsat. Vytvořit událost ale samo o sobě nestačí, je ještě potřeba pro tuto událost definovat dotazníky. To se dá opět udělat kliknutím na odkaz ze stránky nastavení projektu. Hned vedle výše zmiňovaného tlačítka je i tlačítko s názvem *Designate Instruments For My Events*. Po kliknutí budete přesměrováni na jinou stránku, kde vidíte přehled událostí a pro ně definovaných dotazníků (Obrázek 6.2). Po kliknutí na možnost *Begin Editing* budete moci zaškrtávat jednotlivé události pro dotazníky. Dotazník je pro danou událost definovaný, pokud je pole označeno zelenou značkou. Všechny úpravy je ještě potřeba potvrdit tlačítkem *Save*.



	Event #	Event Name	Custom Event Label (optional)	Unique event name (auto-generated)
	1	in1		in1_arm_1
	2	mid1		mid1_arm_1
	3	out1		out1_arm_1
	4	in2		in2_arm_1
	5	mid2		mid2_arm_1
	6	out2		out2_arm_1
	7	in3		in3_arm_1
	8	mid3		mid3_arm_1
	9	out3		out3_arm_1
	10	in4		in4_arm_1
	11	mid4		mid4_arm_1
	12	out4		out4_arm_1
	13	in5		in5_arm_1
	14	mid5		mid5_arm_1
	15	out5		out5_arm_1
	16	in6		in6_arm_1
	17	mid6		mid6_arm_1
	18	out6		out6_arm_1
Add new event		Descriptive name for this event	Custom Event Label (optional) Example: [visit_date], [weight] kg	

Obrázek 6.1: Tvorba nové události

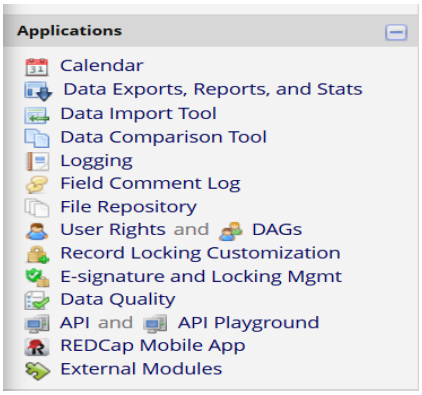


Data Collection Instrument	in1 (1)	mid1 (2)	out1 (3)	in2 (4)	mid2 (5)	out2 (6)	in3 (7)	mid3 (8)	out3 (9)	in4 (10)	mid4 (11)	out4 (12)	in5 (13)	mid5 (14)	out5 (15)	in6 (16)	mid6 (17)	out6 (18)
Pacient	✓																	
ADB000	✓		✓	✓		✓	✓		✓	✓		✓	✓		✓	✓		✓
ADL000	✓		✓	✓		✓	✓		✓	✓		✓	✓		✓	✓		✓
CLC102	✓			✓			✓			✓			✓			✓		
CLC103	✓			✓			✓			✓			✓			✓		
CLC104	✓			✓			✓			✓			✓			✓		
CLC202			✓			✓			✓			✓			✓			✓
CLC203			✓			✓			✓			✓			✓			✓
DNU100	✓			✓			✓			✓			✓			✓		
DNU200			✓			✓			✓			✓			✓			✓
GSD0K1	✓		✓	✓		✓	✓		✓	✓		✓	✓		✓	✓		✓
GUG004	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
GUG006	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
HBO001	✓			✓			✓			✓			✓			✓		
HBO002	✓			✓			✓			✓			✓			✓		
HBO102	✓			✓			✓			✓			✓			✓		
HBO202	✓			✓			✓			✓			✓			✓		
IAD000	✓		✓	✓		✓	✓		✓	✓		✓	✓		✓	✓		✓
IAD0K0	✓		✓	✓		✓	✓		✓	✓		✓	✓		✓	✓		✓
LSH100	✓		✓	✓		✓	✓		✓	✓		✓	✓		✓	✓		✓

Obrázek 6.2: Dotazníky definované pro jednotlivé události

6.2 Vygenerování tokenu

Pro následný import dat pomocí vytvořené aplikace je nejprve potřeba vygenerovat si token. Token si můžete jednoduše vygenerovat po přihlášení do uživatelského účtu. Vyberete projekt, do kterého budete data nahrávat, následně najdete v menu na levé straně záložku API (Obrázek 6.3). Po kliknutí vám na další stránce bude nabídnuta možnost token vygenerovat. Pokud tato možnost chybí, je pravděpodobné, že nemáte dostatečná práva a je potřeba obrátit se na administrátora.



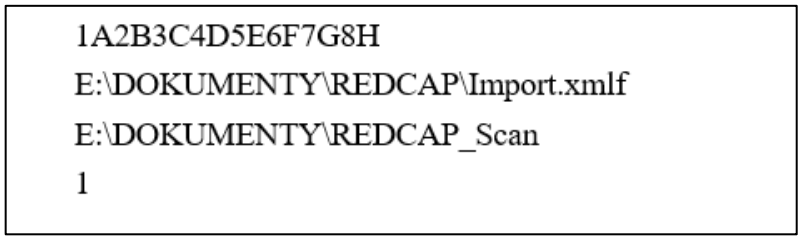
Obrázek 6.3: Levé menu se záložkou API

6.3 Práce s aplikací

Aplikaci je možné spustit kliknutím na exe soubor nazvaný REDCapImport. Následně budete vyzváni zapsání cesty k textovému souboru, který obsahuje potřebné údaje pro práci s REDCapem a nahrávání dotazníků. Jak tento soubor vytvořit bude popsáno v další podkapitole. Po nahrání souboru už aplikace sama začne pracovat se souborem a nahrávat data. Případné další pokyny budou vypsány do konzole. Nahrávání dat může trvat desítky minut, v závislosti na velikosti nahrávaného souboru (počtu dotazníků). Do doby, než bude vypsána hláška o dokončení přenosu dat, prosím program nevypínejte. Po ukončení aplikace je pravděpodobné, že u některých dotazníků nedošlo k uložení do REDCapu nebo je potřeba některé dotazníky zkontrolovat. Seznam dotazníků, které je potřeba zkontrolovat naleznete v textovém souboru nazvaném Problems.txt, který by měl být uložený ve složce, ve které jsou naskenované dotazníky. Seznam dotazníků, u kterých z nějakého důvodu nebylo možné uložení do REDCapu naleznete v souboru Errors.txt, taktéž uloženém ve složce s naskenovanými dotazníky. Tyto dva soubory je potřeba po dokončení nahrávání zkontrolovat a neuložené dotazníky přepsat ručně. Dotazníky, které nahrané byly, ale vyskytla se nějaká jiná chyba je potřeba v REDCapu zkontrolovat (nejčastěji se bude jednat o špatně zaškrtnuté pole). Kontrolu můžete provést manuálním porovnáním dat uloženým ve formuláři s naskenovaným dotazníkem (taktéž uložený v REDCapu). Formuláře, které vyžadují kontrolu poznáte v REDCapu podle žluté značky (u vyplněných formulářů svítí zelená značka).

6.4 Soubor s potřebnými daty

Příklad tohoto souboru je možné nalézt na přiloženém CD. Jedná se o klasický textový soubor, který je možné vytvořit pomocí např. poznámkového bloku nebo PSPadu. Jak má tento soubor vypadat ukazuje Obrázek 6.4. Na první řádce je uložen vygenerovaný token uživatele (jak získat token bylo popsáno výše), na další řádce se nachází cesta k souboru (včetně názvu souboru), ze kterého se budou importovat data do REDCapu. Další, třetí řádka, obsahuje cestu k adresáři, ve kterém se nachází naskenované dotazníky. Čtvrtá řádka obsahuje číslo pobočky, tedy 1 pokud jde o Prahu a 2 pokud jde o Pardubickou pobočku.



Obrázek 6.4: Dokument s informacemi pro aplikaci

7 Testování

Aplikace byla průběžně testována na vzorovém XML dokumentu, který obsahoval cca 2000 sesbíraných dotazníků. K testovacímu XML souboru byl odpovídající počet naskenovaných dotazníků. Během průběžného testování byla získávána důležitá zpětná vazba a průběžně opravovány chyby a dělány různé modifikace. Příkladem jedné z těchto úprav (momentálně již významné části programu) bylo vytvoření třídy PrepareXML, která zajišťuje úpravu XML dokumentu do takového tvaru, ze kterého je možné již bez větších potíží získat data pro import. Další úpravou provedenou po průběžném testování bylo vytvoření 6-řádkových převodních tabulek.

Testování v průběhu tvorby aplikace bylo prováděno pro jednotlivé formuláře zvlášť. Z testovacího XML souboru byl vybrán vždy vzorek cca 3 dotazníků odpovídajícího jména a následně byla aplikace spuštěna jen pro import těchto dotazníků. Toto testování bylo zavedeno z důvodu postupného přepisu nových dotazníků do REDCapu, zároveň kvůli ověření správného pojmenování proměnných v REDCapu a také pro testování nových úprav apod. Při prvních testováních byly zároveň zjištěny nepřesnosti v XML souboru, kdy například u některých otázek byly označeny 2 odpovědi jako vybrané, přesto že v papírovém dotazníku byla vybrána pouze jedna odpověď. Někde také chyběly přepsané texty, případně datum.

Zároveň v pozdějších fázích tvorby aplikace bylo prováděno i testování na celém XML souboru, pro všechny formuláře zároveň. Z tohoto testování vyplynul velký problém, a to v počtu vytvořených http spojení za minutu. Při překročení nastavené hranice dojde k zablokování přístupu z používané IP adresy (k čemuž také během vypracování došlo). Nicméně byly provedeny úpravy v programu (nahrávání celého dotazníku pro 1 pacienta naráz, prohledávání jen těch volných událostí, pro které je dotazník definován apod.), které mají tomuto jevu zabránit. Při dalších testováních již k zablokování IP nedocházelo a počet http spojení za minutu se pohyboval kolem poloviny přípustného množství http spojení za minutu. V REDCapu je taktéž možné manuálně nastavit hranici přípustných požadavků za minutu, proto je v případě potřeby možné při nahrávání záznamů tuto hranici zvýšit nebo případně na nezbytně nutnou dobu omezení zcela vypnout.

Závěrečné testování proběhlo jak pro jednotlivé formuláře zvlášť, tak především pro všechny formuláře zároveň a na velkém množství dat. Testování proběhlo dle očekávání, aplikace upravila soubor XML do vhodného tvaru, následně byla získána potřebná data, ta byla importována do REDCapu pod správná ID. Naskenované dotazníky byly taktéž importovány k odpovídajícím záznamům. Na vybraném vzorku importovaných dotazníků proběhla ruční kontrola podle dat v XML souboru a naskenovaných dotazníků.

8 Diskuse

V rámci této práce byl navržen systém ukládání dat z geriatrických vyšetření. Jako systém pro ukládání dat byl zvolen REDCap, který umožňuje zároveň i tvorbu webových dotazníků, sběr dat rovnou do webové aplikace a ukládání dat pomocí API.

Pro import dat byla vytvořena aplikace, která dokáže získat potřebná data z formátu XML a přenést je do REDCapu. Tato aplikace zahrnuje i úpravu XML souboru před importem, získání potřebných dat z XML, export potřebných dat z REDCapu (např. pro zjištění ID záznamu nebo volné události) a import záznamů i naskenovaných dotazníků do REDCapu. Přístup k datům přes REDCap může mít každý uživatel, kterému byla přidělena příslušná práva. K aplikaci byly také navrženy převodní tabulky, které obsahují názvy elementů nebo atributů, jejichž hodnoty je potřeba importovat a názvy polí v REDCapu. Tyto tabulky byly vytvořeny pro dotazníky dodané k této práci. Pokud by v budoucnu vznikly nové dotazníky, byla by potřeba tabulky dotvořit.

Během práce se jako problém ukázala málo obsáhlá dokumentace k využívané knihovně RedcapApi, jelikož při problémech s metodami nebylo někdy možné nalézt v dokumentaci řešení. Zároveň byly nalezeny některé problémy se souborem XML, které ale bylo možné v rámci aplikace ošetřit, případně poté ručně zkontrolovat.

9 Závěr

Cílem této bakalářské práce bylo navrhnout způsob ukládání dat z funkčních geriatrických vyšetření a vytvořit aplikaci pro migraci již sesbíraných dat do databáze. Jednotlivé cíle práce a stanovené požadavky na databázový systém i aplikaci pro migraci dat byly dosaženy za využití moderních technologií.

Hlavním záměrem této práce bylo umožnit přesun doposud nasbíraných dat z gerontologických vyšetření do jednotného úložiště (databáze), ke které bude snadný přístup a zároveň bude možné sbírat data i do online dotazníků.

Prvním hlavním cílem v této práci tak bylo navrhnout a uvést do provozu systém pro ukládání dat. Po pečlivém porovnání několika podobných systémů a zhodnocení splnění stanovených požadavků byl vybrán systém REDCap.

Pro další postup v práci bylo nutné vytvořit webové dotazníky v REDCapu a následně vybrat způsob importu. Po seznámení se se způsoby importu dat byl vybrán import dat pomocí API.

Druhý hlavní cíl práce byla tvorba aplikace, která bude umět importovat data z XML formátu přímo do REDCapu. Tato aplikace byla vytvořena v programovacím jazyce C# a následně testována na vzorovém souboru s daty. Práce s aplikací není pro uživatele nijak složitá, je potřeba pouze vytvořit jednoduchý textový soubor, zadat cestu k tomuto souboru a následně již aplikace pracuje bez uživatele.

V budoucnu je plánováno využít části aplikace ve webové aplikaci, která byla vytvořena v jedné z předcházejících pracích zabývajících se tímto problémem.

Aplikace není omezena žádnou licencí. Aplikace i zdrojový kód mohou být šířeny, využívány a upravovány bez licenčních omezení, stejně tak nemusí být uváděno jméno autora.

Seznam použité literatury

[1] TYGLEROVÁ, Terezie. Do poloviny století bude o polovinu více seniorů. *Statistika&my* [online]. 2019 [cit. 2019-03]. Dostupné z: http://www.statistikaamy.cz/2019/02/do-poloviny-stoleti-bude-o-polovinu-vice-senioru/

[2] WARD, MD, Katherine T a David B REUBEN, MD. Comprehensive geriatric assessment. In: *UpToDate* [online]. 2019 [cit. 2019-03]. Dostupné z: https://www.uptodate.com/contents/comprehensive-geriatric-assessment

[3] TOPINKOVÁ CSC., Prof. MUDr. Eva a Doc. MUDr. Jiří NEUWIRTH CSC. Funkční geriatrické vyšetření - komplexní pohled na starého člověka. *SANQUIS* [online]. 2002, **2002**(20) [cit. 2019-03]. Dostupné z: http://www.sanquis.cz/index1.php?linkID=art754

[4] ŠLAJCHRT, Petr. *Komplexní informační systém pro funkční vyšetření rizika pádů u pacientů ve vyšším věku*. Kladno, 2011. Diplomová práce. České vysoké učení technické v Praze, Fakulta biomedicínského inženýrství. Vedoucí práce Mgr. Radim Krupička.

[5] JANKŮ, Tomáš. *GDiag: Webová aplikace pro správu a export funkčních vyšetření*. Kladno, 2013. Bakalářská práce. České vysoké učení technické v Praze, Fakulta biomedicínského inženýrství. Vedoucí práce Mgr. Radim Krupička.

[6] ŠUSTEK, Pavel. *Webová aplikace pro digitalizaci papírových dotazníků*. Kladno, 2016. Bakalářská práce. České vysoké učení technické v Praze, Fakulta biomedicínského inženýrství. Vedoucí práce Mgr. Radim Krupička, Ph.D.

[7] XML Schema. *W3schools* [online]. [cit. 2019-03]. Dostupné z: https://www.w3schools.com/xml/xml_schema.asp

[8] About. *REDCap* [online]. [cit. 2019-03]. Dostupné z: https://projectredcap.org/about/

[9] HARRIS, PH.D., Paul A., Robert TAYLOR, M.A., Robert THIELKE, PH.D., Jonathon PAYNE, B.S., Nathaniel GONZALEZ, B.S.C.S. a Jose G. CONDE, M.D. Research Electronic Data Capture (REDCap) - A metadata-driven methodology and workflow process for providing translational research informatics support. *Journal of Biomedical Informatics* [online]. 2009, **2009**, 5 [cit. 2019-03]. DOI: 10.1016/j.jbi.2008.08.010. Dostupné z: https://www.sciencedirect.com/science/article/pii/S1532046408001226?via%3Dihub

[10] Popis framework REDCap. *Katedra biomedicínské informatiky* [online]. [cit. 2019-04]. Dostupné z: http://kbi.fbmi.cvut.cz/sites/default/files/REDCap%20dokumentace.pdf

[11] REDCap General Security Overview. *University of Washington* [online]. [cit. 2019-04]. Dostupné z: https://www.iths.org/wp-content/uploads/About-REDCap-Vanderbilt.pdf

[12] 20 Best Clinical Trial Management Software of 2019. *FinancesOnline* [online]. [cit. 2019-04]. Dostupné z: https://financesonline.com/clinical-trial-management/#castor

[13] OpenClinica Review. *FinancesOnline* [online]. [cit. 2019-04]. Dostupné z: https://reviews.financesonline.com/p/openclinica/

[14] Clinical Studio Review. *FinancesOnline* [online]. [cit. 2019-04]. Dostupné z: https://reviews.financesonline.com/p/clinical-studio/

[15] Security Statement. *Castor* [online]. [cit. 2019-04]. Dostupné z: https://www.castoredc.com/security-statement/

[16] OpenClinica user guide. *Optimistic* [online]. [cit. 2019-04]. Dostupné z: https://optimistic-dm.eu/images/com_projectfork/intranet2/user_guide.pdf

[17] Electronic Data Capture System Overview. *ClinicalStudio* [online]. [cit. 2019-04]. Dostupné z: http://www.clinicalstudio.com/overview/

Příloha A: Obsah přiloženého CD

- klíčová slova v českém i anglickém jazyce
- abstrakt v českém jazyce
- abstrakt anglickém jazyce
- naskenované zadání bakalářské práce
- kompletní bakalářská práce
- aplikace (REDCapImport.exe)
- zdrojový kód aplikace pro import dat
- testovací data
- převodní tabulky
- vzorový soubor s daty pro aplikaci