ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta biomedicínského inženýrství
Katedra biomedicínské informatiky

Webová aplikace pro digitalizaci papírových dotazníků

Web application for paper questionnaire digitization

Bakalářská práce

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

Vedoucí práce: Mgr. Radim Krupička, Ph.D.

Pavel Šustek

Kladno, květen 2016

České vysoké učení technické v Praze, Fakulta biomedicínského inženýrství

Katedra biomedicínské informatiky

Akademický rok: 2015/2016

Zadání bakalářské práce

Student:	**Pavel Šustek**
Obor:	Biomedicínská informatika
Téma:	**Webová aplikace pro digitalizaci papírových dotazníků**
Téma anglicky:	Web Application for Paper Questionnaire Digitization

Zásady pro vypracování:

Cílem bakalářské práce je vytvořit webovou aplikaci, která umožní digitalizovat dotazníky z funkčních vyšetření. Aplikace zpracuje naskenované dotazníky, ve kterých jsou pozice jednotlivých prvků definované pomocí XML šablony. Dotazníky a XML šablony jsou vytvořené GDiag editorem vyvinutém na KBI. Aplikace extrahovaná data importuje do databáze systému GDiag. Při implementaci se inspirujte již vytvořeným desktopovým programem pro digitalizaci dotazníků a kladťe důraz na snadné a intuitivní ovládání. Funkčnost ověřte zpracováním již naskenovaných funkčních vyšetření.

Seznam odborné literatury:

[1] Christian Nagel, Bill Evjen, Jay Glynn, C# 2008 programuje profesiálně, ed. 1, 2009, Computer Press, 978-80-251-2401-7
[2] Mary Delamater, Murach's ASP.NET 4.5 web programming with C# 2012, ed. 5., New York: Mike Murach & Associates, 2012, ISBN 978-1-890774-75-2
[3] Tomáš Janků, GDiag Webová aplikace pro správu a export funkčních vyšetření , 2013

zadání platné do: 30.09.2017

Vedoucí: Mgr. Radim Krupička, Ph.D.

.....
vedoucí katedry / pracoviště

.....
děkan

V Kladně dne 22.02.2016

Prohlášení

Prohlašuji, že jsem bakalářskou práci s názvem Webová aplikace pro digitalizaci papírových dotazníků vypracoval samostatně a použil k tomu úplný výčet citací použitých pramenů, které uvádím v seznamu přiloženém k bakalářské 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).

V ……. dne …… …….

Podpis

Poděkování

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

Abstrakt

Posuzování zdravotního stavu pacientů v gerontologii je prováděno pomocí funkčních vyšetření. Jedná se o řadu specializovaných testů, jejichž výsledky jsou zaznamenávány do papírových formulářů. Tato práce se zabývá tvorbou webového programu pro převod dat z funkčních vyšetření do digitální formy a jejich následný export. Formuláře jsou přes FTP odesílány ze scanneru v lokálních zařízeních přímo na centrální server aplikace. Zpracovaná data jsou ukládána do předem vytvořených XML šablon. Celý proces urychlují automatické nástroje pro rozpoznávání zaškrtnutých polí, přiřazení šablony dle čárového kódu a předvyplňování naposledy vložených hodnot. Cílem této práce je modernizovat postupy zpracování funkčních vyšetření. Jelikož se jedná o program určený pro zdravotnický personál, je kladen důraz na přehledné a intuitivní ovládání. Samotná aplikace je vyvíjena v programovacím jazyce C# s využitím frameworku ASP.NET.

Klíčová slova

Digitalizace dat, webová aplikace, funkční vyšetření, informační systém

Abstract

Functional examination in gerontology is performed using standardized paper questionnaires. The main target of this work is to create an internet application for digitization data from scanned questionnaires and to export them. The scanners in local facilities send the questionnaires via FTP directly to the application server. Data is stored into predefined XML templates. The whole process is accelerated with automated tools for optical mark recognition, barcode reading and remembering last entered values. The software is ment to be used by medical staff. Therefore it is necessary to design effortless and user-friendly environment. Application itself was done using latest technologies and it is implemented in C# programming language with use of an ASP.NET framework.

Key words

Data digitization, web application, functional examination, information system

Obsah

1 Úvod

GDiag je rozsáhlý projekt, který má za úkol pomoci modernizovat postupy při práci s funkčními vyšetřeními v gerontologii a to od samotné tvorby konkrétního formuláře, přes digitalizaci vyplněných dat, až po archivaci na webovém uložišti. Funkční vyšetření jsou prováděna pomocí specializovaných papírových dotazníků.

Cílem této práce je vytvořit webovou aplikaci pro snadný převod vyplněných dat z papírové do elektronické podoby a jejich následný export do XML šablon. Tento webový scanner by měl být součástí celého projektu GDiag, ale také použitelný samostatně. Aplikace má poskytovat nástroje pro práci s ofocenými dotazníky a několik funkcí pro co nejrychlejší digitalizaci dat, jako je například automatizované čtení zaškrtnutých polí. Program bude využíván přímo gerontologickými centry a měl byt tedy být vytvářen s ohledem na cílovou skupinu uživatelů, kterou je zdravotnický personál. Proto je třeba implementovat co nejjednodušší a nejintuitivnější uživatelské rozhraní. Tato práce je inspirována již existující desktopovou aplikací FormScan. Z hlediska udržitelnosti a dostupnosti programu bylo však výhodné zvolit webové řešení, které bude dostupné z jakéhokoliv počítače s připojením k internetu a v případě modernizací nebo úprav, nebude potřeba přistupovat ke každému počítači zvlášť.

Hotové řešení je nahráno na server centrální aplikace GDiag pro správu a export. Konkrétně na webovou adresu http://gdiag.fbmi.cvut.cz/scan/, kde je zároveň testováno. Během období testování je volně dostupné bez potřeby autentifikace.

Bakalářská práce je rozdělená do sedmi částí. Výše v kapitole Úvod je nastíněna podstata aplikace a uvedeny cíle této bakalářské práce. V části Analýza současného stavu a motivace je analyzován aktuální stav řešení dané problematiky, vysvětleny základní pojmy z gerontologie a popsány programy a metody, které se v současné době používají. V kapitole Návrh a popis řešení jsou důkladně rozebrány jednotlivé požadavky na program, dále potom uvedeny použité technologie a nástroje pro vývoj a v závěru specifikována funkčnost aplikace. Čtvrtá část, Uživatelská dokumentace, popisuje fungování programu z pohledu uživatele. Měla by sloužit jako příručka k použití. Obsahuje návod k obsluze všech funkčností aplikace a postupně provádí celým procesem

digitalizace ofoceného dotazníku. Implementace detailněji vysvětluje provedení hlavních funkčností. Jsou zde stručně popsány jednotlivé operace s dotazníkem a činnost programu například při automatickém vyplňování dat, exportu nebo zobrazování formulářů. Veškeré zmíněné metody a techniky programování jsou spolu s celým kódem programu k dispozici na přiloženém DVD. V Diskuzi je polemizováno nad širším využitím aplikace a výhledy do budoucna. Závěr popisuje výsledky práce a splnění cílů,

2 Analýza současného stavu a motivace

V této kapitole je stručně popsána oblast, ve které se aplikace bude používat, bude objasněn význam této bakalářské práce, vysvětlena aktuální situace v oblasti a analyzován současný stav řešení podobné problematiky.

O současné době je možné s jistotou říci, že společnost zažívá významný technologický rozvoj. Tento pokrok techniky se dotýká života většiny lidí, přináší s sebou nové problémy, ale také nová řešení a metody, které se postupně dostávají do různých odvětví a mnohdy významně usnadňují nebo urychlují práci a efektivitu v dané oblasti. Vývoj a integrace technologií neminula ani zdravotnictví. Díky moderním přístrojům je nyní možné běžně provádět zákroky, které kdysi zvládali pouze výjimeční a zkušení lékaři nebo určovat diagnózu s mnohem větší přesností. Rozvoj se však netýká pouze modernizace diagnostických či operačních přístrojů, ale zasahuje i do organizace procesů ve všech oborech zdravotnictví. Tato bakalářská práce má za cíl zmodernizovat a zjednodušit proces digitalizace dat z geriatrických funkčních vyšetření.

2.1 Gerontologie a funkční vyšetření

Gerontologie je nauka o stárnutí a stáří. Zabývá se třemi hlavními směry, experimentálním, tento obor zkoumá důvody a sbírá poznatky o tom, proč živé organismy stárnou a možnosti s tím spojené. Další oblastí gerontologie je sociální, která studuje vztah starého člověka a společnosti například v psychologických nebo sociologických, ale i dalších aspektech. Třetím a z hlediska této práce nejdůležitějším směrem, je gerontologie klinická neboli geriatrie.

Geriatrie studuje zdravotní a funkční stav ve stáří, zvláštní nemoci s ním spojené, jak tyto choroby diagnostikovat, ale také jak staré lidi ošetřovat. Zvláštní pozornost je věnována stařecké multimorbiditě, křehkosti a geriatrickým syndromům, jako jsou demence, hypomobilita, imobilita, anorexie, kognitivní deficit nebo malnutrice a další. Tento medicínský obor se také zabývá dlouhodobou péčí o disabilní pacienty. [1] [2]

Nezbytnou součástí geriatrie jsou funkční vyšetření. Umožňují specifikovat zhoršování stavu, stanovit léčebný plán nebo odhalení dosud nediagnostikovaných poruch. Slouží také k posouzení soběstačnosti pacienta. Testy se provádějí opakovaně a tím zaručují informace o účinnosti léčby a průběhu onemocnění. Proto poskytují velice cenná data pro výzkum a napomáhají zlepšení celkové poskytované péče. Dělí se na tři typy. Významným nástrojem k hodnocení stavu výživy je test MNA (Mini Nutritional Assessment). K hodnocení bolesti jsou využívány analogové škály, kdy je pacient dotazován na bolest a v případě kognitivních poruch potom škály PAINAD nebo MOBID.

Biomedicínská vyšetření, mají za úkol získávat informace o funkčním stavu pacientů s konkrétní chorobou, monitorovat vyskytující se poruchy smyslů, fyzickou zdatnost nebo výživu. Příklad vyšetření hodnotícího mobilitu je test „Get up and go“. Pacient je požádán, aby vstal, ušel 3 metry a vrátil se zpět na židli. Může se hodnotit podle potřeného času nebo jsou bodovány úkony jako postavení, posazení, chůze, otočení.

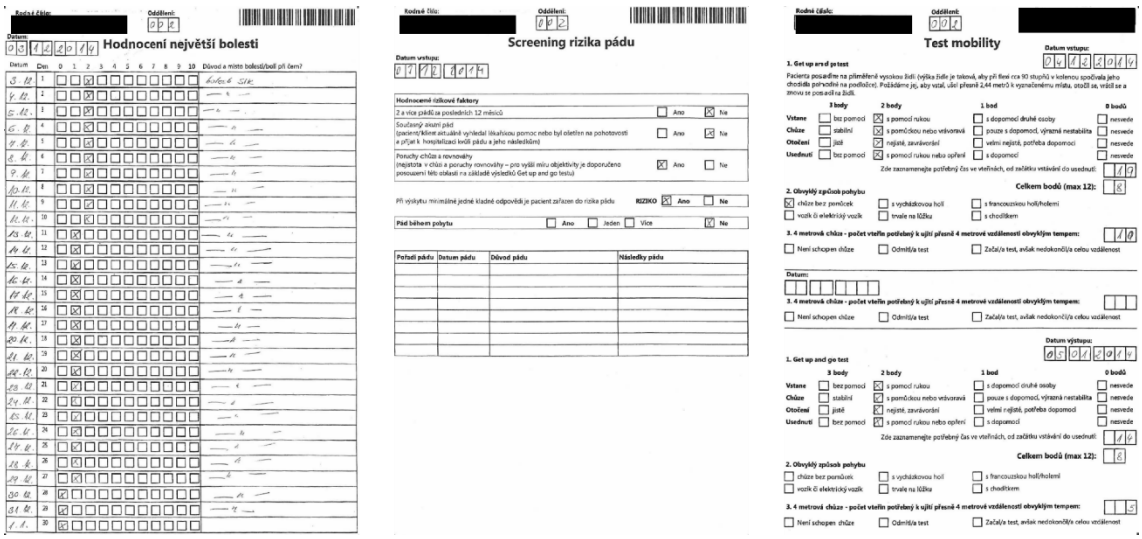
Další oblastí získávaných dat je psychická, která sleduje přítomnost kognitivních poruch, deprese a další psychiatrické poruchy. Mezi kognitivní vyšetření patří například MMSE (Mini-Mental-State Examination), jedná se o test zaměřený na jednotlivé kognitivní domény s maximem 30 bodů. 26 bodů již představuje patologický nález. Dalším testem je například MoCA – Montrealský test. Je velmi podobný, ale o trochu citlivější. K orientačnímu vyšetření může sloužit také test hodin, jedná se o nenáročné vyšetření, kdy je pacient požádán, aby nakreslil ciferník a vyznačil polohu ručiček v určitém čase. Vyšetření dokáže velmi citlivě odhalit přítomnost kognitivních poruch a na jeho základě mohou být doporučena další vyšetření. K vyšetření depresivity se používá například vyšetření GDS (Geriatrická škála deprese podle Yesavage). Zahrnuje 15 otázek, v případě, že pacient dosáhne 11 bodů, může se jednat i o klinickou až těžkou depresi.

Třetím typem jsou dotazníky pro sociálně ekonomickou oblast ke zkoumání kvality mezilidských vztahů, zapojení do společenských aktivit nebo i kvalitu bydlení a ekonomickou situaci. [2] [3]

2.2 Zpracování dotazníků

Funkční vyšetření jsou vysoce specializované testy, jejichž výsledky jsou zaznamenávány do předtištěných papírových dotazníků. V současné době jsou tyto výsledky po zhodnocení ošetřujícím personálem většinou uchovávány v archivech v papírové podobě a v případě potřeby je celý dotazník ofocen a odeslán k výzkumu, kde jsou data ručně přepisována do digitální formy. Tento zavedený systém má za následek horší přístup k získaným informacím pro jejich vyhodnocování a také nedostatečnou dostupnost pro výzkum.

Příklad několika gerontologických funkčních vyšetření je zobrazen na Obrázek 1 - Ukázka funkčních vyšetření. Zleva Hodnocení největší bolesti, Screening rizika pádu a Test mobility.



Obrázek 1 - Ukázka funkčních vyšetření

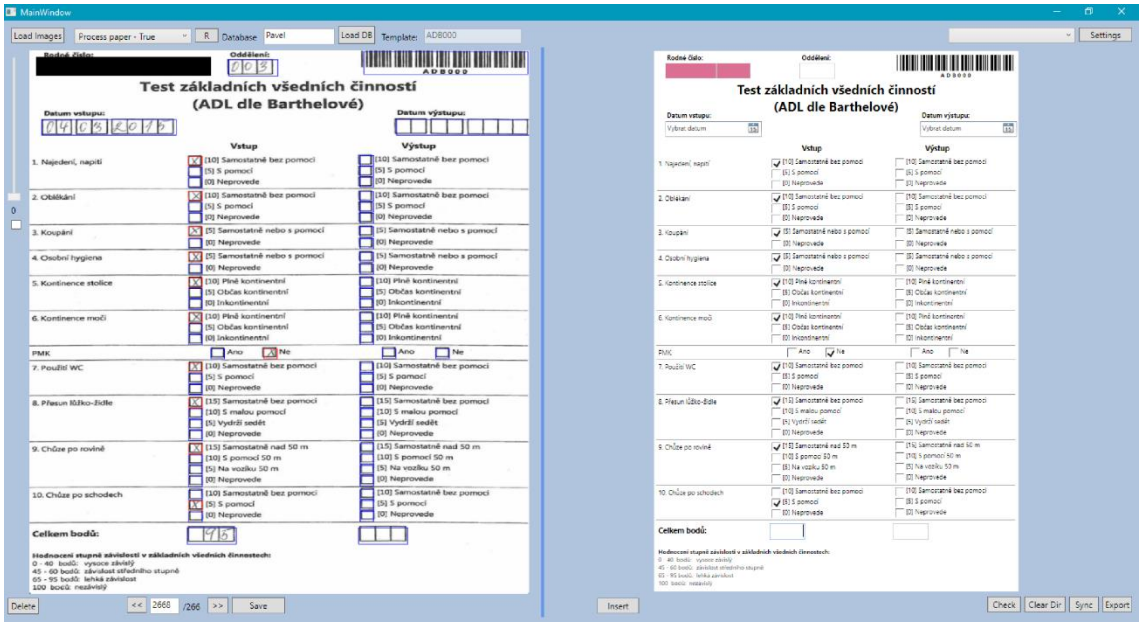
2.2.1 FormScan

Několik gerontologických center v České Republice od roku 2012 využívá modernější systém, kterým je již existující aplikace pro digitalizaci dotazníků - FormScan. Tento software vznikl na Fakultě biomedicínského inženýrství Českého vysokého učení technického v Praze. A byl navržen jako desktopová aplikace pro

operační systém Windows. Spolu s ním byl vyvinut nástroj pro tvorbu specifických šablon, které defiují prvky formuláře a do nichž jsou zároveň získaná data ukládána, FormEdit. Nabízí řadu prvků, které se nejčastěji vyskytují v geriatrických dotaznících, ale i v dotaznících obecně. Umožňuje tím tedy přehlednou a rychlou tvorbu formuláře. Výstupem aplikace není pouze dotazník připravený k vytisknutí a podání pacientovi, ale také jeho kopie v XML formátu s přesně definovanými prvky, do které jsou později data z vyplněného formuláře pomocí FormScanu zkopírována.

Desktopový scanner FormScan nabízí důležité funkce pro zpracování funkčních vyšetření, jako je úprava barev a kontrastu původního snímku pro lepší čitelnost dat, ořezání podle předem definovaných rohů a hlavně automatické čtení zaškrtnutých polí pomocí implementovaného OMR algoritmu, čímž za uživatele odvádí monotónní práci náchylnou k chybě. Dále kopírování výsledků do XML šablony a jejich následný export na disk. Všechna pole jsou označena barvou, podle toho, zda byla zaškrtnuta nebo ne a tím je uživateli poskytována možnost zkontrolovat, zda nedošlo k chybě. Nechává také prostor pro doplnění dalších údajů z formuláře. Během toho FormScan validuje, zda byly do XML předlohy vyplněny klíčové vstupy a jestli mají správný tvar, například rodné číslo nebo datum.

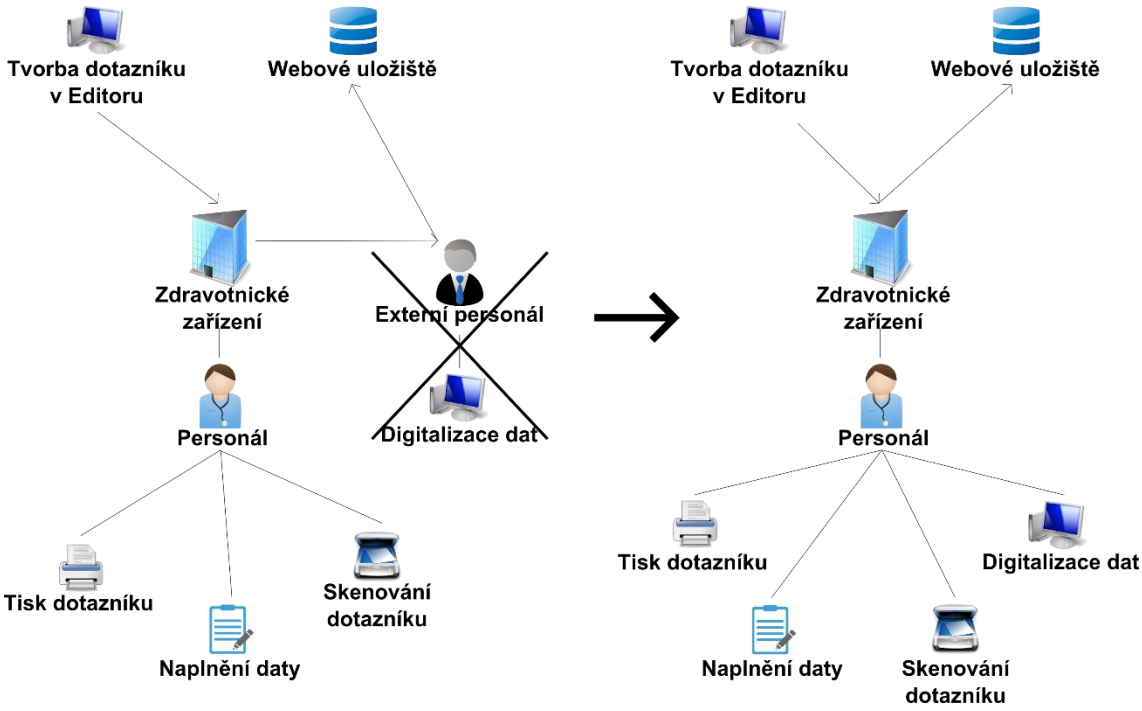
Nicméně obsluha tohoto programu není dostatečně intuitivní a je poměrně složitá. Vyžaduje určitou míru technických znalostí, což se ukázalo jako komplikace pro zdravotnický personál. Také je problém s aktualizací systému a s jejich instalací na počítačích v gerontocentrech. Celá procedura zpracování tedy nese řadu problémů. Oskenované formuláře jsou proto v současné době odesílány zpět ke zpracování, kde je pomocí FormScanu digitalizují proškolené pověřené osoby. Získaná data se potom ručně nahrávají na webové uložiště GDiag, kde jsou k dispozici pro další úpravy a vyhodnocování. Náhled prostředí aplikace je ukázán na Obrázek 2.



Obrázek 2 - Pracovní plocha programu FormScan po načtení dat

Z těchto důvodů byl vznesen požadavek Gerontologickým centrem v Praze 8, které spolupracuje s Českou alzheimerovskou společností, Obvodním ústavem sociálně zdravotnických služeb a řadou domovů pro seniory, na tvorbu intuitivnější a co nejjednodušší aplikace, kterou by mohl samostatně využívat zdravotnický personál. FormScan by tedy měl být nahrazen právě GDiag Webovým scannerem, který je vyvíjen s ohledem na co nejjednodušší ovládání a snadnou integraci do procesu zpracování funkčních vyšetření a byl navržen tak, aby splňoval všechny funkční i nefunkční požadavky. Požadavky jsou analyzovány v kapitole 3.1.

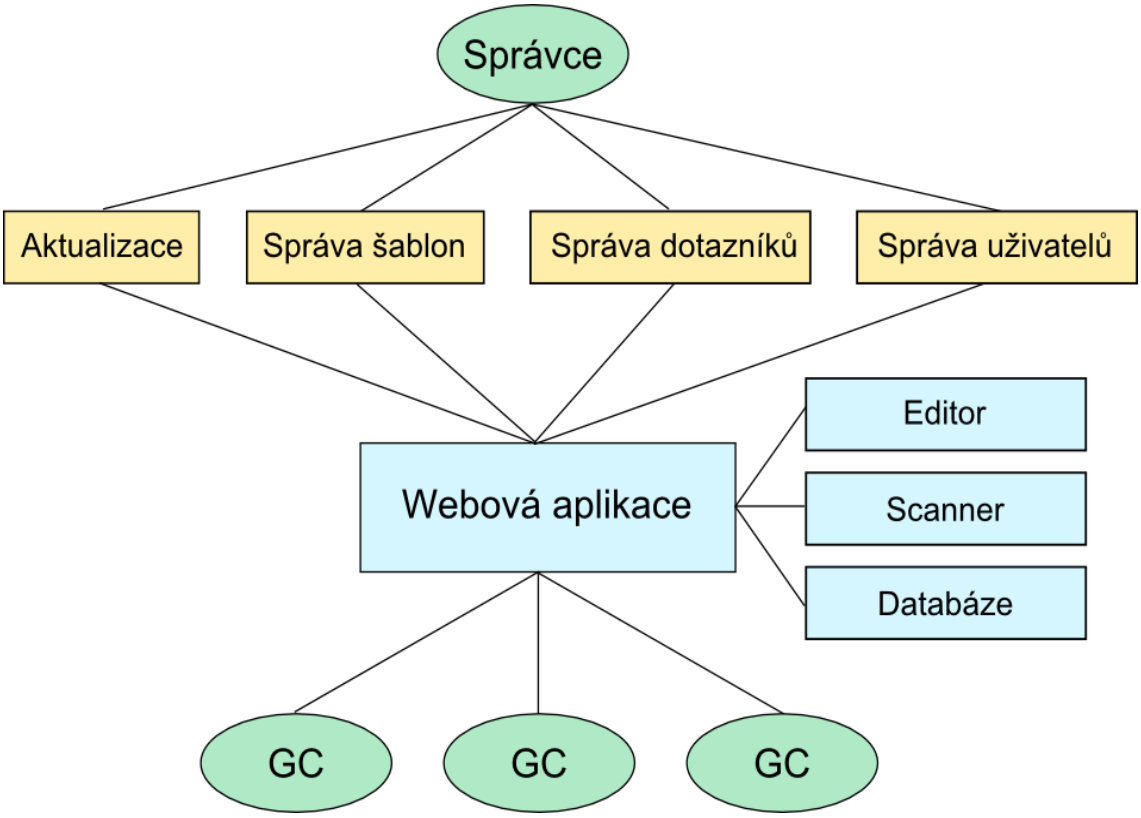
Výhoda přechodu na webovou verzi skeneru, využívanou přímo zdravotnickým personálem je znázorněna na Obrázek 3.



Obrázek 3 - Přechod na digitalizaci dotazníků zdravotnickým personálem

2.2.2 GDiag webová aplikace

Tento rozsáhlý software využívá moderních technologií a jejím cílem je vytvořit kompletní zázemí pro zpracování geriatrických funkčních vyšetření od jejich vzniku a návrhu, až po konečné archivování tak, aby vše bylo dostupné na jednom místě. GDiag původně tvořila webová část pro správu a export a desktopové programy FormScan a FormEdit. Myšlenkou projektu je však využít nové možnosti webových vývojových prostředků a i obě tyto desktopové části převést do internetové formy tak, aby bylo vše přístupnější a na jednom místě, jak je znázorněno na Obrázek 4. GDiag se tím stává kompletně webovou aplikací a je dostupný z jakéhokoliv počítače s připojením k internetu, čímž se odbourává nutnost instalovat jej na každou stanici a přibývá možnost provádět globální aktualizace nebo vkládat nové šablony do jedné databáze, aniž by bylo třeba řešit nové předlohy u každého konkrétního počítače zvlášť.



Obrázek 4 - Schéma webového GDiagu, poznámka: GC = gerontocentrum

Komponenty softwaru

Webová aplikace GDiag se skládá z několika větších modulů, všechny jsou internetové a každý je určen ke konkrétnímu procesu s funkčním vyšetřením. Kromě samotného provedení testu a vyplňování dotazníku tedy pokrývá všechny úkony se zpracováním formuláře spojené.

Webový Editor

Tento modul vychází z desktopové aplikace FormEdit. Jeho cílem je umožnit co nejsnadnější tvorbu formuláře pro konkrétní funkční vyšetření. Disponuje širokou škálou nastavení a nástrojů, například přesné definování rohů formuláře nebo pomocná mřížka pro dokonalé rozvržení prvků. Nabízí hotová textová pole, zaškrtávací pole, speciální místa pro obrázky, čárové kódy, rodná čísla nebo datumy a další. Po vytvoření je dotazník vytištěn a zároveň exportován do XML šablony, která se poté plní digitalizovanými daty. Vytištěný formulář je vyplňován při vyšetření.

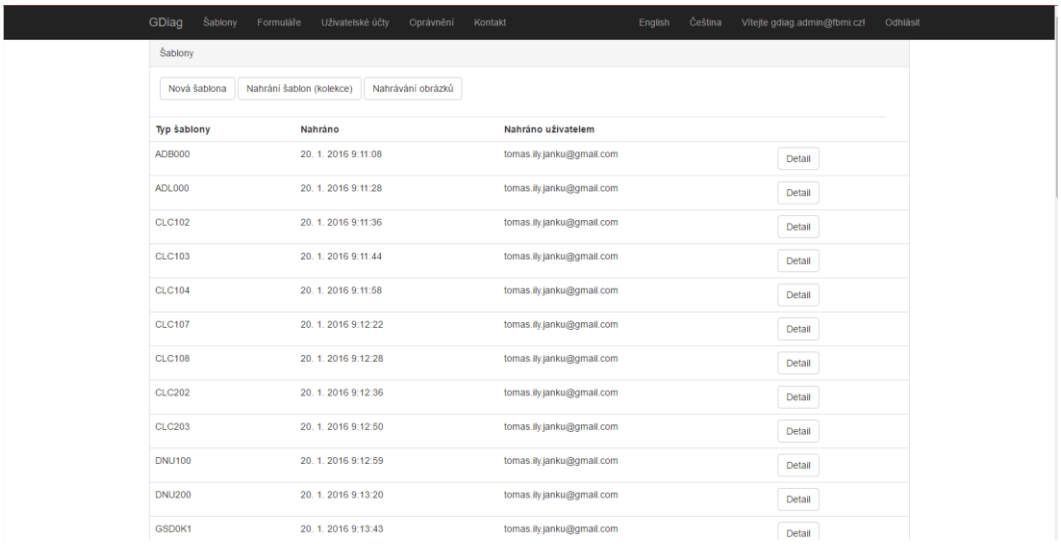
Webový Scanner

Po provedení testů je potřeba získaná data převést do digitální formy a následně odeslat do aplikace pro správu a export. K digitalizaci dat slouží právě webový scanner, který je výstupem této práce. Konkrétní požadavky na funkčnost a bližší specifikace jsou uvedeny v kapitole 3.1. [3]

Webová aplikace pro správu a export

Modul pro správu a export je centrální částí aplikace s databází, do které jsou ukládána digitalizovaná funkční vyšetření. Celý modul je vyvinut ve webové nástavbě Microsoft .NET frameworku ASP.NET, který umožňuje programovat v objektově orientovaném jazyce C# a využívat rozsáhlé knihovny .NET. Využívá architekturu MVC 4. Tyto technologie jsou používány i v této práci, bližší specifikace jsou uvedeny v kapitole 3.2. Náhled do prostředí aplikace zobrazuje Obrázek 5.

Pro vstup do aplikace je třeba mít registrovaný uživatelský účet a je tedy dostupná pouze autorizovaným uživatelům. V roli běžného uživatele se nabízí možnosti práce s vyšetřeními, vyhledávání ve všech hotových dotaznících, jejich správa a procházení, vyhodnocování nebo úprava v nich uložených dat a jejich export nebo tisk. Správce systému může upravovat a přidávat uživatelské účty, nahrávat nové šablony nebo editovat stávající a provádět další funkce spojené se spravováním aplikace. [4]



Obrázek 5 - Webová aplikace pro správu a export (stránka správy šablon)

2.3 Podobné aplikace

Optical mark recognition je proces, který rozpoznává ručně zaškrtnutá pole na předem definovaných polích. Protože výstup této práce je OMR software, použitelný kromě začlenění v projektu GDiag i samostatně, je vhodné analyzovat současnou situaci v oblasti aplikací zpracovávajících data z oskenovaných formulářů do digitální podoby.

Kromě FormScanu, který byl popsán výše, existuje programů umožňující OMR ještě mnoho, zde je výběr těch nejpoužívanějších.

- **Remark OMR**

  – společnost Remark vyvíjí OMR software již od roku 1991 a je pravděpodobně nejrozšířenější na trhu. Kromě digitalizace dat umožňuje i tvorbu dotazníku a statistické výpočty z výsledků, také podporuje rozeznávání čárových kódů. Není konkrétně zaměřený na určitou oblast a data dovede exportovat do obdivuhodných 35 formátů. Největší nevýhody jsou, že je to desktopová aplikace pouze pro operační systém Windows, díky mnoha funkcím je velmi složitý na ovládání a pro Českou Republiku není možné si přímo pořídit licenci, tudíž není v českém jazyce. [5]

- **TestsChecker**

  – nejpoužívanější rozpoznávací software v ČR již od roku 1998, používá jej přes 100 středních a vysokých škol. Méně se používá se také v lékařství. Podporuje čtení čárových kódů a upozorňuje na sporně vyplněné testy. Náročnost ovládání je středně obtížná. Je však vytvořen původně pouze pro čtení polí a tedy neumožňuje doplňovat do šablon vlastní text v průběhu kopírování dat. Další nevýhodou je, že se jedná o desktopový program a licence se uděluje na konkrétní počítač. [6]

- **FormScanner**

  – tento desktopový program je určen pro operační systémy Windows a Linux. Velkou výhodou je, že je zdarma. Poskytuje také základní tvorbu dotazníku. Slouží však spíše pro vyhodnocení správnosti předem navolených odpovědí a export výsledků ve formátu CSV. [7]

3 Návrh a popis řešení

Tato kapitola se věnuje návrhu a popisu řešení aplikace. Úvodem jsou analyzovány požadavky na vyvíjený software a dále odůvodněny a popsány zvolené vývojové technologie.

3.1 Analýza požadavků

Sběr a analýza požadavků je jednou z nejdůležitějších částí procesu vývoje jakéhokoliv informačního systému. Je klíčové stanovit již na samotném začátku, jaké funkčnosti by měla aplikace splňovat. Podle toho jsou totiž voleny vývojové nástroje, určena časová a prostředková náročnost a vytvořen celý návrh aplikace. Vývoj je tedy plně přizpůsoben daným cílům a v případě špatně zadaných nebo pozdě vyřčených požadavků může být již nesmírně náročné je implementovat. [4]

Hlavní požadavky byly vysloveny koncovým zákazníkem, Gerontologickým centrem Praha 8 a následně přetlumočeny vedoucím této bakalářské práce. Na jejich základě potom byly diskutovány další požadavky a cíle a zvoleny vhodné metody vývoje.

Primárním požadavkem bylo implementovat funkčnosti již existujícího desktopového programu a navrhnout systém tak, aby byl software snadno ovladatelný zdravotnickým personálem a tedy vytvořit co nejjednodušší prostředí pro uživatele s intuitivními ovládacími prvky. Dalším cílem bylo umožnit rychlý export dat z aplikace. Důležitým požadavkem ovlivňujícím celý vývoj a návrh aplikace, bylo využít nové moderní vývojové nástroje a učinit software webový, protože by program v budoucnu měl být využíván více zdravotnickými zařízeními a internetové řešení nabízí lepší dostupnost uživatelům a zvýšenou schopnost udržování programu.

3.1.1 Funkční požadavky

Funkční požadavky popisují a specifikují požadované funkce programu a jeho chování v určitých situacích.

Možnost práce online

Pro běh aplikace online, bylo třeba zvolit vhodné technologie. Program byl vyvíjen v nadstavbě .NET frameworku ASP.NET na architektuře MVC a běží na serveru IIS. Více informací je uvedeno v kapitole 3.2.

Příjem ofocených dotazníků ze scanneru

Nejvýhodnější a nejjednodušší metoda pro nahrávání vyplněných dotazníků ze zdravotnických zařízení do aplikace je odesílání přímo ze scanneru. Tento způsob je navíc výhodný díky své jednoduchosti a tedy není třeba učit zdravotnický personal nic nového. Formuláře jsou odeslány protokolem FTP na server aplikace do příslušné složky, kde jsou uloženy ve formátu TIF jako soubor snímků z konkrétního vyšetření a odtud jsou dále zpracovávány.

Zobrazení přijatých dotazníků

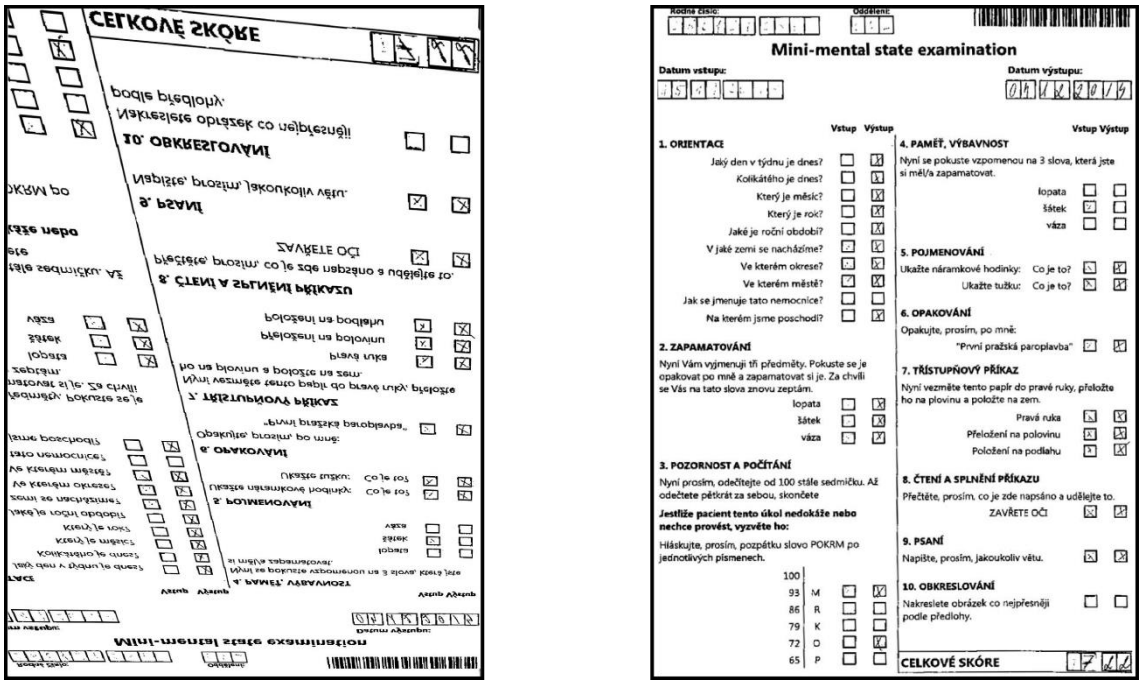
Je potřeba mít možnost přijaté dotazníky procházet a vybrat konkrétní ke zpracování. Proto ve výpisu ze složky s dotazníky musí být možnost rychlého náhledu konkrétního snímku. Při každém spouštění aplikace se proto zpracovávají všechny nově příchozí soubory a probíhá konvertování balíčků z formátu TIF do jednotlivých snímků ve formátu JPEG, vždy se společným názvem, ve kterém je uvedeno i pořadí v konkrétním balíku. Ve složce na serveru tedy zůstanou pouze JPEG soubory. Také se vytváří jejich zmenšené náhledy. Ty jsou potom zobrazovány uživateli při najetí myší na konkrétní dotazník. Více k implementaci této funkčnosti je uvedeno v kapitole 5.1.

Ořezání ofoceného snímku podle rohů

Ofocený snímek je nutno správně ořezat podle definovaných rohů, označených jako čtverce na okraji formuláře. OMR pro rozpoznávání zaškrtnutých polí totiž funguje podle přesně daných souřadnic, které jsou zapsány v XML šabloně a určeny vzhledem k levému hornímu rohu formuláře uvnitř ohraničení. Nepočítá tedy s okraji papíru, navíc snímek může být naskenován nakřivo a ořezáním podle definovaných rohů se tedy vyrovná. Tato operace je prováděna automaticky při otevření konkrétního souboru ke zpracování. Zároveň je upraven kontrast a barvy obrazu pro lepší čitelnost dat. Implementace této funkce je blíže popsána v kapitole 5.2.1.

Ruční označení rohů

V případě, že je snímek špatně oskenován a nemohou být nalezeny všechny čtyři rohy, což se v praxi stává poměrně často, ořízne se snímek špatně a rohy musí být označeny ručně. K tomuto případu slouží nástroj Najít rohy, kde uživatel myší klikne postupně po směru hodinových ručiček na každý roh, začíná levým horním, poté pravým horním, pravým dolním a končí levým dolním. Rohy by měly být označeny co nejpřesněji, proto může uživatel vždy začít znovu. Více v kapitole 5.2.3. Na ilustraci Obrázek 6 – Ukázka nesprávně nalezených rohů je vidět vlevo ukázka špatně oskenovaného dokumentu, rohy byly nalezeny nesprávně, vpravo snímek po ručním označení rohů.



Obrázek 6 – Ukázka nesprávně nalezených rohů

Přiřazení a načtení XML šablony podle čárového kódu

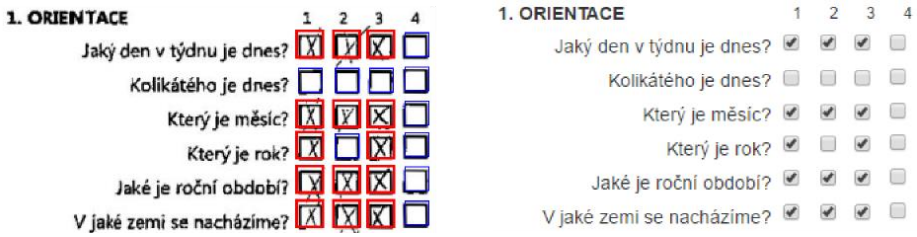
Další požadovanou funkčností je přečtení čárového kódu ze snímku a zobrazení příslušné šablony pro doplnění dat. Šablony jsou uloženy ve složce na serveru, je tedy vždy potřeba ji načíst a převést z XML formátu do zobrazení v HTML.

Ruční přiřazení šablony

Čitelnost čárového kódu je závislá na kvalitě snímku, a proto musí být k dispozici záložní řešení pro případ, že kód nebude rozeznán. Uživatel je v této situaci vyzván, aby šablonu přiřadil ručně, k dispozici má všechny aktuálně nahrané předlohy se zkratkou názvu, v případě, že nezvolí správnou, může ihned vybrat jinou a až potom pokračovat vložením dat.

Automatizované čtení zaškrtnutých polí

Velký důraz byl kladen na požadavek automatického čtení zaškrtnutých dat. Tato funkce může ušetřit spoustu času a stereotypní práce při zpracování dotazníku. Byl aplikován OMR algoritmus, který kontroluje každé zaškrtávací pole, podle předem definovaných souřadnic. Tyto souřadnice čte z dříve vytvořeného XML souboru pro daný formulář. Software je náchylný k hůře vyplněným polím, proto je potřeba vždy data zkontrolovat. Pro usnadnění kontroly slouží označení nalezených zaškrtnutých polí červeně a prázdných polí modře, Obrázek 7. V případě, že je prázdné pole červené nebo obráceně, lze upravit hodnotu ručně. Vysvětlení implementace je uvedeno v kapitole 5.3.1.



Obrázek 7 - Automatické rozpoznání zaškrtnutých polí

Ruční vkládání textových a číselných hodnot do formuláře

Kromě zaškrtávacích polí jsou ve funkčních vyšetřeních vždy i místa, kam se vyplňují například získané body, rodné číslo, datum a další. Při zpracování je tedy vždy ponechán prostor pro vepsání těchto dat do příslušných polí v HTML formuláři, který reprezentuje XML šablonu.

Validace vyplněných dat

Názvy XML šablon s vyplněnými daty, které jsou odesílány na webové uložiště, jsou generovány podle určitých klíčových atributů konkrétního formuláře. Tyto atributy tedy musí být kontrolovány před uložením vyplněné šablony. Zjišťuje se, zda byly vyplněny a u některých, například u rodného čísla i jestli ve správném formátu. Toto se vykonává ještě před samotným zápisem do XML, a pokud je některá hodnota nesprávně, uživatel je na ní upozorněn vyskakovacím oknem a nelze pokračovat dále. Jedinou výjimkou je rodné číslo ve špatném formátu. Stává se totiž, že je vyplněno chybně již do papírového formuláře. V tomto případě je tedy uživatel pouze upozorněn podbarvením textového pole, že rodné číslo je nevalidní, nicméně může pokračovat dále. Více v kapitole 5.3.2. Barevné podbarvení je ukázáno na Obrázek 8.

Obrázek 8 - Upozornění na nevalidní rodné číslo

Ukládání a zobrazení zpracovaných dotazníků

Pro pozdější správu a práci s dotazníky je potřeba ukládat získaná data do předvytvořené šablony. Jelikož HTML formulář, do kterého jsou v prohlížeči data zapisována, je převedená kopie XML šablony, data jsou jednoznačně identifikovaná a převod zpět i s daty je tím usnadněn. Zároveň je nutné uchovávat i původní snímek pro zpětnou kontrolu. Obě tyto položky jsou proto ukládány do jedné složky se společným názvem, každý dotazník má tedy vlastní a takto jsou data uložená na serveru, dokud se je uživatel nerozhodne exportovat. Do té doby je vždy možné se k již zpracovaným dotazníkům vrátit a provést změny.

Export

Digitalizované informace musí být snadno exportovatelné, proto byl zvolen jednoduchý způsob ukládání celého souboru do komprimovaného archivu ve formátu ZIP, který je možno stáhnout. Název staženého souboru obsahuje také přesné datum, kdy byl vytvořen. Dotazníky v tomto formátu jsou plně kompatibilní s webovou aplikací pro

správu a export a po nahrání jsou automaticky uloženy do databáze. Implementace export je více popsána v kapitole 0.

Mazání dotazníků

Funkční vyšetření by mělo být možné smazat kdykoliv v průběhu zpracování, to znamená hned při příchodu na server, během digitalizace i po dokončení digitalizace společně s vyplněnou XML šablonou. K tomuto účelu je vždy vhodně umístěno tlačítko pro smazání dotazníku, které v závislosti na situaci smaže příslušné soubory ze serveru. Data jsou mazána i vždy po jejich úspěšném exportu.

3.1.2 Nefunkční požadavky

Tyto požadavky definují další nároky na aplikaci, například její vlastnosti, chování systému nebo další veličiny, které nevypovídají přímo o funkčnosti programu.

Ovládání

Aplikace by měla být ovladatelná pomocí standartních prvků. Je důležité na uživatele klást co nejméně požadavků na učení se nových věcí a vytvořit obsluhu programu intuitivní a přirozenou. Ovládání bylo proto při návrhu diskutováno s několika méně technicky zdatnými osobami a optimalizováno tak i pro uživatele s nižšími počítačovými dovednostmi.

Vzhled

Vzhled programu by měl být příjemný, ale přehledný a jednoduchý a měl by nabízet uživateli snadnou orientaci v prostředí.

Minimalizace potřebné práce

Protože digitalizace velkého množství dotazníků je stereotypní a navíc časově náročná práce, byl vznesen požadavek, aby uživatel musel provádět opravdu pouze nezbytné operace. Systém tedy co nejvíce vede uživatele procesem a minimalizuje počet potřebných úkonů a tím i snižuje pravděpodobnost chyby.

Spárování šablony a snímku

Po převedení dat z jednoho formuláře do elektronické podoby jsou exportovány dva vzniklé soubory, XML vyplněná šablona s ořezaný a upravený snímek. Tento snímek musí mít název vygenerovaný podle klíčových atributů a toto označení musí být uvedeno v digitalizovaném dotazníku pro budoucí spárování na webovém uložišti.

Podpora

Při práci může nastat neobvyklá situace nebo chyba, ale také se může stát, že si uživatel pouze neví rady. Aby nebylo nutné vždy komunikovat se správcem systému, nabízí aplikace záložky s nejčastějšími možnými problémy, které se objevily během testování, také obsahuje uživatelskou dokumentaci s návodem k použití.

3.2 Využité technologie

Jak již bylo v této práci několikrát řečeno, výstupem je webová aplikace. To s sebou však nese řadu komplikací spojených s možnostmi vývojových prostředků a se strukturou a principem internetových aplikací obecně. S nahlédnutím k již hotové části webového GDiagu, kterou je aplikace pro správu a export a po konzultaci s vedoucím této bakalářské práce, bylo rozhodnuto využít stejný technologický základ. Následující technologie umožňují splnit všechny funkční požadavky a zároveň jsou moderním nástrojem a nabízí snadnou udržitelnost vyvinutého softwaru.

3.2.1 Aplikační rozhraní

Webová aplikace GDiag pro správu a export běží na platformě Microsoft Windows, na Webový scanner platil požadavek, aby spolu mohly sdílet jeden stejný server. Proto bylo rozhodnuto pro použití stejných technologií, které jsou s touto platformou kompatibilní a podporují práci s Microsoft .NET frameworkem. Ten nabízí širokou škálu knihoven, od zpracování obrazových dat po práci s databázemi a mnoho dalších rozšíření, ale především také prostředí umožňující běh aplikace.

Nejrozšířenější nástroj pro práci s webovými aplikacemi pod tímto frameworkem je ASP.NET. Jedná se o nadstavbu, která poskytuje moderní a profesionální přístupy k tvorbě rozsáhlých webových aplikací v prostředí Microsoft Windows.

Jako vývojové prostředí bylo zvoleno Microsoft Visual Studio Enterprise 2015. Nabízí velké množství možností práce s kódem, debugger, editor GUI, podporu verzovacích systémů, diagnostické nástroje nebo snadnou integraci NuGet balíčků.

ASP.NET je odvozeno od starší technologie ASP, je založeno na CLR (Common Language Runtime), který je sdílen všemi aplikacemi podporujícími platformu .NET. Umožňuje využívat programovací jazyky C#, J#, VB.NET, psát objektově orientovaný kód, navíc zpřístupňuje všechny nástroje, které nabízí .NET framework. Internetové aplikace vytvářené na ASP.NET jsou v zásadě tří typů. Nejstarší přístup pod tímto frameworkem je ASP Web Pages, v současnosti je spíše na ústupu. Dále velice hojně používaný ASP Web Forms, který disponuje širokou škálou funkcí a inovativním přístupem. A třetí, ASP.NET MVC, který je založen na architektuře MVC (Model View Controller). [4]

Pro vývoj této aplikace byla zvolena metoda vývoje ASP.NET MVC 5, což je v této době nejnovější technologie, kterou ASP.NET Framework nabízí. Odděluje od sebe logickou, datovou a prezentační vrstvu.

Model-View-Controller

Architektonický vzor Model-View-Controller je v dnešní době velice populární. Nejedná se pouze o architekturu určenou k vývoji webových aplikací, ale i k tvorbě desktopových a dalších, především rozsáhlých softwarů. Jak již bylo řečeno výše, MVC od sebe odděluje logiku aplikace, datovou doménovou vrstvu a uživatelské rozhraní, což je velmi výhodné i u internetových programů. Jeden soubor běžné webové stránky obsahuje databázové dotazy, logiku i HTML tagy a je tedy vše zamotané do sebe jako špagety, proto se tento kód označuje pojmem „spaghetti code“, po čase začne být velmi nepřehledný. Naopak v MVC je pro stejnou stránku potřeba vytvořit tři soubory, tři samostatné třídy, dědící z abstraktních tříd Model, View a Controller.

Model – obsahuje datovou vrstvu aplikace, mohou to být databázové dotazy nebo validace a další. Model vůbec neví o tom, jak budou výstupní data zformátována a

vypsána. Tato komponenta však může mít i některé metody, které se volají ještě před vytvořením její instance. Například pro validaci dat, jako je délka hesla. Udržuje určitou datovou strukturu, se kterou se dále v Controlleru pracuje. Modely však na Views ani Controllerech nejsou vůbec závislé.

View – prezentace aplikace. Z pohledu uživatele se jedná o uživatelské rozhraní, je to část aplikace, se kterou interaguje a jediná kterou vidí. Může zobrazovat data zpracovaná Controllerem, ale také čistě z Modelů. Obsahuje všechny HTML tagy a Javascriptové funkce.

Controller – komunikuje s datovou i prezentační vrstvou. Zprostředkovává zpracování dat pro naplnění Views, ale i příchozí data z této uživatelské vrstvy. Controller drží celý systém pohromadě, propojuje komponenty. Obsahuje pouze kód vybraného programovacího jazyku, v případě této aplikace je to C#. [8] [9]

Další využité technologie

ASP.NET framework disponuje řadou nástrojů a funkcí, jako je například Razor technologie. Nejedná se o čistý programovací jazyk, nicméně podporuje syntaxi C# i VB. Umožňuje vkládat serverově založený kód do Views, podobným stylem jako PHP nebo klasické ASP. Součástí je i technologie Razor Helpers, která zjednodušuje běžné operace jako odesílání formulářů nebo validaci vstupů. [10]

Další funkcí ASP.NET jsou Partial views. Jedná se o samostatnou komponentu, která je renderována v nadřazeném View pomocí Razoru. Tato technologie umožňuje dosáhnout rozdělení určité stránky na několik částí, kde do každého Partial view mohou být vykreslena rozdílná data vlastním Controllerem, který lze zavolat i pomocí JQuery funkce load() a tím provádět změny na stránce bez nutnosti ji celou znovu načíst.

Pro dynamické prvky webu, bez nutnosti obnovovat aktuální View byly použity technologie JavaScript, JQuery a AJAX.

Pro návrh vzhledu aplikace a designování jednotlivých prvků, byl využit Twitter Bootstrap 3, který poskytuje snadno implementovatelné nástroje a především unifikované pro většinu moderních prohlížečů.

3.2.2 Externí knihovny

- **AForge** (URL: http://www.aforgenet.com/)
  – pro veškerou práci s obrazem, jako je úprava barev a kontrastu ofocených dotazníků, čtení zaškrtnutých polí nebo ořezávání snímků, byly využity funkce knihovny AForge. Tato knihovna je zveřejněna pod GPL licencí.
- **Barcode imaging class**
  (URL:http://www.codeproject.com/Articles/42852/Reading-Barcodes-from-an-Image-III)
  – třída pro nalezení a rozpoznání čárových kódů na stránce, vydána pod CPOL licencí.
- **DotNetZip** (URL: https://dotnetzip.codeplex.com/)
  – knihovna pro komprimování a archivování souborů a složek do formátu ZIP. Zveřejněna pod licencí Ms-Pl.

3.3 Funkční specifikace

Tato kapitola popisuje chování aplikace, fáze procesu zpracování dotazníku a blíže specifikuje práci s daty.

3.3.1 Práce s dotazníkem

Naskenované dotazníky jsou z tiskárny ve zdravotnickém zařízení odesílány skenerem přes FTP přímo na server aplikace do podsložky Data. Složka přístupná přes FTP obsahuje také podsložku Templates, do které jsou nahrávány XML šablony. Funkční vyšetření jsou přijímána ve formátu TIF, jeden takový soubor vždy obsahuje balíček dotazníků od jednoho pacienta. Takto uložené nové dotazníky jsou při každém otevření aplikace převedeny na jednotlivé snímky formátu JPEG a označeny názvem balíčku s pořadím v něm.

Před digitalizací dat z dotazníku je potřeba učinit několik úprav. K tomu jsou využívány možnosti knihovny AForge. Snímek je nejprve převeden na černobílý a prahováním připraven k hledání objektů v obraze. Jsou hledány předdefinované rohy,

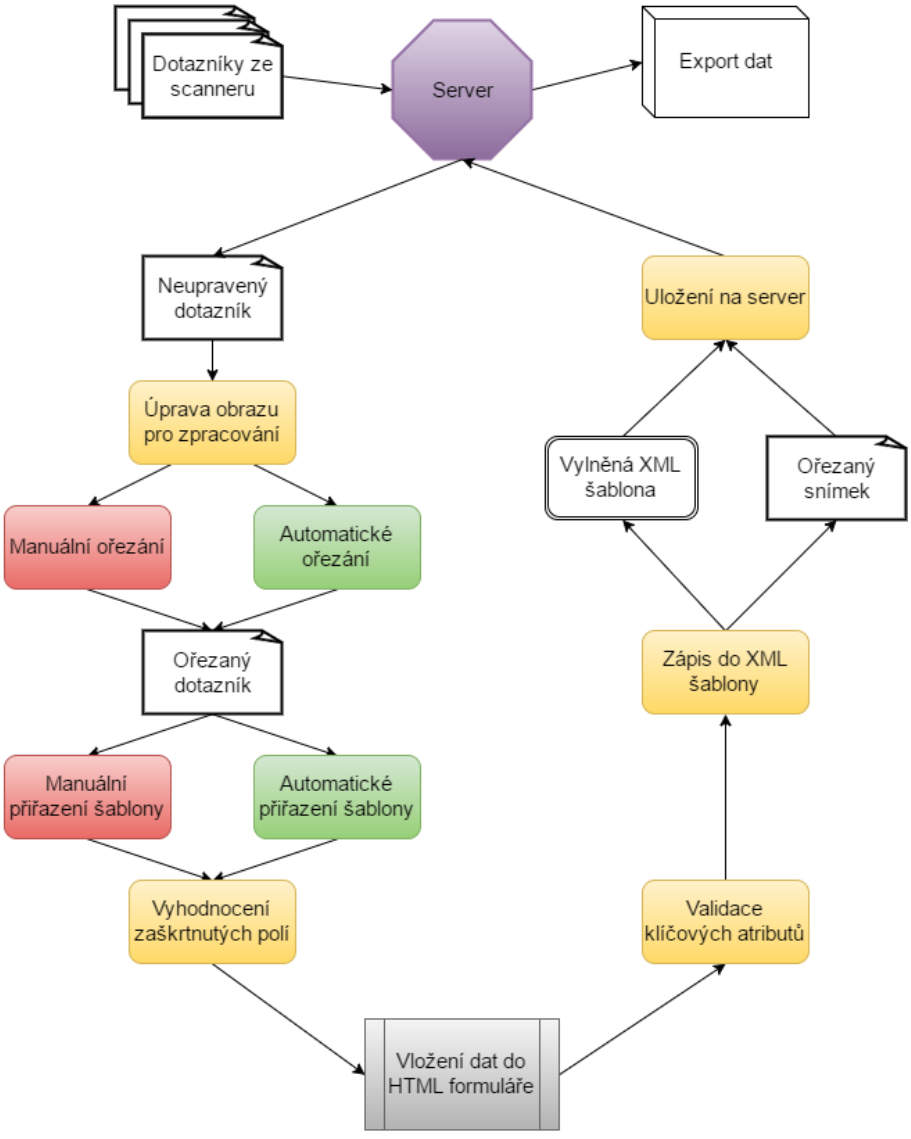
čtyři černé čtverce v rozích dokumentu. Pokud jsou všechny nalezeny, je snímek ořezán a zůstane pouze plocha formuláře. V opačném případě aplikace vybídne uživatele k označení rohů myší v prohlížeči a poté je teprve snímek ořezán. Ve chvíli, kdy je snímek ořezán, je procházen funkcemi knihovny BarcodeImaging, je hledán čárový kód. Po nalezení je navrácen název šablony. Pokud kód není rozpoznán, opět je uživatel aplikací upozorněn a název šablony musí být vybrán ručně pomocí nabídky, která zobrazuje obsah složky s šablonami. Automaticky nebo manuálně získané označení je využito k načtení dané šablony ze složky na serveru.

Šablona je uložena ve formátu XML, pro vykreslení slouží funkce knihovny GDiag, které převádí XML elementy na HTML. Každý element obsahuje také atribut s přesnými souřadnicemi daného prvku v oskenovaném snímku, ty jsou využity v další fázi digitalizace.

Čtení dat ze zaškrtnutých polí ofoceného dotazníku je prováděno postupným procházením každého příslušného elementu v XML šabloně, vyříznutím jej ze snímku podle daných souřadnic a vyhodnocováním zastoupených barev v oblasti pro zaškrtnutí. Hodnoty jsou uloženy do konkrétních elementů XML šablony a ta je vykreslena v prohlížeči. Jakmile uživatel HTML formulář zkontroluje a doplní další hodnoty podle oskenovaného snímku, odešlou se data na server. Nejprve je provedena kontrola, zda byly vyplněny klíčové vstupy. Ty jsou označeny atributem *key*. Pokud ne, je jejich název navrácen do prohlížeče, kde je zobrazen ve vyskakovacím oknu s výzvou k doplnění. Až když jsou všechny klíčové položky vyplněny, mohou být data uložena.

Data na server přichází s názvem elementu a vyplněnou hodnotou. Jsou postupně procházena a po jednom ukládána do XML. Klíčové atributy jsou použity k vytvoření názvu souboru. Stejně je pojmenován i původní snímek a tento název je přidán do XML šablony, pro snadnější pozdější spárování. Důležitou funkcí je také zapamatování si klíčových položek pro příští dotazník k urychlení jejich vyplnění. U následujícího dotazníku je ověřováno, zda obsahuje zapamatované elementy a ty, které ano, jsou vloženy do příslušného pole. Soubory jsou ve formátu XAML ukládány na serveru ve složce Zpracované dotazníky a pro každý snímek s příslušnou doplněnou XML šablonou je vytvořena zvláštní složka.

Export všech zpracovaných souborů musí dodržovat určitý formát, aby byl kompatibilní s webovou aplikací pro správu a export. Balíček všech snímků i vyplněných šablon musí být pohromadě v jedné složce a šablony musí obsahovat atribut s názvem snímku. Při exportu jsou tedy obsahy všech složek umístěny do jednoho společného archivu ve formátu ZIP a ten je možné stáhnout. Takto zabalený archiv lze snadno nahrát do aplikace pro správu a export, kde je obsah uložen do databáze. Celý proces práce s dotazníkem je zobrazen na Obrázek 9.

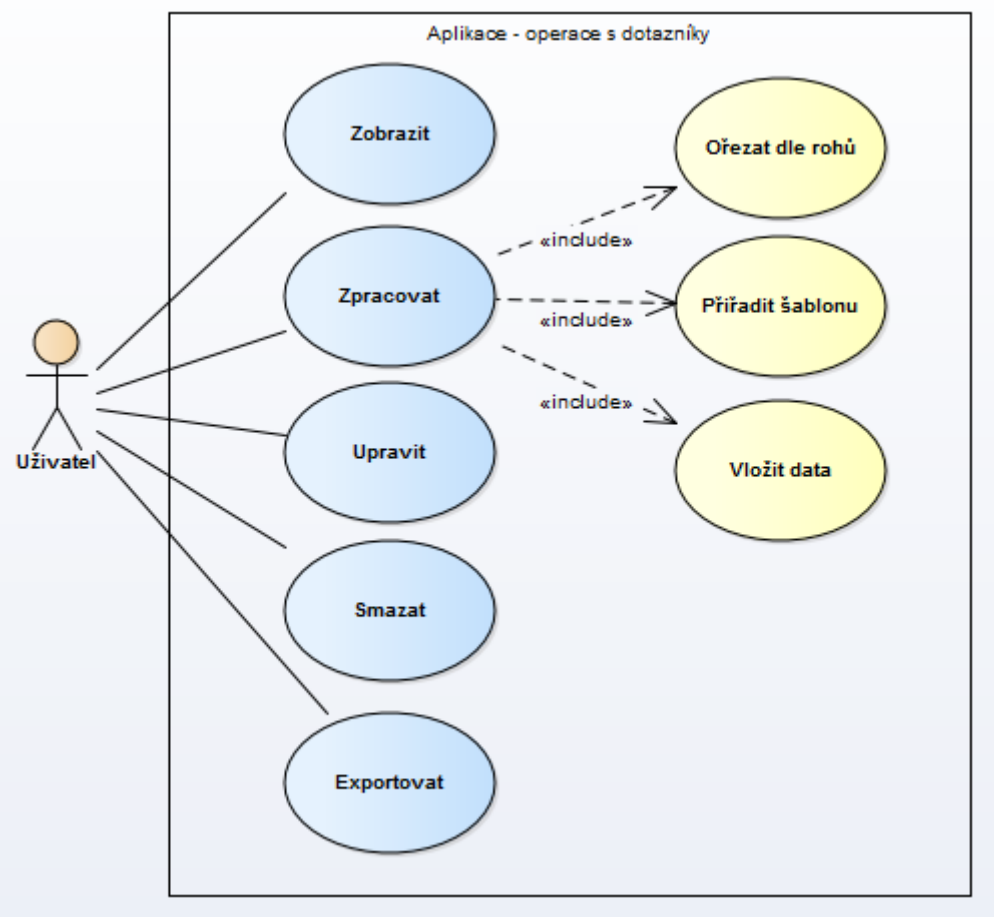


Obrázek 9 - Diagram zobrazující práci s dotazníkem v aplikaci

3.3.2 Případy užití

UML je soubor grafických notací, který se používá k navrhování a specifikaci programových systémů. Způsob navrhování diagramů je standardizován skupinou Object Management Group.

Diagram užití (Use Case diagram), zobrazuje chování aplikace z hlediska uživatele. Use Case diagram popisuje každý případ užití pomocí základních objektů. Actor, představuje kohokoliv nebo cokoliv mimo systém, v tomto případě jej představuje pouze uživatel. Use Case, akce vedoucí k dosažení určitého cíle, například zpracování dotazníku nebo export. Určuje funkcionalitu, kterou by měl systém mít. A ohraničení, které udává hranice systému. Diagram užití pro tuto aplikaci je zobrazen na Obrázek 10. [12] [13]



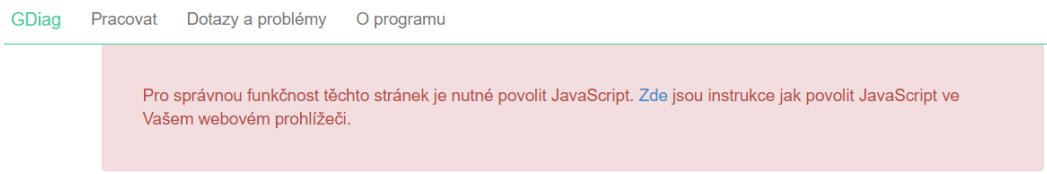
Obrázek 10 - Use Case diagram aplikace GDiag webový scanner

4 Uživatelská dokumentace

Uživatelská dokumentace slouží jako návod pro obsluhu programu. Popisuje všechny funkce a pomáhá uživateli při jejich používání. V této kapitole budou navíc popsány technické požadavky pro správnou funkčnost.

4.1 Technické požadavky

Pro správnou funkčnost aplikace je nezbytné mít v prohlížeči povolenou podporu JavaScriptu. Velké množství operací v aplikaci je zpracováváno pomocí této technologie. Většina internetových prohlížečů má JavaScript povolen automaticky, v opačném případě je uživatel upozorněn varovnou hláškou obsahující i odkaz k postupu, jak tuto technologii povolit.



Obrázek 11 - Varovná hláška upozorňující na chybějící podporu JavaScriptu

GDiag webový scanner byl testován a plně funkční v několika nejpoužívanějších prohlížečích. Níže jsou uvedeny jejich minimální verze pro správný chod aplikace. Na starších verzích nemusí být zaručena plná funkčnost a doporučuje se aktualizace na novější.

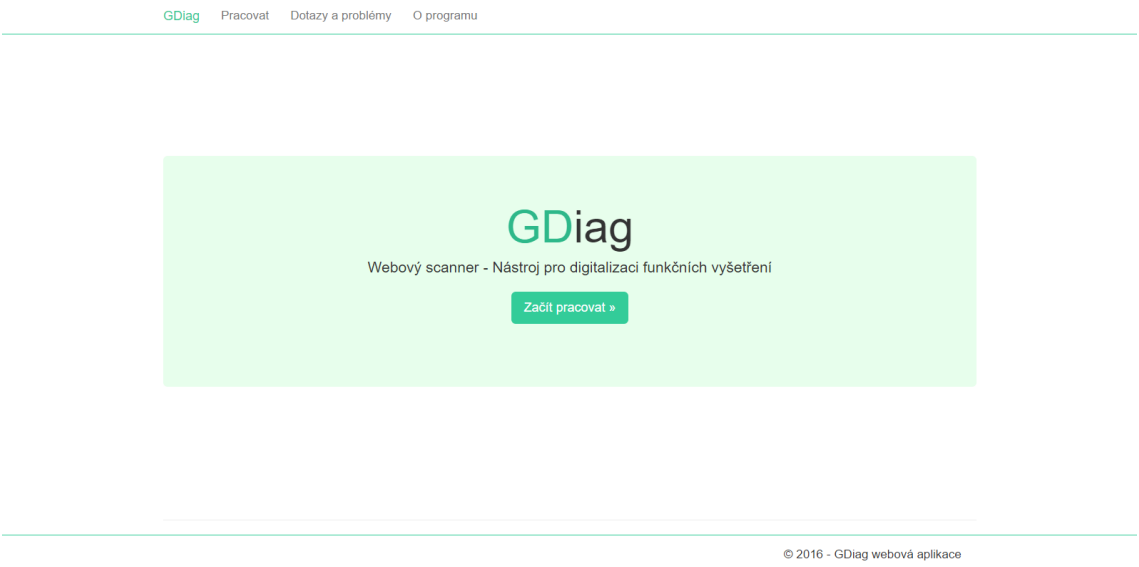
- Google Chrome 48.0 a vyšší
- Microsoft Internet Explorer 9 a vyšší
- Mozilla Firefox 44.0 a vyšší
- Microsoft Edge 2.5 a vyšší
- Vivaldi 1.0 a vyšší

4.2 Ovládání a funkce

Tato sekce obsahuje návod k obsluze programu a jeho jednotlivých funkcí. Má za cíl provést uživatele celou procedurou zpracování funkčního vyšetření v této aplikaci.

4.2.1 Úvodní strana

Při vstupu do aplikace se zobrazí úvodní stránka. Tlačítko *Začít pracovat*, slouží k okamžitému vstupu do webového scanneru. Další možností jak rychle vstoupit do aplikace je odkaz *Pracovat* v menu stránky. Mimo to jsou zde záložky *Dotazy a problémy* a *O programu.*



Obrázek 12 - Úvodní strana s odkazy pro vstup do funkční části aplikace

Do sekce *Dotazy a problémy* budou postupně přidávány nejčastější dotazy týkající se aplikace a problémy, které mohou nastat. Bude sloužit jako první orientační bod v případě jakýchkoliv komplikací.

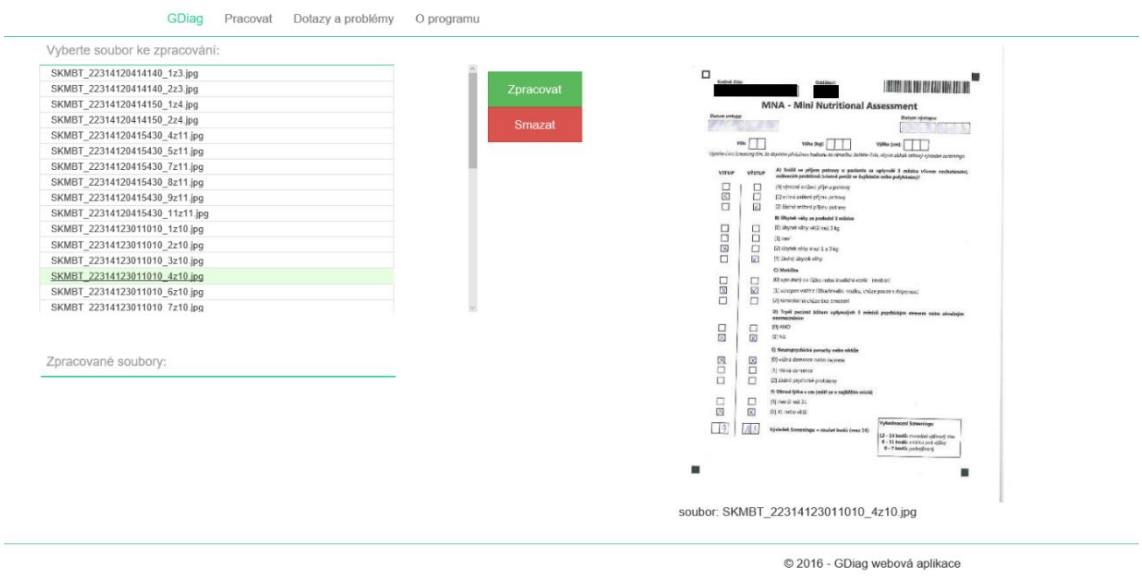
Sekce *O programu* obsahuje kompletní uživatelskou dokumentaci, s návodem jak aplikaci obsluhovat a dalšími postupy.

4.2.2 Pracovní plocha I. - Uživatelská nástěnka

Uživatelská nástěnka je domovskou stránku uživatele, která poskytuje přehled o současném stavu funkčních vyšetření na serveru. Z tohoto bodu je možné vstoupit do dalších sekcí aplikace. Samotná nástěnka se skládá z několika částí.

Soubory ke zpracování

V levé horní části se nachází seznam souborů ke zpracování. Každý řádek obsahuje jméno souboru a po najetí ukazatelem myši na řádek se v pravé části obrazovky zobrazí náhled snímku. Aplikace poskytuje možnosti zpracovat nebo smazat tento snímek. Pro zpracování souboru je třeba kliknout na zvolený řádek, či použít zelené tlačítko *Zpracovat*. Pro odstranění slouží červené tlačítko *Smazat* a tato operace je nevratná.

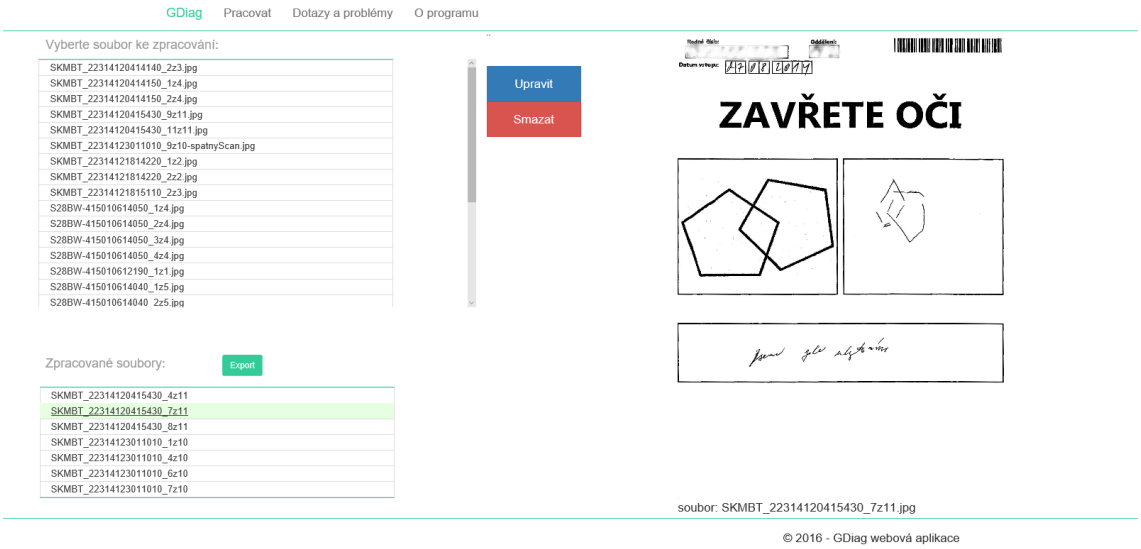


Obrázek 13 - Uživatelská nástěnka se soubory čekajícími na zpracování

Zpracované soubory

V levé části se také nachází seznam zpracovaných souborů. Při najetí ukazatelem myši na řádek se souborem se opět objeví náhled zpracovaného snímku. Místo zeleného tlačítka *Zpracovat* je nyní k dispozici modré tlačítko *Upravit*. Jeho stiskem nebo kliknutím na řádek seznamu je možné se vrátit k přepisování dat z oskenovaného

formuláře a upravit hodnoty ve vyplněné šabloně, či zkontrolovat jejich správnost. Červené tlačítko *Smazat* odstraní ofocený soubor i s vyplněnou šablonou.



Obrázek 14 - Uživatelská nástěnka při najetí myší na zpracované soubory

Export

Napravo od nadpisu *Zpracované soubory* je umístěno zelené tlačítko s názvem *Export*, které umožňuje exportovat zpracovaná funkční vyšetření. Digitalizované dotazníky jsou staženy v jednom souboru ZIP, který v názvu obsahuje datum provedení exportu. Takto exportovaný soubor je plně kompatibilní s aplikací pro správu a export. Po provedení této operace jsou všechny zpracované soubory z aplikace smazány. Pro případ, že by během stahování došlo k chybě, jsou dotazníky dočasně uloženy na serveru. Pro jejich obnovení je potřeba kontaktovat administrátora.

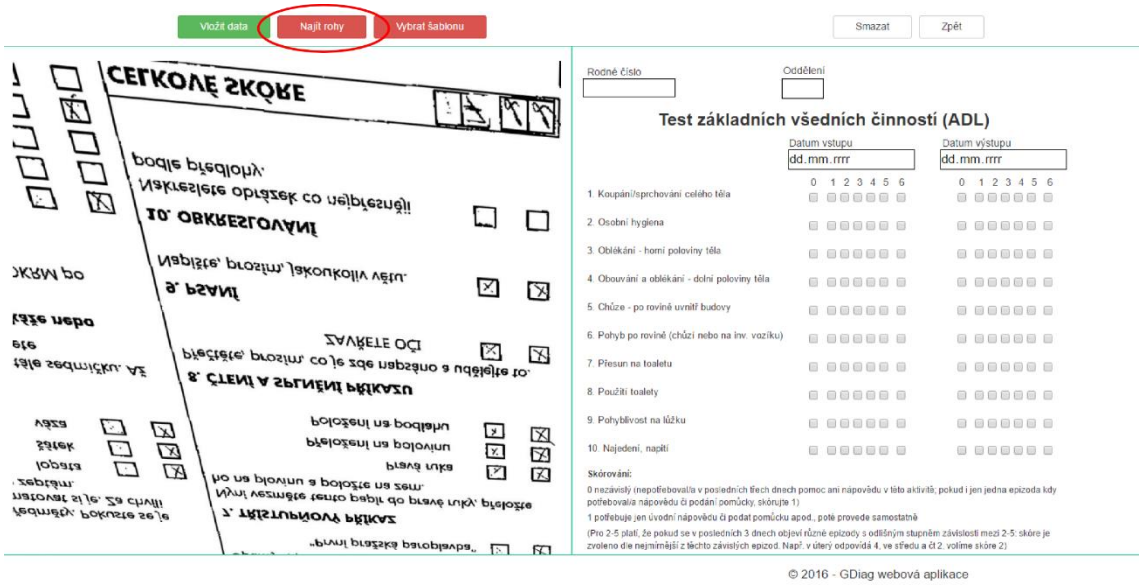
4.2.3 Pracovní plocha II. – Digitalizace

V této sekci pracovní plochy je prováděno zpracování souboru a přiřazení specifické šablony podle čárového kódu.

Obsahuje tlačítka Vložit data, Najít rohy, Vybrat šablonu, Smazat a Zpět.

Najít rohy

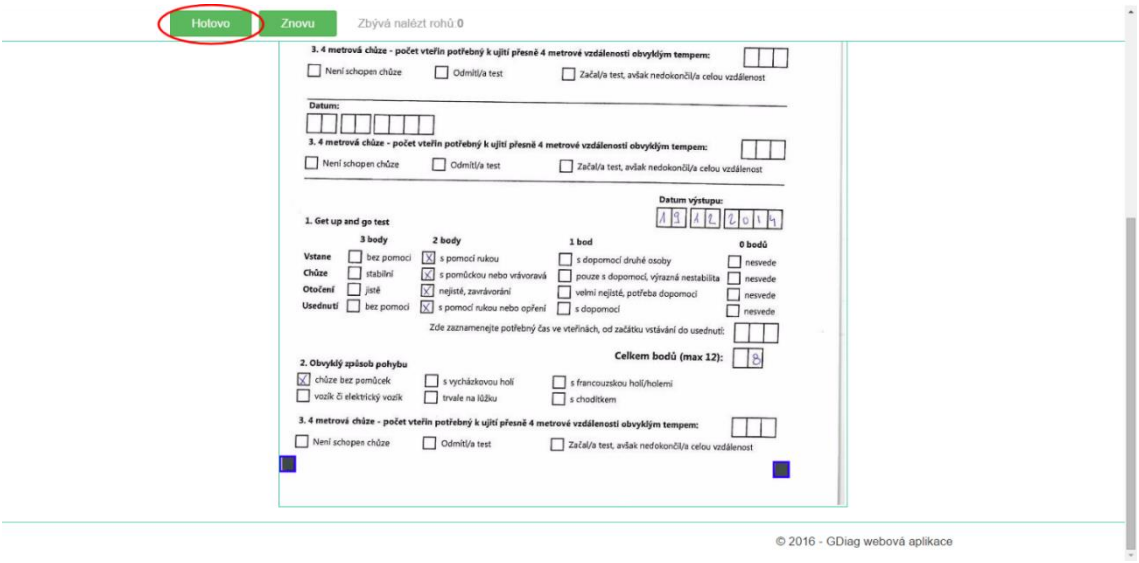
Snímek je nejprve ořezán podle předdefinovaných rohů, které vytyčují oblast formuláře a jsou znázorněny čtyřmi černými čtverci. Pokud je některý z těchto bodů hůře čitelný nebo úplně chybí, ofocený dotazník je ořezán špatně. V tomto případě je potřeba najít rohy ručně a to kliknutím na tlačítko *Najít rohy*.



Obrázek 15 – Špatně ořezaný dokument

V sekci pro hledání rohů je zobrazen pouze původní snímek. Ruční označování rohů musí být prováděno postupně takto: Levý horní roh, Pravý horní roh, Pravý dolní roh, Levý dolní roh. Roh se označuje kliknutím levým tlačítkem myši do středu čtverce určující roh stránky. Pokud dojde k nepřesnému označení, je možné začít znovu pomocí

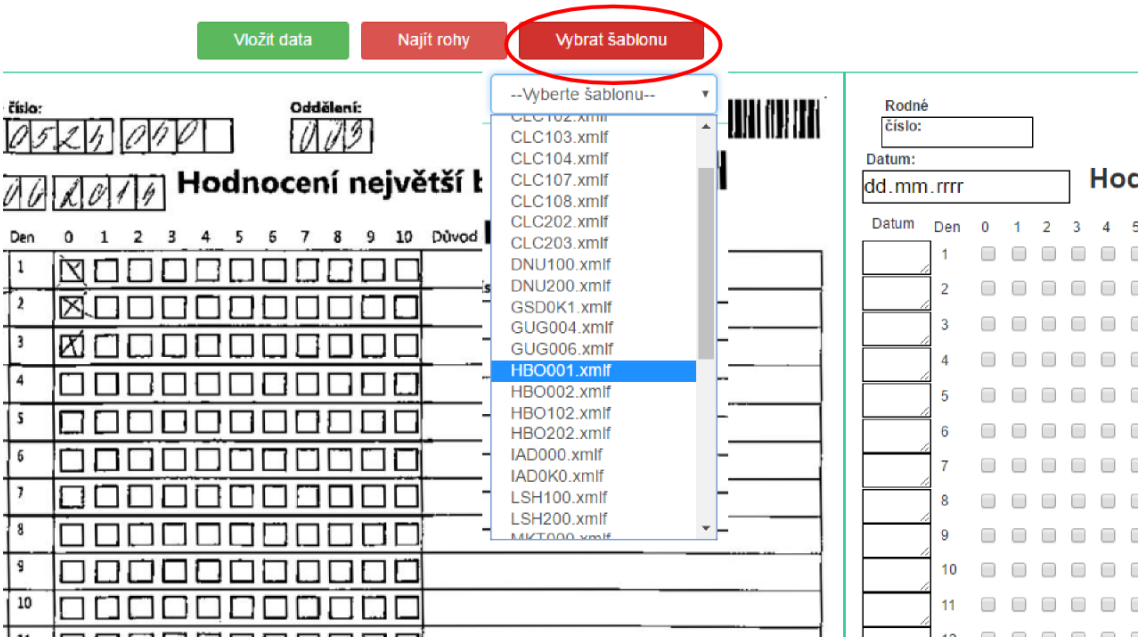
tlačítka Znovu. Až v případě, že již byly označeny všechny rohy, je možné použít tlačítko Hotovo a tím ořezat snímek a vrátit se na pracovní plochu pro přepis dat do formuláře.



Obrázek 16 – Ručně označené rohy

Vybrat šablonu

Po nalezení rohů je podle čárového kódu přiřazena šablona. Pokud je čárový kód hůře čitelný nebo je snímek špatně ořezán, může být načtena špatná šablona nebo se nezobrazit vůbec. Tuto situaci řeší tlačítko *Vybrat šablonu*. Uživatel má na výběr ze všech formulářů, které jsou aktuálně uloženy v aplikaci. Šablona se vybere kliknutím na konkrétní název.

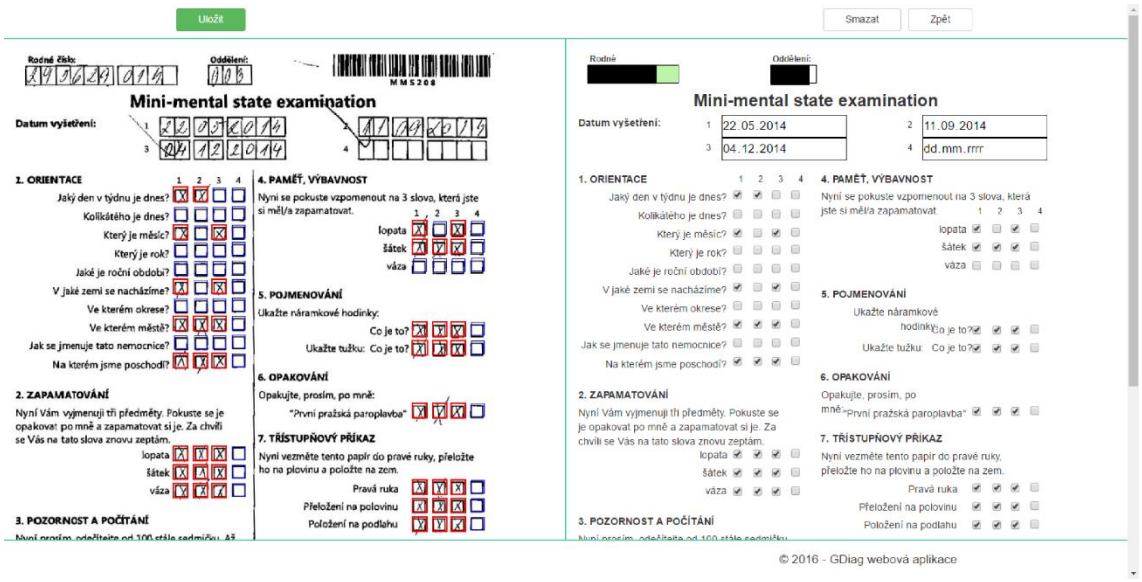


Obrázek 17 - Vybrat šablonu

Vložit data

Tlačítko *Vložit data* slouží pro načtení zaškrtnutých polí a vložení jejich hodnot do prázdné šablony. Vyplněná zaškrtávací políčka jsou označena na původním snímku červeně a prázdná modře. Protože jsou data vyplňována ručně, mohou být špatně rozpoznána a je třeba je před uložením zkontrolovat.

Kromě těchto dat jsou během práce ukládány také další hodnoty jako rodné číslo, oddělení nebo datum a pokud je následující formulář opět obsahuje, jsou vložena na příslušná místa. Tato data jsou brána z předchozího zpracovávaného dotazníku, pokud následující vyšetření není stejného pacienta nebo je vyplněn jiný den, je potřeba je přepsat.

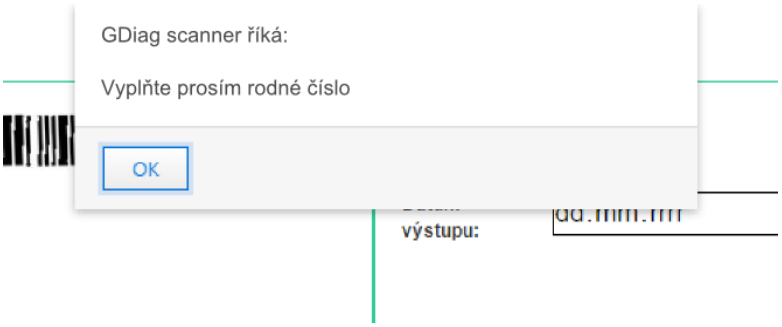


Obrázek 18 - Pracovní plocha po vložení dat

Kontrola rodného čísla a klíčových atributů

Rodné číslo je jeden z klíčových parametrů pro následné ukládání vyplněné šablony. Proto je důležité, aby bylo vyplněno správně. Protože se rodné číslo vyplňuje ručně, obsahuje aplikace základní kontrolní funkci pro zamezení překlepům nebo jiné náhodné chyby. V případě, že je rodné číslo nevalidní, je uživatel upozorněn zčervenáním příslušného textového pole. Pokud je vše v pořádku, políčko je zelené.

Každá šablona má několik klíčových atributů, které musí být vždy vyplněny. Nejčastěji rodné číslo, oddělení a datum. Pokud není některý atribut vyplněn, šablonu není možné uložit a konkrétní chyba je uživateli vypsána pomocí vyskakovacího okna.



Obrázek 19 - Chybová hláška při chybějícím rodném čísle

Uložit

Funkce *Uložit* vyplněnou šablonu uloží ve formátu XML a spolu s ořezaným snímkem připraví na export. Soubory přesune do složky *Zpracované soubory*, kde je možné se k nim vrátit v sekci *Uživatelská nástěnka*. Pro urychlení práce je po uložení automaticky načten další snímek ze složky *Soubory čekající na zpracování*.

Smazat a Zpět

Tlačítko *Smazat* slouží k odstranění snímku, tato akce se nedá vrátit. Odkaz *Zpět* je určen k návratu do sekce *Uživatelská nástěnka*.

5 Implementace

V kapitole Implementace jsou stručně popsány programátorské postupy u jednotlivých funkčností aplikace.

Jak již bylo vysvětleno v kapitole 3.2, aplikace využívá architektury MVC 5, frameworku ASP.NET. Během implementace však nebylo potřeba vytvářet modely, proto je struktura aplikace založena pouze na Controllerech a Views. Právě Controller umožňuje využití rozsáhlých knihoven C# a tím pracovat například se snímky dotazníků nebo XML šablonami. Mnohdy náročné výpočty a metody tedy zůstávají odděleny od prezentační vrstvy. Controller zde přímo plní daty View, na rozdíl od běžného MVC, kde tuto funkci může vykonávat i Model.

Je-li uživatelem vyvolána určitá akce, například vložení dat nebo i samotné zobrazení domovské stránky, chová se aplikace ve většině případů obecně stejně. Nejprve je zavolán Controller nesoucí vždy v názvu označení dané akce. Pokud by se jednalo o akci Ukázka, bude název UkázkaController. Ten zpracuje data. Mohou to být data vyplněná uživatelem nebo i data uložená na serveru. Po dokončení je odešle do View s příslušným označením Ukázka, kde jsou zobrazena uživateli. Jak již bylo řečeno výše, Controller může ke zpracování dat využívat knihovny C# a další pomocné třídy a metody. Ty jsou zpravidla uloženy ve složce Helpers.

5.1 Zobrazení uživatelské nástěnky

V této situaci je při vstupu do uživatelské nástěnky zavolán kontroler *ScanController*, který zpracuje dotazníky uložené na serveru ve složce Content a cesty k těmto souborům odešle do view s názvem *Scan* a ten je vypíše uživateli.

Controller nejdříve prochází složku s nezpracovanými dotazníky. Nově příchozí funkční vyšetření jsou většinou uložena ve formátu TIF a obsahují jeden až několik snímků. Tyto snímky jsou jednotlivě procházeny v cyklu a pomocí funkcí *Image.SelectActiveFrame()* a *Image.Save()* je každý zvlášť ukládán ve formátu JPEG. Je označen původním názvem doplněným o jeho pořadí v původním souboru. Zároveň je každý snímek za využití pomocné statické třídy *ImgProcess* zmenšen na jednotnou velikost. Kromě toho je uložen i náhledový obrázek o menších rozměrech. Před zpracováním každého balíčku oskenovaných formulářů je kontrolováno, zda se již ve složce nenachází. Po převedení je původní TIF soubor smazán. Ve složce jsou procházeny také všechny JPEG snímky pro případ, že byl přijat na server dotazník i v tomto formátu.

Cesty k těmto snímkům a k již digitalizovaným souborům na server, jsou ukládány

```
images.SelectActiveFrame(FrameDimension.Page, i - 1);
//načte a pokud je snímek moc velký, zmenší ho
img = new Bitmap(images);
if (img.Width > 1024) {
img = ImgProcess.Resize(img, 1024);
}
//pokud již soubory s tímto názvem ve složce jsou, neprovede se zpracování
if (i == 1){
    dupl = ScanHelper.CheckForDuplicate(dupl, nameOfFile);
}
if (dupl == false){
    SaveImgAndThumb(nameOfFile, img);
    filesToView.Add(nameOfFile);

}
```

Kód 1 - Převod souboru z balíčku TIF na samostatný JPEG soubor

do struktur *List*. Ty jsou pomocí speciální vlastnosti ASP.NET frameworku *ViewBag*, předány do View a dále vypsány do HTML tabulek technologií Razor. Zde jsou reprezentovány jako odkazy vedoucí do další sekce. Odkazy jsou opět tvořeny syntaxí Razor. *Html.ActionLink* obsahuje, kromě základních tří parametrů s destinací odkazu, také dva URL parametry (s názvem souboru a příponou) a JavaScriptové funkce *Hover()* a *Loading()*. První jmenovaná funkce předává odkaz tlačítkům ke zpracování a zamění náhledový obrázek. Druhá po kliknutí spouští načítací animaci.

```
<tbody>
    @foreach (string item in ViewBag.files)
        {
            var parameter = item.Split('.');
            <tr>
                <td style="font-size:small" class="row col-md-12">
                    @Html.ActionLink(item, "Process", "Process",
                    new { name = parameter[0], ext = parater[1] },
                    new { onmouseover = "Hover('" + item + "')",
                    onclick = "Loading()", style = "color: #424242; display:block;"
                    })
                </td>
            </tr>
        }
```

Kód 2 - Vypsání tabulky pomocí Razor syntax a definice parametrů odkazu

5.2 Příprava k digitalizaci

Před samotnou digitalizací dat je potřeba nejprve upravit snímek. To znamená ořezat jej podle definovaných rohů, aby obsahoval pouze část s formulářem a zvýraznit kontury pro lepší čitelnost čárového kódu a zaškrtávacích polí. V této fázi je také přiřazena šablona. Všechny tyto akce jsou obstarávány Controllerem *ProcessController*, který přijímá dva parametry, název a příponu obrázku, pro úspěšné nalezení snímku ve složce. Výsledná data odesílá do *ProcessView*.

Kód v Controlleru je rozdělen do dvou bloků, pro levou stranu view, která vrací upravený snímek a pravou část převádějící přiřazenou XML šablonu do HTML.

5.2.1 Upravení snímku

Zpracování snímku je prováděno za využití tříd *ImgProcess* a *PaperProcess*, které jsou přejaty z původního desktopového programu. Obraz je nejprve převeden na černobílý, poté ekvalizován a prahován, čímž jsou zvýrazněny hrany. Následně jsou použity metody *FindCorners()* a *CropByCorners()* k ořezání dle rohů. Obě využívají funkcí knihovny AForge.

První zmíněná metoda nejprve obraz invertuje. Poté využívá třídy *BlobCounter* k nalezení objektů oddělených černým pozadím v určitém rozmezí velikosti. Následně jsou další funkcí AForge filtrovány objekty ve tvaru čtverce. Ty jsou na základě své polohy seřazeny, a jelikož hledané rohy tvoří okraje obrazu, jsou zvoleny nejkrajnější

objekty, odpovídající požadovaným vlastnostem a odeslány jako parametr do funkce *CropByCorners()*. Zde jsou vyhledány vnitřní krajní body jednotlivých rohů a využitím filtru *QuadrilateralTransformation* je snímek podle těchto souřadnic ořezán a vyrovnán.

Pro zjištění čárového kódu slouží funkce *GetPaperType()* využívající knihovnu BarcodeImaging.

```
//Upravení snímku před hledáním rohů
public static Bitmap PrepareImage(System.Drawing.Image images)
{
    var img = new Bitmap(images);
    //Upravení fotky
    img = ImgProcess.Grayscale(img);
    img = ImgProcess.Equalize(img);
    img = ImgProcess.Threshold(img);
    int treshVal = PaperProcess.GetCropThresholdValue(img);
    img = PaperProcess.Thresholding(img, treshVal);
    img = ImgProcess.IndexedBitmapToRgb(img);
    return img;
}
//Automatiké hledání rohů
public static Bitmap AutomaticallyFindCorners(Bitmap img)
{
    List<Blob> corners = PaperProcess.FindCorners(img);
    img = PaperProcess.CropByCorners(img, corners);
    return img;

}
```

Kód 3 - Předzpracování snímku a ořezání podle nalezených rohů

5.2.2 Vykreslení šablony ve view

XML šablona je do HTML převáděna pomocí parseru ze společné knihovny GDiag. Implementace je provedena obdobným způsobem jako u webové aplikace pro správu a export, který se v praxi osvědčil.

Metoda *GXmlToHtmlString()* postupně prochází šablonu a pomocí nástroje ASP.NET *HtmlTextWriter* přepisuje informace z XML do značkovacího jazyku HTML. Šablony jsou nativně tvořeny tak, aby byly přepsatelné do webové formy. Mají tedy, kromě samotných názvů elementů a hodnot, také atributy určující styl konkrétního elementu. Například *text-align*, *font-size*, *padding* a další. Kromě toho mají i přesně dané souřadnice. Těmito atributy je jasně určen styl i pozice každého elementu. V závislosti na tom, o jaký typ elementu se jedná (např.: textfield, label), jsou tyto údaje přidávány do

konstrukce jazyku HTML, čímž vzniká kód. Takto je vytvořen stukturovaný formulář připravený k vykreslení.

Html formulář je odeslán do view, kde je následně dekódován technologií Razor jako PartialView. V případě, že čárový kód nebyl rozpoznán, uživatel zvolí šablonu ručně a technologií AJAX je volána metoda *ChooseTemplatePartial()* pro znovuvykreslení šablony do stejného PartialView.

```
case "label":
    if (nameValueStyle.NameAttribute != String.Empty){
        htmlTextWriter.WriteLine(
        "<label name='" + nameValueStyle.NameAttribute + "' value='" +
        nameValueStyle.ValueAttribute + "' style='"
        + nameValueStyle.StyleAttributes + "'>" + xElement.Value + "</label>");
    }
    else{
        htmlTextWriter.WriteLine(
        "<label name='l" + id + "' value='" + nameValueStyle.ValueAttribute +
        "' style='" + nameValueStyle.StyleAttributes + "'>" + xElement.Value +
        "</label>");
        id++;
        }
    break;
```

Kód 4 - Ukázka tvorby HTML elementu Label

5.2.3 Ruční hledání rohů

Pokud dojde k chybnému nalezení rohů, je použit nástroj pro ruční hledání. Ten využívá moderní technologie HTML5 Canvas, která umožňuje kreslit na plátno pomocí JavaScriptu přímo v prohlížeči. Tomuto plátnu je jako pozadí nastaven původní snímek dotazníku. Při kliknutí myší na konkrétní místo canvasu, je za využití funkce *canvas.strokeRect()* vykreslen čtverec označující tuto pozici. Slouží pro kontrolu, zda se shoduje s původním označením rohu formuláře. Canvas také disponuje funkcí *getBoundingClientRect()*, kterou lze využít k ukládání souřadnic jednotlivých kliknutí do pole. Tyto souřadnice jsou následně při stisku tlačítka Hotovo odeslány do akce *CornersProcess* Controlleru *ProcessController*, který pomocí knihovny Aforge snímek ořízne a uloží. Souřadnice udávají střed značky pro ořezání, okraj formuláře je však dán vždy vnějším okrajem této značky, proto musí být koordináty o tuto vzdálenost posunuty.

```
public static Bitmap CropByHandFoundCorners(Bitmap bitmap, List<int> souradniceX,
List<int> souradniceY)
{
    bitmap = ImgProcess.Resize(bitmap, 800);
    List<AForge.IntPoint> corners = new List<AForge.IntPoint>();
    corners.Add(new AForge.IntPoint(souradniceX[0]+10, souradniceY[0]+10));
    corners.Add(new AForge.IntPoint(souradniceX[1]-10, souradniceY[1]+10));
    corners.Add(new AForge.IntPoint(souradniceX[2]-10, souradniceY[2]-10));
    corners.Add(new AForge.IntPoint(souradniceX[3]+10, souradniceY[3]-10));

    QuadrilateralTransformation filter = new QuadrilateralTransformation(corners,
    bitmap.Width, bitmap.Height);

    return filter.Apply(bitmap);

}
```

Kód 5 - Funkce pro oříznutí snímku podle ručně označených rohů

5.3 Digitalizace

Proces digitalizování dat se skládá z několika částí. Implementace algoritmu pro čtení zaškrtnutých polí, validace klíčových vstupů, odeslání HTML formuláře s daty do Controlleru, přepis těchto dat do XML a následné uložení.

5.3.1 Automatické vložení dat

Většina dat v dotaznících je reprezentována hodnotami v zaškrtávacích polích. Pro co nejrychlejší digitalizaci byla tedy implementována funkce pro rozpoznávání těchto polí. Byl využit stejný algoritmus jako v desktopovém programu FormScan, který vyhodnotí obsah pole a podle toho vrátí hodnotu *true* nebo *false*. Pozice jednotlivých polí jsou přesně definovány v XML šabloně daného dotazníku. Souřadnice začínají nulou v levém horním rohu formuláře, proto je velice důležité jej přesně ořezat. Podle těchto koordinát je poté každý element zvlášť vybrán a vyhodnocen.

Vyhodnocení řeší třída *ImgCheckBox*. Element Checkbox má kromě souřadnic také atributy typu padding, ty určují jeho odsazení a prostor kolem něho. Podle těchto údajů je vypočítáno ohraničení, ve kterém se zaškrtávací pole nachází. Tato část obrazu je vyříznuta a dále se již pracuje pouze s ní. Výřez je nejprve upraven pro snadnější nalezení objektů. K tomu jsou využity filtry knihovny AForge. Pomocí třídy *BlobCounter* je potom vybrán čtverec, který je nejblíže od středu výřezu. Pro vyhodnocení zaškrtnutí je nejprve odfiltrován okraj čtverce, hodnotí se pouze jeho vnitřek. Zde jsou v cyklu procházeny

jednotlivé pixely a pomocí funkce *GetPixel()* je určováno, zda jsou černé nebo bílé. Jelikož byl snímek pro účely hledání objektů invertován, zaškrtnutá část má bílou barvu a pozadí černou. V případě, že bílá část přesáhne určitou hodnotu, je pole považováno za vyplněné.

Hodnota je následně zapsána do XML šablony. Celý formulář je potom odeslán do view a vykreslen uživateli pro kontrolu a doplnění dalších údajů.

```
for (int i = innerRect.X; i < innerRect.Width + innerRect.X; i++)
{
    for (int j = innerRect.Y; j < innerRect.Height + innerRect.Y; j++)
    {
        if (img.GetPixel(i, j).B < 127){
            blackarea++;
        }
    area++;
    }
}
```

Kód 6 - Procházení pixelů v zaškrtávacím poli a vyhodnocování barvy

Další funkcí pro urychlení digitalizace je vkládání naposledy vyplněných klíčových dat. Protože dotazníky od stejného pacienta jsou odesílány v jednom balíčku, který obsahuje někdy i 15 formulářů, klíčová data (například rodné číslo a oddělení) se v této sadě opakují. Při zápisu dat z HTML formuláře jsou tedy vybrány klíčové atributy pro danou šablonu XML a jejich hodnoty zvlášť uloženy do speciální proměnné *Session[lastVals]*. Ukládají se v páru název a hodnota atributu do struktury Dictionary. Pokud následující šablona obsahuje prvky se stejným názvem, jsou tyto údaje pomocí kombinace Razor syntax a JavaScriptu doplněny do příslušných textových polí.

```
private static void GetKeyValues(Dictionary<string,string>sessionData,XElement xE)
{
    if (xE.Attribute("key") != null && xE.Attribute("key").Value == "true"){
        //pokud existuje atribut name, ulož jeho hodnotu jako název
        if (xE.Attribute("name") == null){
            sessionData.Add(xE.Name.ToString(), xE.Value);
        }
        else{
        sessionData.Add(xE.Attribute("name").Value.ToString(), xE.Value);
        }
    }
}
```

Kód 7 - Ukládání klíčových atributů do struktury Dictionary

5.3.2 Validace

Před uložením je potřeba zkontrolovat, zda byly do HTML formuláře vyplněny všechny klíčové atributy, protože je z nich později vytvářen název souboru. K validaci slouží dvě akce. *Validate()* a *ValidateRC()*.

Akce *Validate()* kontroluje přítomnost klíčových atributů. Formulář je nejprve AJAX funkcí odeslán na server. Zde je použita technologie LINQ k nalezení klíčových elementů dané šablony na disku a tyto atributy jsou vyhledány v příchozích datech z view. Pokud je některá položka prázdná, je zpět odeslána varovná hláška. Je-li vše v pořádku, jsou data převedena do XML a uložena.

Druhá akce slouží k validaci formátu rodného čísla. AJAX metoda zavolá serverovou akci ihned po vyplnění textového pole s označením RC. Ke kontrole je využita funkce z knihovny GDiag, která ověřuje všechny náležitosti rodného čísla. Podle navrácené hodnoty je potom obarveno textové pole.

5.3.3 Převod z HTML do XML

Data z HTML formuláře přichází na server jako textový řetězec, který obsahuje pouze vyplněné údaje. Jednotlivé údaje jsou odděleny symbolem "&" a vždy nesou název elementu a jeho hodnotu. Názvy elementů jsou tvořeny podle XML předlohy a tím pádem je přesně dané, kam v XML šabloně hodnoty patří. Elementy s názvem "checkbox" však v původní šabloně neobsahují atribut *checked*, který nese hodnotu zaškrtnutí a podle níž je potom checkbox vykreslen v prohlížeči. Proto je nejprve ke všem těmto elementům atribut přidán se základní hodnotou *false*. Dalším nezbytným předpokladem je přidat název snímku, ke kterému XML patří, ke zpětnému dohledání. Je tedy přidán atribut *image* a prozatím ponechán prázdný. Poté se prochází jednotlivé hodnoty z formuláře a pomocí LINQ dotazů je k nim dohledán odpovídající element v XML. Na základě typu je potom hodnota vhodně vložena. Například u zaškrtávacího pole je přidána změnou atributu *checked* z *false* na *true*, u rodného čísla je navíc hashována a u datumu změněn formát.

Zároveň je kontrolováno, zda se jedná o klíčovou hodnotu. Tyto hodnoty mají prioritu danou atributem *index*. Z těchto klíčových elementů je poté vytvořen název souboru i původního snímku. Je uložen do atributu *image* pro zpětné dohledání. Je-li klíčový atribut rodné číslo, je v názvu uveden pouze jeho hash. Takto označený snímek i XML soubor jsou uloženy na server a připraveny k exportu.

```
switch (xE.Name.ToString()){
    case "rc":
        string rcHash = FormManip.GetHashString(nameValueSplit[1]);
        XAttribute hash = new XAttribute("hashcode", rcHash);
        xE.Add(hash);
        xE.SetValue(nameValueSplit[1]);
        break;
    case "checkbox":
        xE.Attribute("checked").Value = "true";
        break;
    case "date":
        string[] tempf = nameValueSplit[1].Split('-');
        if (tempf.Count() > 1){
            string temp = tempf[2] + "." + tempf[1] + "." + tempf[0];
            xE.SetValue(temp);
        }
        break;
    case "textfield":
        xE.SetValue(nameValueSplit[1]);
        break;
}
//pokud je klíčový atribut, přidat do klíčových atributů s pořadím index
GetKeyValues(sessionData, exInd, xE);
```

Kód 8 - Zapisování hodnot z HTML formuláře do XML v dle typu elementu

5.4 Export dat

Po kliknutí na tlačítko Export, je zavolána stejnojmenná JavaScriptová funkce. Ta obsahuje metodu *window.location()*, která umožní spustit libovolnou akci ve zvoleném Controlleru bez nutnosti obnovit stránku. Akce je pojmenována *Export()* a vrací archiv zpracovaných souborů ve formátu ZIP, což vyvolá v prohlížeči zobrazení dialogu pro uložení archivu. Pro práci se souborem ZIP je použita knihovna DotNetZip.

```
using (ZipFile zip = new ZipFile())
{
    MemoryStream output = new MemoryStream();
    //funkce GetFilesToZip načte soubory do zip archivu pomocí zip.AddFile()
    GetFilesToZip(zip);
    zip.Save(output);
    output.Seek(0, SeekOrigin.Begin);
    //po uložení vše smaže a nabídne okno pro stažení
    ScanHelper.DeleteAll();
    zip.Dispose();

    return File(output, "application/zip", "GDiag-" + DateTime.UtcNow + ".zip");
}
```

Kód 9 - Archivace do formátu ZIP a odeslání souboru zpět do View

6 Diskuze

GDiag webový scanner je dle mého názoru velmi užitečným nástrojem pro digitalizaci papírových formulářů z funkčních vyšetření v gerontologii. Obzvláště ve spojení se softwarovou sadou vytvořenou na KBI, do které patří také editor na tvorbu formulářů a XML šablon a aplikace pro správu a export, tvoří velmi silnou trojici a komplexní informační systém, provádějící celým životním cyklem formuláře, od jeho tvorby, po možnost vyhodnocování digitalizovaných výsledků.

Výstup této práce je však použitelný do jisté míry také samostatně. Má veškeré předpoklady pro to, aby byl schopný zpracovat libovolný formulář definovaný jeho XML šablonou. Díky této všeobecnosti je použitelný také v jiných institucích pracujících s papírovými dotazníky. Implementace na platformě ASP.NET navíc přináší řadu výhod a s využitelností programovacího jazyku C# jsou dostupné také knihovny pro tento jazyk vytvořené. Další výhodou je zvolená architektura MVC, která oddělením jednotlivých vrstev kódu od sebe zjednodušuje jeho modifikovatelnost.

Desktopový program, ze kterého tato aplikace vychází, měl hlavní nevýhodu v náročnosti na ovládání, dále v nutnosti instalace na každou pracovní stanici zvlášť a ve špatné dostupnosti při aktualizaci. Tato práce měla za cíl vytvořit novější verzi, která tyto nedostatky eliminuje. Prostředí programu bylo vyvíjeno již od začátku s ohledem na cílové uživatele, kteří nemusí mít téměř žádné počítačové dovednosti. Webové řešení přináší dostupnost z jakéhokoliv počítače s nainstalovaným prohlížečem dle minimálních požadavků.

Díky těmto předpokladům si myslím, že by se GDiag webový scanner mohl postupně rozšířit mezi více gerontologických center a zpřístupnit tak cenné informace z funkčních vyšetření k výzkumu a zlepšení kvality péče.

Program nabízí několik funkcí k urychlení digitalizace, které jsou popsány v kapitole Funkční specifikace. Tyto nástroje pomáhají při práci a ubírají na počtu stereotypních operací s dotazníkem, čímž snižují riziko chyby. Někdy je však potřeba zasáhnout ručně a také je nutné automatizované procesy vždy zkontrolovat. V příštích krocích je na základě testování potřeba odladit tyto nástroje k co nejmenší chybovosti.

Dalším vylepšením do budoucna by zcela určitě byla funkce rozpoznávání ručně psaných čísel. V současné chvíli je potřeba některá pole (například celkový počet bodů, datum, věk nebo váhu pacienta) vyplňovat ručně, toto vylepšení by tedy vedlo ke značnému urychlení práce. Bylo by také dobré implementovat přímé propojení exportu s databází GDiag aplikace pro správu a export. V neposlední řadě by velmi pomohlo běhu aplikace a zpříjemnilo její užívání optimalizovat výkon a zrychlit některé operace.

Jelikož algoritmy na zpracování obrazu byly přejaty z desktopového programu Formscan, nebylo cílem práce tyto algoritmy vytvářet nebo optimalizovat. Některé z nich však ještě vyžadují další doladění.

Klíčové pro všechna případná vylepšení a hlavně samotné využívání aplikace, bude zpětné hodnocení cílových uživatelů při nasazení do provozu. V prvních fázích ostrého provozu bude nezbytné sbírat názory uživatelů a na jejich základě pak software optimalizovat.

7 Závěr

Cílem této bakalářské práce bylo vytvořit webovou aplikaci pro digitalizaci papírových formulářů z funkčních vyšetření v gerontologii a export získaných dat. Jednotlivé body ze zadání práce a stanovené požadavky na tento software byly dosaženy za využití moderních technologií. K možnosti vývoje aplikace v prostředí internetu byla použita platforma ASP.NET. V současné době je aplikace laděna a testována na URL adrese http://gdiag.fbmi.cvut.cz/scan. Testována je na již naskenovaných funkčních vyšetřeních. V blízké době by měla být začít využívána Gerontologickým centrem v Praze 8.

Jedním ze záměrů této práce bylo urychlit a usnadnit proces digitalizace. Tomu velmi napomáhá snadný import oskenovaných formulářů do aplikace. Snímky jsou odesílány přímo ze scannerů ve zdravotnických zařízeních přes FTP na server aplikace. Není tedy potřeba je ručně nahrávat a program rovnou zpracovává příchozí soubory a převádí je z původního formátu TIF, který obsahuje několik snímků od jednoho pacienta, na jednotlivé soubory typu JPEG. Uživatel pracující s programem může tyto snímky následně procházet, zpracovávat, mazat a digitalizované formuláře exportovat. Jsou tedy splněny všechny základní požadované funkčnosti. K práci s ofoceným formulářem byly využity některé funkce a metody již existující desktopové aplikace, které se během jejího používání osvědčily.

K urychlení práce přispívají také možnosti automaticky ořezávat snímky podle vyznačených rohů, programově číst čárový kód a přiřazovat dle něho příslušnou šablonu, algoritmus pro rozpoznávání hodnot v zaškrtávacích polích, validace formátu rodného čísla a přítomnosti klíčových atributů, které jsou zároveň předvyplňovány při zpracovávání dalšího souboru. Jelikož funkce ořezávání snímku a přiřazování šablony jsou závislé na kvalitě ofoceného dokumentu, v případě horší kvality obrazu je možné tyto operace provést i ručně.

Dalším požadavkem byl export digitalizovaných dat do centrální aplikace pro správu a export. Ten je v současné době umožněn do souboru ve formátu ZIP, který obsahuje jak data v XML šablonách, tak ořezané snímky pro zpětnou kontrolu. Takto

vytvořený archiv je plně kompatibilní s webovou aplikací pro správu a export a při nahrání je obsah převeden do databáze GDiag, kde je připraven k procházení.

Důležitým cílem bylo také vytvořit systém tak, aby byl snadno ovladatelný. Prostředí aplikace bylo tedy navrženo s ohledem na cílové uživatele. Obsahuje minimální počet ovládacích prvků a nevyžaduje zvláštní počítačové dovednosti. Návod k obsluze je k dispozici přímo v aplikaci.

Díky zvolené vývojové platformě ASP.NET a strukturování kódu dle architektury MVC, byla maximalizována schopnost udržitelnosti aplikace. Tato platforma nabízí řadu výhod, které při implementaci nebyly plně využity a zůstává zde prostor pro zlepšení a další vývoj funkčností softwaru.

GDiag webový scanner je primárně určen pro gerontologická centra, avšak je implementován s ohledem na možnost budoucí modifikace a využití i v jiných institucích. Byl vyvíjen tak, aby byl schopen zpracovat jakékoliv formuláře a XML šablony vytvořené GDiag editorem.

8 Reference

[1] KALVACH, Z. *Geriatrie a gerontologie*. Praha: Grada Publishing, 2004.

[2] MUDR. JURAŠKOVÁ BOŽENA, Ph.D. Funkční geriatrické vyšetření a testy. *Gerontologické partnerství* [online]. b.r. [cit. 2016]. Dostupné z: http://www.gepa.cz/download/Funkcni%20geriatricke%20vysetreni%20a%20testy[1].ppt

[3] ŠTĚPÁNKOVÁ, H, C HÖSCHL a L VIDOVICOVÁ. *Gerontologie: Současné otázky z pohledu biomedicíny a společenských věd*. Charles University in Prague, Karolinum Press, 2015.

[4] GDiag. *Katedra bioinformatiky* [online]. b.r. [cit. 2016]. Dostupné z: http://kbi.fbmi.cvut.cz/cs/gdiag

[5] JANKŮ, Tomáš. *Webová aplikace pro správu a export funkčních vyšetření*. Kladno: ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE Fakulta biomedicínského inženýrství Katedra biomedicínsé informatiky Vedoucí práce Mgr. Radim Krupička, 2013. Bakalářská práce.

[6] *Remark Software* [online]. 2016 [cit. 2016]. Dostupné z: http://remarksoftware.com

[7] *LightComp* [online]. 2016 [cit. 2016]. Dostupné z: http://www.lightcomp.cz/TestsChecker

[8] *FormScanner* [online]. 2015 [cit. 2016]. Dostupné z: http://www.formscanner.org

[9] BERNARD, Borek. Úvod do architektury MVC. *Zdrojak.cz* [online]. 2009 [cit. 2016]. Dostupné z: https://www.zdrojak.cz/clanky/uvod-do-architektury-mvc/

[10] ČÁPKA, David. MVC architektura. *ITnetwork.cz* [online]. b.r. [cit. 2016]. Dostupné z: http://www.itnetwork.cz/navrhove-vzory/mvc-architektura-navrhovy-vzor/

[11] ASP.NET Razor. *W3schools.com* [online]. b.r. [cit. 2016]. Dostupné z: http://www.w3schools.com/aspnet/razor_intro.asp

[12] Use case diagram. *UML a OO, metodologie* [online]. b.r. [cit. 2016]. Dostupné z: http://mpavus.wz.cz/uml/uml-b-use-case-3-2-1.php

[13] ČÁPKA, David. UML - Use Case Diagram. *ITnetwork.cz* [online]. b.r. [cit. 2016]. Dostupné z: http://www.itnetwork.cz/navrhove-vzory/uml/uml-use-case-diagram/

Seznam obrázků

Seznam ukázek kódů

Seznam použitých symbolů a zkratek

ASP ..... Active Server Pages
CPOL ..... Code Project Open License
FTP ..... File Transfer Protocol
GPL ..... General Public License
HTML ..... HyperText Markup Language
IIS ..... Internet Information Services
IS ..... Informační systém
JPEG ..... Joint Photographic Experts Group
LINQ ..... Language Integrated Query
MVC ..... Model view controller
OMR ..... Optical mark recognition
TIF ..... Tagged Image File
UML ..... Unified Modeling Language
XML ..... Extensible Markup Language
XAML ..... Extensible Application Markup Language