FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání diplomové práce

Název:	Webové rozhraní pro zpracování SPAMM tagovaných dat z magnetické rezonance
Student:	Bc. Tomáš Taro
Vedoucí:	Ing. Petr Pauš, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Webové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2022/2023

Pokyny pro vypracování

Cílem práce je vytvoření webového rozhraní pro existující aplikaci, která provádí zpracování SPAMM tagovaných dat získaných pomocí magnetické rezonance (MR). Webové rozhraní by mělo umožnit nahrání dat typu DICOM, jejich zpracování a zobrazení v prohlížeči, zadání parametrů pro algoritmus SPAMM včetně interaktivní úpravy mřížky, spuštění algoritmu a následné zobrazení výsledků.

1) Analyzujte možnosti zpracování medicinských dat ve formátu DICOM pomocí webových technologií.
2) Analyzujte možnosti propojení webového rozhraní s existující aplikací.
3) Analyzujte vhodné frameworky pro tvorbu webového rozhraní.
4) Navrhněte prototyp webové aplikace pro zpracování SPAMM tagovaných dat z MR.
5) Prototyp implementujte.
6) Proveďte vhodné testování na reálných SPAMM tagovaných datech z MR.

Elektronicky schválil/a Ing. Jaroslav Kuchař, Ph.D. dne 26. ledna 2022 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práca

Webové rozhranie pre spracovanie SPAMM tagovaných dát z magnetickej rezonancie

Bc. Tomáš Taro

Katedra softwarového inženýrství

Vedúci práce: Ing. Petr Pauš, Ph.D.

19. apríla 2023

Poďakovanie

Prehlásenie

Prehlasujem, že som predloženú prácu vypracoval(a) samostatne a že som uviedol(uviedla) všetky informačné zdroje v súlade s Metodickým pokynom o etickej príprave vysokoškolských záverečných prác.

Beriem na vedomie, že sa na moju prácu vzťahujú práva a povinnosti vyplývajúce zo zákona č. 121/2000 Sb., autorského zákona, v znení neskorších predpisov, a skutočnosť, že České vysoké učení technické v Praze má právo na uzavrenie licenčnej zmluvy o použití tejto práce ako školského diela podľa § 60 odst. 1 autorského zákona.

V Prahe 19. apríla 2023 . . . . . .

České vysoké učení technické v Praze

Fakulta informačních technologií

Odkaz na túto prácu

Taro, Tomáš. *Webové rozhranie pre spracovanie SPAMM tagovaných dát z magnetickej rezonancie*. Diplomová práca. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2023.

Abstrakt

Klíčová slova

Abstract

Keywords

Obsah

Zoznam obrázkov

Kapitola 1

Úvod

Kapitola 2

Magnetická rezonancia

Magnetická rezonancia (MR) je jedna zo zobrazovacích techník, ktorá je používaná k zobrazeniu vnútorných orgánov tela. Narozdiel od röntgenového žiarenia a počítačovej tomografie (CT), magnetická rezonancia nepoužíva ionizujúce žiarenie. Avšak medzi spoločné znaky týchto troch zobrazovacích techník patrí ich neinvazívnosť a bezbolestné vyšetrenie [1] (vlastný preklad).

Magnetická rezonancia sa používa najmä pri:

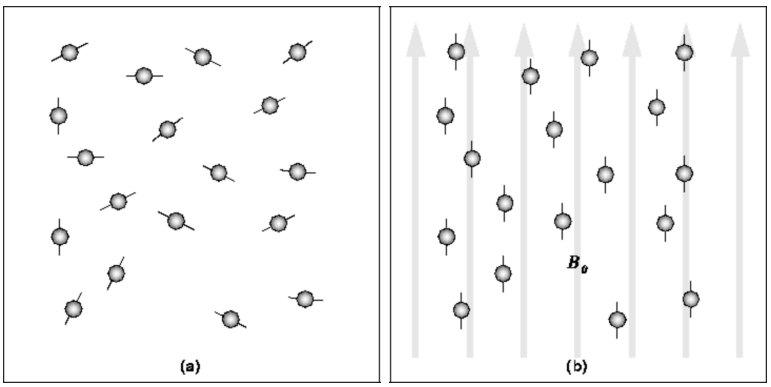
- podozrení na anomálie mozgu a miechy, nádory a cysty,
- poranení kĺbov a mäkkých tkanív,
- podozrení na srdcové problémy,
- rozličných ochoreniach pečene a iných brušných orgánov, atď. [2] (vlastný preklad).

Pred niektorými MR procedúrami sa pacientovi môže intravenózne podať kontrastná látka, ktorá zlepší kontrast a vzájomnú odlíšiteľnosť orgánov a mäkkých tkanív [3].

Bohužiaľ, existujú aj určité kontraindikácie, pri ktorých použitie MR pre daného človeka nie je možné. Jedným z kontraindikácií je implantovaný kardiostimulátor, v prípade že nie je kompatibilný s MR prístrojom. Všeobecne sa za kontraindikáciu považuje použitie akéhokoľvek magnetického materiálu v tele. Taktiež je MR vyšetrenie kontraindikované ženám v prvom trimestri tehotenstva [4].

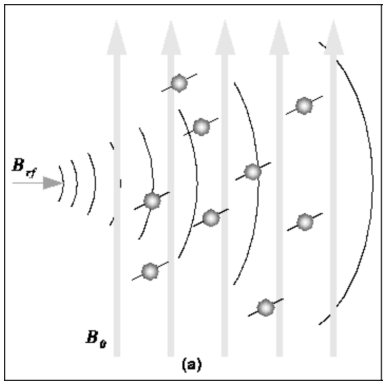
2.1 Princíp magnetickej rezonancie

Princípom magnetickej rezonancie je smerové magnetické pole (moment - $\mathcal{B}_0$) spojené s pohybom voľných jadier vodíku v tele subjektu. Tieto jadrá majú charakteristický pohyb (spin) vytvárajúci malý magnetický moment s určitým smerom (ktorý je náhodný) a veľkosťou. Keď je subjekt umiestnený vo veľkom magnetickom poli (v tubuse MR prístroja), voľné vodíkové jadrá sa zarovnajú v smere $\mathcal{B}_0$ (smer y) a vytvoria magnetický moment $\mathcal{M}$ paralelne k $\mathcal{B}_0$. Vodíkové jadrá začnú náhle prechádzať okolo smeru magnetického poľa ako gyroskopy. Toto správanie sa nazýva Larmorova precesia [1] (vlastný preklad).



Obr. 2.1 Na ľavom obrázku je možné vidieť voľný pohyb vodíkových jadier a na pravom ich zarovnanie v smere $\mathcal{B}_0$ [1].

Následne sa aplikuje rádiofrekvenčný impulz $\mathcal{B}_{rf}$ kolmo na $\mathcal{B}_0$. Tento impulz rovnajúci sa frekvencii Larmorovej precesie spôsobí posun $\mathcal{M}$ od $\mathcal{B}_0$ [1] (vlastný preklad).



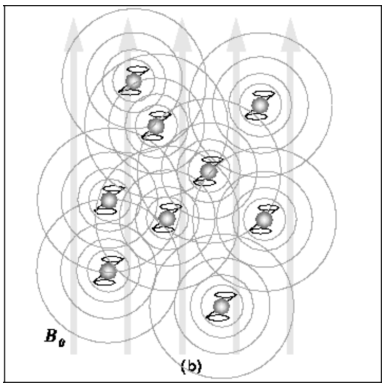
Obr. 2.2 Kolmá aplikácia RF impulzu $\mathcal{B}_{rf}$ na vodíkové jadrá [1].

Frekvenciu Larmorovej precesie (nazývaná ako Larmorova frekvencia), je definovaná nasledovne:

$$\omega_0 = -\gamma * \mathcal{B}_0,$$

kde γ predstavuje gyromagnetický pomer a $\mathcal{B}_0$ intenzitu magnetického poľa. Gyromagnetický pomer je konštanta závislá na jadre danej častice. Pre vodík sa táto konštanta rovná 42.6 MHz/Tesla [1] (vlastný preklad).

Akonáhle prestane pôsobiť rádiofrekvenčný impulz $\mathcal{B}_{rf}$, jadrá vodíka sa presunú naspäť tak, že ich $\mathcal{M}$ je znovu paralelný s $\mathcal{B}_0$. Tento návrat vodíkových jadier sa nazýva relaxácia. Počas nej jadrá strácajú energiu vysielaním ich vlastného rádiofrekvenčného signálu. Tento signál sa nazýva „voľný indukčný rozpad“ – z anglického Free Induction Decay (FID). Ten sa zmeria vodivým poľom MR prístroja za účelom vyhotovenia 3D MR snímku v odtieňoch šedej [1] (vlastný preklad).



Obr. 2.3 Emitovanie FID signálu vodíkovými jadrami [1].

Avšak, na jeho vytvorenie musí byť FID signál enkódovaný pre každý rozmer pomocou frekvenčného a fázového kódovania. Kódovanie v axiálnom smere sa dosiahne pridaním gradientového magnetického poľa $\mathcal{G}_y$ v smere $\mathcal{B}_0$ (v smere y). Po pridaní $\mathcal{G}_y$ sa hodnota Larmorovej frekvencie zmení lineárne v axiálnom smere, tzn. že pre konkrétny axiálny rez existuje konkrétna Larmorova frekvencia, ktorá sa aplikuje vyslaním rádiofrekvenčného impulzu $\mathcal{B}_{rf}$. $\mathcal{G}_y$ sa potom odstráni a ďalší gradient, $\mathcal{G}_x$, sa aplikuje kolmo na $\mathcal{G}_y$. Výsledkom je, že rezonančné frekvencie jadier sa menia v smere x vďaka $\mathcal{G}_x$ a majú fázovú variáciu v smere y v dôsledku predtým aplikovaného $\mathcal{G}_y$. Vzorky v smere x sú

teda kódované frekvenciou a v smere y fázou. 2D inverzná Fourierova transformácia sa následne použije pre transformáciu vzoriek na snímku [1] (vlastný preklad).

Kontrast získanej snímky závisí od nasledujúcich dvoch parametrov:

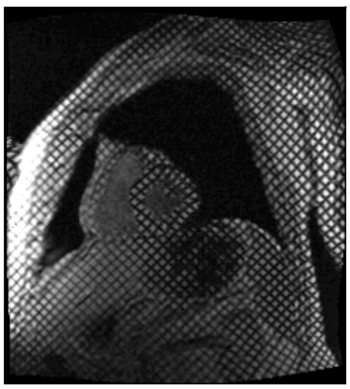
- od času pozdĺžnej relaxácie - T1
- a od času priečnej relaxácie - T2.

Čas T1 je čas potrebný pre jadrá vodíkov k ich relaxácii a čas T2 predstavuje čas za ktorý sa FID signál prechádzajúci cez dané tkanivo rozpadne. Oba časy závisia od daného typu látky nachádzajúcej sa v subjekte [1] (vlastný preklad).

Po získaní MR snímky sa impulz $\mathcal{B}_{rf}$ opakuje vopred stanovenou rýchlosťou. Zmenou sekvencie impulzov ($\mathcal{B}_{rf}$) sa vytvárajú rôzne typy snímkov. Čas opakovania (TR) je množstvo času medzi po sebe nasledujúcimi pulznými sekvenciami aplikovanými na rovnaký rez. Time to Echo (TE) je čas medzi dodaním impulzu $\mathcal{B}_{rf}$ a prijatím odozvy. Úpravou TR je možné meniť výsledný kontrast na snímke medzi rôznymi typmi tkanív [1] (vlastný preklad).

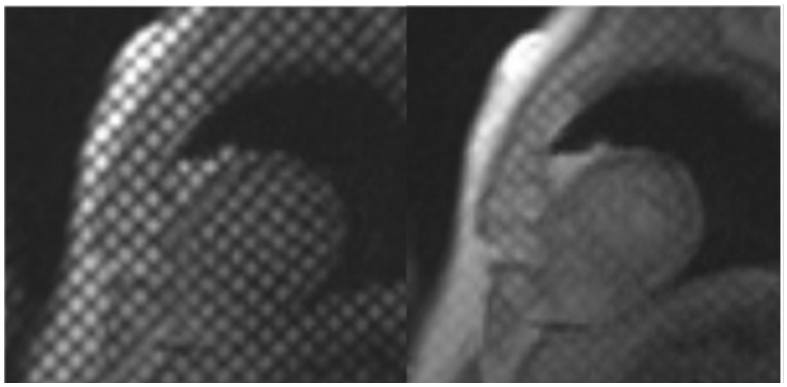
2.2 SPAMM

SPAMM – z anglického (SPAtial Modulation of Magnetization) → „priestorová modulácia magnetizácie“ – je technika používajúca rádiofrekvenčné saturačné impulzy pre umiestnenie mriežky na myokard, za cieľom sledovania jeho pohybu počas srdcového cyklu. V súčasnej praxi sa SPAMM technika používa v situáciách, kde informácia o kontrakcii myokardu je kľúčová, ako napr. podozrenie na ischemickú chorobu srdca alebo abnormality týkajúcej sa neprirodzeného pohybu steny myokardu [5] (vlastný preklad).



Obr. 2.4 Otagovaný snímok myokardu pomocou techniky SPAMM [5].

Nevýhodou použitia tejto techniky je skutočnosť, že táto mriežka sa stráca s blížiacim sa koncom srdcového cyklu. Samotné čiary mriežky sa pri konci systoly (časť srdcového cyklu, počas ktorej sa komory srdca sťahujú po naplnení krvou) môžu zlúčiť alebo úplne vyblednúť, čo sťažuje následnú analýzu pohybu srdca [5] (vlastný preklad).



Obr. 2.5 Ľavý obrázok zobrazuje začiatok systoly, pravý jej koniec.

Kapitola 3

Analýza súčasnej aplikácie

Táto kapitola sa zaoberá účelom súčasnej aplikácie a analýzou použitých technológií, ktoré sú dôležité pre celkovú funkcionalitu aplikácie. Následne sú popísané technológie, ktoré boli použité pri vývoji súčasnej aplikácie až po podprogramy, ktoré riešia výpočtovo náročnejšie úlohy v rámci tejto aplikácie. Jednému z najdôležitejších podprogramov, `grid-tracker`, je venovaná väčšia pozornosť. Táto kapitola sa taktiež venuje následnému zostaveniu a spusteniu aplikácie, ktoré je dôležité pre nasledujúcu sekciu, ktorá analyzuje používateľské rozhranie aplikácie.

3.1 Účel aplikácie

Účelom súčasnej aplikácie – DICOM Viewer – je analýza pohybu srdcového svalu, pomocou ktorej by bolo možné zistiť anomálie v jeho pohybe. Aplikácia umožňuje importovať súbory typu DICOM (popísaného v sekcii 3.2.3), ktoré sú otagované SPAMM mriežkou.

Na importovaných snímkach je následne používateľom vytvorená mriežka, ktorej parametre môžu byť následne používateľom upravené. Štruktúra a parametre týchto mriežok sú neskôr poslané podprogramu `grid-tracker`, ktorého úlohou je zarovnanie používateľom vytvorených mriežok s mriežkami vytvorenými SPAMM technikou. Po ich zarovnaní je možné pomocou techniky grafových rezov odsegmentovať srdečné komory a tým pádom zúžiť analýzu pohybu srdcového svalu len na tieto komory.

Cieľom tejto diplomovej práce je vytvorenie webovej verzie súčasnej aplikácie vrátane zarovnania mriežok pomocou `grid-tracker` podprogramu (3.3).

3.2 Použité technológie

Popisu použitých technológií v súčasnej desktopovej aplikácii sa venuje táto sekcia. Na základe zistenia, aké technológie a aplikačné závislosti aplikácia využíva, bude nakoniec možné aplikáciu zostaviť a vyskúšať. Nasledujúci prehľad použitých technológií je založený na dôslednom preskúmaní zdrojového kódu aplikácie, ktorý sa vyznačoval skoro neexistujúcou dokumentáciou. Bez tejto dokumentácie bola orientácia v zdrojovom kóde náročnejšia, čoho dôsledkom bolo predĺženie času potrebného pre analýzu celkovej architektúry súčasnej aplikácie.

3.2.1 Qt

Súčasná aplikácia bola vyvinutá pomocou Qt[1] – cross-platformového frameworku určeného pre vytváranie aplikácií najmä v jazyku C++.[2] Aplikácie vyvinuté týmto frameworkom majú výhodu v tom, že sú spustiteľné na rôznych operačných systémoch s minimálnym počtom zmien v zdrojovom kóde [6] (vlastný preklad). V súčasnosti (od roku 2014) zastrešuje vývoj tohto frameworku spoločnosť The Qt Company.

Výsadou Qt frameworku je taktiež rozdelenie jeho funkcionality do jednotlivých modulov. Pri následom vytváraní aplikácie sa použijú len také moduly, ktoré sú v danej aplikácii potrebné [6] (vlastný preklad).

Existujúca aplikácia využíva tento framework vo verzii 5.15, ktorá sa vyznačuje tým, že je to verzia s dlhodobou podporou. Koniec podpory tejto verzie je naplánovaný na 26.5.2023. Najnovšia verzia Qt frameworku je momentálne verzia 6.5, ktorá je zároveň verziou s dlhodobou podporou.

Čo sa týka súčasnej desktopovej aplikácie, v nej boli použité nasledovné moduly:

- Qt Core
- Qt Widgets
- Qt GUI
- Qt Test

Qt Core[3] modul obsahuje najpoužívanejšie triedy ako napr. `QCoreApplication`, `QObject`, `QDebug` a iné. Nakoľko sú tieto triedy používané aj inými modulmi,

[1] https://www.qt.io/product/qt6
[2] https://isocpp.org
[3] https://doc.qt.io/qt-5/qtcore-index.html

je tento modul implicitne nalinkovaný Qt frameworkom pri budovaní aplikácie [7] (vlastný preklad).

Qt Widgets[4] modul poskytuje UI elementy, ktoré sú určené pre vytváranie používateľského rozhrania. Tieto elementy môžu zobrazovať rozličné dáta, prijímať vstup z klávesnice, byť štylizované a zoskupované do rozličných usporiadaní [8] (vlastný preklad). Existujúca aplikácia používa z modulu napr. triedu `QMainWindow`, ktorá je zodpovedná za vykreslenie aplikačného okna a taktiež triedu `QGraphicsScene`, ktorá je zodpovedná za vykreslenie snímkov z magnetickej rezonancie v DICOM formáte.

Qt GUI[5] modul obsahuje triedy určené pre zobrazovanie aplikačného okna a iného grafického obsahu s následnou obsluhou udalostí. Taktiež obsahuje triedy, ktoré sú zodpovedné za zobrazovanie 2D grafiky, fontov a typografie [9] (vlastný preklad). Súčasná aplikácia z tohto modulu používa napr. triedu `QImage`, ktorá obsahuje metódy pre priamy prístup k pixelom snímkov a ich manipuláciu.

Qt Test[6] modul poskytuje rozličné triedy pre jednotkové testovanie Qt aplikácií a príslušných knižníc [10] (vlastný preklad) – v súčasnej aplikácii bol tento modul využitý pri testovaní grafického používateľského rozhrania a funkcionalít súčasnej aplikácie, ako napr. testovanie zmien v nastaveniach vykreslenej mriežky na obrázku z magnetickej rezonancie.

3.2.2 Qmake

Pre zjednodušenie písania Makefilov, ktoré definujú, ako má byť program skompilovaný, bol použitý nástroj Qmake.[7] Tento nástroj pochádza taktiež z dielne The Qt Company. Qmake umožňuje vývojárom definovať vytváranie rozličných Makefilov pre daný program pomocou syntaxu definovaného programom Qmake [11] (vlastný preklad). Výsledkom tohto procesu je súbor s príponou `.pro` obsahujúci inštrukcie, ako daný Makefile vytvoriť. Následne sa pomocou príkazu `qmake` s argumentom cesty k `.pro` súboru vytvorí `Makefile` súbor, pomocou ktorého je možné daný program skompilovať, čoho výsledkom je spustiteľný súbor aplikácie.

Pre súčasnú aplikáciu sa daný súbor volá `Cameo.pro` a nachádza sa v adresári `dicomViewer`.

[4] https://doc.qt.io/qt-5/qtwidgets-index.html
[5] https://doc.qt.io/qt-5/qtgui-index.html
[6] https://doc.qt.io/qt-5/qttest-index.html
[7] https://doc.qt.io/qt-5/qmake-manual.html

3.2.3 DICOM

V súčasnosti sú snímky získané pomocou zobrazovacích techník v medicíne zväčša ukladané v archivačnom a komunikačnom systéme snímkov. Tento systém ukladá nielen snímkové dáta ale aj iné relevantné dáta k týmto snímkam podľa štandardu známom ako DICOM[8] (Digital Imaging and Communications in Medicine) [12] (vlastný preklad). Ten je medzinárodným štandardom pre komunikáciu a manažment informácií o medicínskych obrazových a k nim príslušných dátach. Definuje, ako majú byť takéto dáta spracovávané, ukladané, tlačené a prenášané medzi zariadeniami podporujúcimi príjem týchto dát [13] (vlastný preklad).

Začiatok vývoja DICOM štandardu sa datuje k prelomu 80. a 90. rokoch 20. storočia, kedy započala spolupráca medzi American College of Radiology a National Electrical Manufacturers Association. NEMA taktiež vlastní autorské práva k tomuto štandadu. Momentálne sa DICOM skladá z 22 nezávislých častí, z ktorých avšak nie všetky musia byť implementované daným zariadením podporujúcim tento štandard [14] (vlastný preklad).

Pre účely spracovania snímkových dát, DICOM štandard vo svojej 10. časti definuje dátovú štruktúru (formát) súboru, do ktorého sa tieto dáta ukladajú. Dátová štruktúra súboru, ktorý spĺňa podmienky 10. časti tohto štandaru, býva značená ako „DICOM Part 10“ súbor, inak známy ako DICOM súbor [12] (vlastný preklad).

Štruktúra tohto (binárneho) súboru je nasledovná – prvých 128 bajtov býva zväčša prázdnych (vyplnených 0). Ďalšie 4 bajty obsahujú uložený reťazec „DICM“. Na základe týchto bajtov sa dá určiť, či sa jedná alebo nejedná o DICOM súbor. Ďalej nasleduje hlavička, ktorá je rozdelená na viacero skupín zoskupujúcich súvisiace atribúty. Konkrétne atribúty sa adresujú tagom - ten sa skladá z 8 čísiel v hexadecimálnom formáte. Prvé 4 čísla reprezentujú skupinu, v ktorej sa daný atribút nachádza a posledné 4 čísla jednoznačne identifikujú konkrétny atribút v skupine [12] (vlastný preklad).

Ako príklad bude uvedené získanie informácie o pacientovom veku - všetky informácie o pacientovi sa nachádzajú v skupine 0010. Pacientov vek v tejto skupine sa nachádza na pozícii 1010, tým pádom výsledný tag pod ktorým nájdeme vek pacienta je (0010, 1010). Ku každému tagu je jednoznačne priradená reprezentácia jej hodnoty, ktorý určuje dátový typ, formát a dĺžku hodnoty daného atribútu [12] (vlastný preklad).

Po hlavičke nasleduje skupina 7FE0, ktorá už obsahuje dáta o samotných obrazových pixeloch [12] (vlastný preklad). Typ kódovania týchto dát určuje

[8] https://www.dicomstandard.org

Transfer Syntax – ten udáva, akým spôsobom sú obrazové pixely zakódované. Transfer Syntax obsahuje taktiež informáciu, v akom poradí bajtov sú informácie zakódované (Little Endian vs Big Endian) a aká kompresia obrazových dát bola použitá [15] (vlastný preklad).

3.2.4 DCMTK

DCMTK[9] je knižnica, ktorá implementujú veľkú časť DICOM štandardu v jazykoch C a C++. Úlohou tejto knižnice je okrem iného skúmanie, vytváranie a konverzia DICOM súborov, manipulácia s pamäťovými médiami a odosielanie, resp. prijímanie obrazových súborov cez internetové pripojenie.

Za jej vývojom stojí nemecká firma OFFIS, ktorá túto knižnicu vyvíja nepretržite už od roku 1993. V súčasnosti sa používa v nemocniciach a rôznych spoločnostiach po celom svete, kde predstavuje softvérový základ pri rozličných výskumných projektoch, prototypoch a komerčných produktoch nevynímajúc [16] (vlastný preklad).

Súčasná aplikácia je kompatibilná s najnovšou verziou tejto knižnice, ktorou je verzia 3.6.7. DCMTK knižnica je v tejto aplikácii použitá pre získanie informácií z hlavičiek DICOM súborov, ako napr:

- údaje o pacientovi,
- údaje o snímku a
- údaje o sérií snímkov.

3.2.5 OpenMP

OpenMP[10] poskytuje rozhranie pre programovanie aplikácií (tzv. API), vďaka ktorému je možné vytvárať C,[11] C++ a Fortran[12] aplikácie využívajúce viac vlákien nad zdieľanou pamäťou. Vývoj OpenMP v súčasnosti zastrešuje OpenMP Architecture Review Board.

OpenMP funguje na báze direktív, pomocou ktorých sa jednotlivé časti programu dajú paralelizovať viacerými spôsobmi – a to paralelizáciou prevádzania jednotlivých úloh (tzv. funkčný paralelizmus – vhodný pre paralelizáciu rekurzívnych algoritmov, kde úloha = volanie funkcie) alebo paralelizáciou dátovo nezávislých for loop iterácií (tu sa jedná o tzv. iteračný dátový paralelizmus) [17].

[9] https://dicom.offis.de/en/dcmtk/dcmtk-tools/
[10] https://www.openmp.org
[11] https://www.open-std.org/jtc1/sc22/wg14/
[12] https://fortran-lang.org/en/

Existujúca aplikácia využíva direktívy OpenMP pre paralelizáciu výpočetne náročnejších algoritmov. Súčasná verzia OpenMP, verzia 5.2, je plne kompatibilná s aktuálnym zdrojovým kódom aplikácie.

3.2.6 TNL

Template Numerical Library[13] je numerická knižnica, ktorá poskytuje rozličné dátové štruktúry, ktoré uľahčujú prácu s pamäťou a vývoj efektívnych numerických riešičov. Táto knižnica je implementovaná pomocou C++ s cieľom poskytnúť flexibilné a užívateľsky prívetivé rozhranie. TNL poskytuje natívnu podporu pre moderné hardwarové architektúry ako sú viacjadrové CPU, GPU a distribuované systémy, ktoré je možné spravovať cez jednotné rozhranie [18].

Vývoj TNL knižnice od jej počiatku riadi Tomáš Oberhuber z Katedry matematiky na FJFI ČVUT v spolupráci s Jakubom Klinkovským a Alešom Wodeckim.

V súčasnej aplikácii sú z TNL knižnice použité kolekcie ako napr. `String` pre manipuláciu s reťazcami a `Containers::Array`, `Containers::{Vector, StaticVector,MultiVector}`, čo sú šablóny pre reprezentáciu n-dimenzionálnych polí, ktoré abstrahujú manažment dát a exekúciu bežných operácií na rozličných hardvérových architektúrach.

Aplikácia z TNL knižnice taktiež používa `Solvers::ODE::Merson` – jedná sa o Runge-Kutta-Merson metódu štvrtého rádu s adaptívnym výberom časového kroku, pomocou ktorej vieme získať približné riešenie diferenciálnych rovníc.

Bohužiaľ, súčasná aplikácia nie je kompatibilná s najnovšími zdrojovými kódmi TNL knižnice – pre nájdenie posledného „dobrého stavu“ knižnice, t.j. stavu za ktorého bolo možné aplikáciu skompilovať, bol využitý nástroj `git bisect`. Tento nástroj našiel ako posledný „dobrý stav“ knižnice z 13.5.2021. Pri využití TNL knižnice zostavenej po tomto dátume nebolo možné súčasnú aplikáciu skompilovať.

3.3 Pomocné podprogramy

Súčasná aplikácia obsahuje a využíva nasledujúce 3 podprogramy:

- grid-tracker
- local-variance

[13] https://tnl-project.org

- graph-cuts

`grid-tracker` C++ podprogram určený pre sledovanie pohybu myokardu pomocou detekcie SPAMM mriežky z DICOM snímku a mriežky vytvorenej používateľom. Mriežku vytvorenú používateľom sa snaží zarovnať so SPAMM mriežkou pre každú snímku zo sekvencie snímkov. Výstupom sú upravené koordináty bodov mriežky vytvorenej používateľom – tie sa následne aplikujú namiesto doterajšej mriežky, čoho výsledkom je zobrazenie mriežky s upravenými koordinátmi bodov.

Vstupom tohto programu sú nasledujúce parametre:

- `inputImageFileNames` – vektor reťazcov reprezentujúce cesty k súborom multivektorov (definované TNL knižnicou), ktoré obsahujú enkódované DICOM snímky,
- `inputGridFileNames` – vektor reťazcov reprezentujúce cesty k súborom používateľom definovaných mriežok uložených ako TNL textový multivektor,
- `outputGridFileNames` (nepovinné)– vektor reťazcov reprezentujúce cesty k súborom spracovaných mriežok, ktoré budú uložené ako textový TNL multivektor,
- `curvatureCoefficient` – závislosť vývoja mriežky od zakrivenia, vyššie číslo znamená vyššiu závislosť,
- `forceCoefficient` – závislosť mriežky od gradientu obrazu, vyššie číslo znamená vyššiu závislosť,
- a `stopTime` – časový interval algoritmického výpočtu.

Pre daný výpočet `grid-tracker` využíva technológie TNL a OpenMP.

Jeho výstupom sú predchádzajúce používateľom definované mriežky v textovom TNL multivectore upravené algoritmom tak, aby mriežky zodpovedali mriežkam definovanými SPAMM metódou.

Nasledujúce dva C++ podprogramy nie sú dôležité pre túto diplomovú prácu, avšak pre úplnosť je ich účel vysvetlený.

Úlohou `local-variance` podprogramu je aplikovanie filtra lokálnej variancie pre danú snímku za účelom neskoršej lepšej segmentácie srdcových komôr. Tento filter je implentovaný pomocou jednoduchého rozptylového filtru, ktorý vypočíta priemernú intenzitu pixelov vo štvorcovom okolí každého

pixelu a túto hodnotu dosadí do výberového rozptylu, ktorý sa bude rovnať novej hodnote intenzity pixelu [19].

Podprogram `graph-cuts` slúži pre segmentáciu srdcových komôr, ktorá vychádza z predpokladu, že hranica medzi objektom a jeho pozadím sa nachádza v miestach nekonzistencie susedných pixelov snímky. Implementovaná metóda, ktorá dosiahla dobré výsledky bola metóda grafových rezov, kombinujúca Fordov-Fulkersonovým algoritmom s Preflow-Push algoritmom [19].

3.4 Zostavenie aplikácie a jej spustenie

Pre potrebu popísania používateľského rozhrania je najprv žiaduce dosiahnuť zostavenie existujúcej aplikácie a jej následné spustenie. Po počiatočnej analýze zdrojového kódu, v ktorom bolo zistené, že aplikácia bola vyvíjaná pre Linuxové prostredie, bol vo virtuálnom prostredí nainštalovaný operačný systém Ubuntu 22.10.[14]

Pre zostavenie súčasnej aplikácie bude potrebné nainštalovať Qt framework spolu s nástrojom Qmake, nakoľko ich architektúra súčasnej aplikácia vyžaduje.

Pre inštaláciu Qmake nástroja je potrebné nainštalovať balíček `qt5-default`.[15] Ten obsahuje nie len nástroj `qmake` ale aj aplikáciu Qt Creator.[16]

Qt Creator je aplikácia, ktorá poskytuje prostredie pre integrovaný vývoj aplikácií postavených nad Qt frameworkom. Pomocou tejto aplikácie je možné vyvíjať, testovať, zostavovať, spúšťať a debugovať aplikácie postavené na tomto frameworku. Táto aplikácia bude neskôr využitá pre neskorší debugging súčasnej aplikácie.

Okrem balíčku `qt5-default` je taktiež potrebné nainštalovať balíčky `cmake`[17] a `build-essential`,[18] čo sú balíčky poskytujúce ďalšie nástroje potrebné pre úspešné zostavenie aplikácie. Tieto balíčky už môžu byť v niektorých prípadoch súčasťou použitej linuxovej distribúcie, čo ale neplatilo v prípade použitia Ubuntu 22.10 ako operačného systému.

Pretože súčasná aplikácia využíva DCMTK knižnicu, ako bolo popísané v sekcii 3.2.4, je tiež nutné nainštalovať nasledujúce balíčky: `dcmtk`[19]

[14] https://ubuntu.com
[15] sudo apt install qt5-default
[16] https://doc.qt.io/qtcreator/
[17] sudo apt install cmake
[18] sudo apt install build-essential
[19] sudo apt install dcmtk

a `libdcmtk-dev`.[20]

Prvý z uvedených balíčkov nainštaluje samotnú DCMTK knižnicu, druhý obsahuje knižnice a hlavičkové súbory pre vývoj aplikácií používajúce túto knižnicu. Nakoľko v `Dicom.pro` súbore (ktorý je určený pre výsledné zostavenie aplikácie) sa nachádzajú cesty ku knižniciam z balíčku `libdcmtk-dev`, je aj tento balíček nutný nainštalovať pre neskoršie korektné spustenie súčasnej aplikácie.

Keďže `grid-tracker` podprogram obsahuje OpenMP direktívy pre paralelizáciu výpočetných algoritmov, je potrebné nainštalovať OpenMP prostredníctvom balíčku `libomp-dev`.[21] TNL knižnica bude taktiež benefitovať z inštalácie tohto balíčka, nakoľko sama využíva OpenMP pre paralelizáciu algoritmov, čo znamená, že sa tento balíček nainštaluje ešte pred samotnou inštaláciou TNL knižnice.

Ďalej je potrebné nainštalovať samotnú TNL knižnicu – pre jej inštaláciu bude potrebné stiahnuť zdrojové kódy buď formou zabaleného zdrojového kódu v .zip balíčku, alebo naklonovaním repozitára pomocou programu `git`.[22] Keďže aplikácia je nekompatibilná s najnovšou verziou TNL knižnice, je potrebné stiahnuť alebo naklonovať repozitár so zdrojovými kódmi do dátumu 13.5.2021 (viď 3.2.6).

Nakoľko je nutné samotnú knižnicu zo zdrojových kódov zostaviť, je nevyhnutné mať nainštalovaný kompilátor podporovaný samotnou knižnicou. Medzi podporované kompilátory sa radí GCC[23] kompilátor vo verzii 8.0 a vyššie alebo Clang[24] vo verzii 7.0 a vyššie. Prvý z nich je možné nainštalovať pomocou balíčku `gcc`[25] a druhý pomocou balíčku `clang`.[26]

Následne je potrebné exekuovať inštalačný skript `install`, ktorého účelom je nakonfigurovanie TNL knižnice a jej zostavenia. Pred exekuovaním samotného inštalačného skriptu `install` je potrebné zistiť, či sú nainštalované balíčky `doxygen`,[27] `matplotlib`[28] a `graphviz`.[29] Tieto balíčky sa využívajú pri generovaní dokumentácie počas exekúcie `install` skriptu (pri defaultnej inštalácii). Ak sa niektorý z daných balíčkov nenachádza v systéme, je potrebné

[20] sudo apt install libdcmtk-dev
[21] sudo apt install libomp-dev
[22] https://git-scm.com
[23] http://gcc.gnu.org
[24] https://clang.llvm.org
[25] sudo apt install gcc
[26] sudo apt install clang
[27] https://doxygen.nl/index.html
[28] https://matplotlib.org
[29] https://graphviz.org

ho doinštalovať.

Po nainštalovaní všetkých potrebných balíčkov nastal čas pre zostavenie TNL knižnice spustením `install`[30] skriptu. Po jeho ukončení sa hlavičkové súbory TNL knižnice budú nachádzať v priečinku `/usr/lib/aarch64-linux-gnu`. Názov posledného priečinku sa môže líšiť, nakoľko je závislý na architektúre CPU, na ktorom sa TNL knižnice zostavuje. V tomto prípade bola knižnica zostavovaná na systéme Ubuntu 22.10 bežiacom nad CPU s architektúrou Aarch64 (Apple M1 Pro CPU).

Ďalším krokom pre úspešné zostavenie aplikácie je definovanie cesty k hlavičkovým súborom TNL knižnice v samotnom `Cameo.pro` súbore, určenom pre zostavenie aplikácie pomocou programu `qmake`. Po otvorení súboru `Cameo.pro` je teda potrebné nastaviť hodnotu premennej DCMTK_LIBS – v tomto prípade ju bolo potrebné nastaviť na `/usr/lib/aarch64-linux-gnu` a následne daný súbor uložiť.

Po prevedení všetkých predchádzajúcich krokoch je možné spustiť nástroj `qmake` pomocou krokov popísaných v sekcii 3.2.2. Jeho výsledkom bude `Makefile` súbor, ktorý definuje kroky, ako zostaviť súčasnú aplikáciu. V tomto momente je už možné súčasnú aplikáciu skompilovať pomocou spustenia príkazu `make install`.

Po skompilovaní aplikácie sa v rovnakom priečinku objaví spustiteľný súbor `Cameo`, ktorému je potrebné nastaviť práva pre spustenie príkazom `chmod +x Cameo`. Následne je možné spustiť aplikáciu príkazom `./Cameo`.

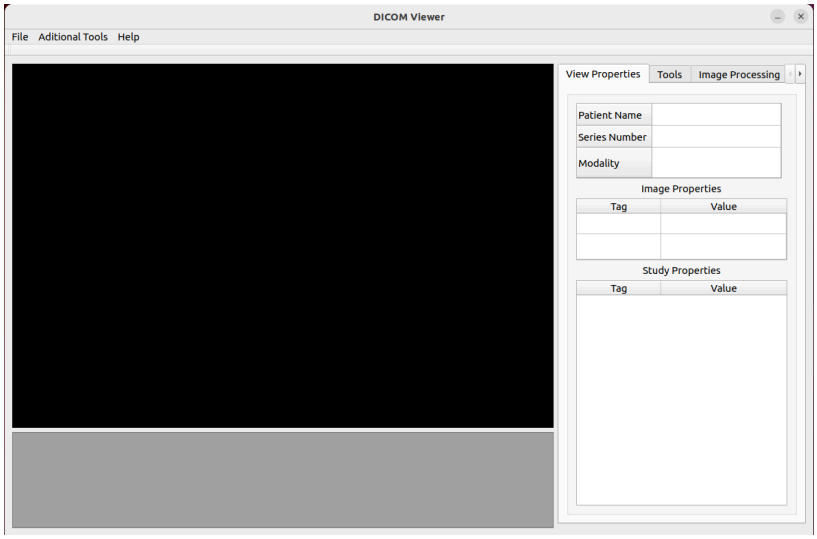
[30] chmod +x install && ./install

3.5 Používateľské rozhranie

Po úspešnom spustení aplikácie sa používateľovi zobrazí hlavné okno aplikácie. V aplikačnom okne sa taktiež zobrazí menu s nasledovnými možnosťami:

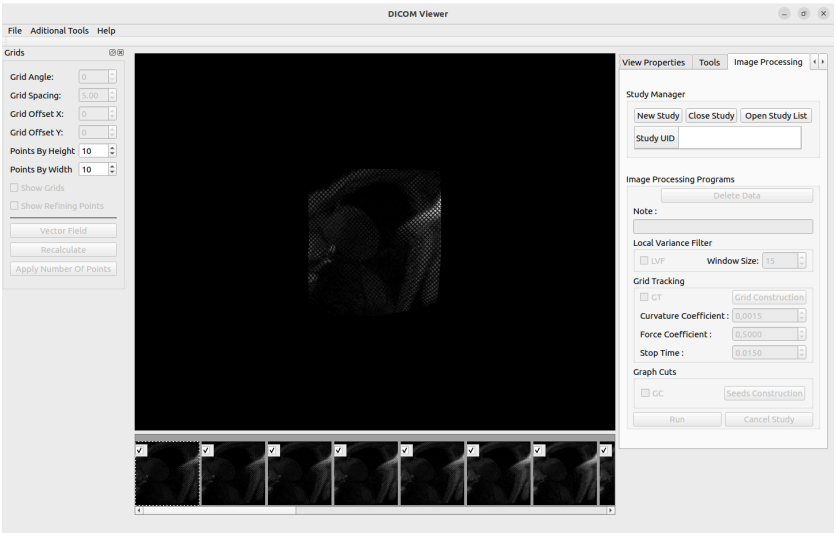
1. File
    a) Open – otvorí systémové okno pre výber priečinku s DICOM snímkami, ktoré sa majú importovať do aplikácie
    b) Save Image
        i. Save View – uloží pohľad na aktuálnu snímku vo zvolenom formáte
        ii. Save Scene – uloží scénu vo zvolenom formáte
        iii. Save Area of Interes [sic] – uloží plochu záujmu vo zvolenom formáte
    c) Save all selected
        i. Save View – uloží pohľad vybraných snímky vo zvolenom formáte
        ii. Save Scene – uloží scénu vybraných snímkov vo zvolenom formáte
        iii. Save Area of Interes [sic] – uloží plochu záujmu vybraných snímkov vo zvolenom formáte
    d) Exit – ukončí aplikáciu
2. Aditional [sic] Tools
    a) Grid Tools – otvorí ľavý postranný panel aplikácie s nastavením mriežky
    b) Lvf Tools – otvorí ľavý postranný panel aplikácie s nastavením filtru lokálnej variácie
    c) Graph Cuts Tools – otvorí ľavý postranný panel aplikácie s nastavením grafových rezov
3. Help
    a) About – zobrazí informácie o DICOM Viewer aplikácii

Po spustení sa aplikácia nachádza v stave, v ktorom nie je možné s ňou interagovať – inými slovami, je najprv potrebné do aplikácie importovať snímky v DICOM formáte, ako je ukázané na nasledujúcom obrázku.



Obr. 3.1 Ukážka DICOM Viewer aplikácie po spustení

To sa docieli pomocou zvolenia možnosti File → *Open* z aplikačného menu. Následne sa otvorí systémové okno pre výber priečinka s DICOM snímkami, ktoré sa majú zobraziť v aplikácii. Bohužiaľ, nie je možné zvoliť snímky jednotlivo z priečinku, čo zapríčiní načítanie všetkých snímok z daného priečinku do aplikácie. Po zvolení priečinku sa zobrazí prvý snímok v aplikácii. Po výbere ľubovolnej možnosti, ktoré ponúka Aditional [sic] Tools menu, sa taktiež zobrazí ľavý postranný panel v aplikácii, ako je možné vidieť na snímke aplikácie nižšie. V tomto prípade bola zvolená možnosť „Grid Tools“.

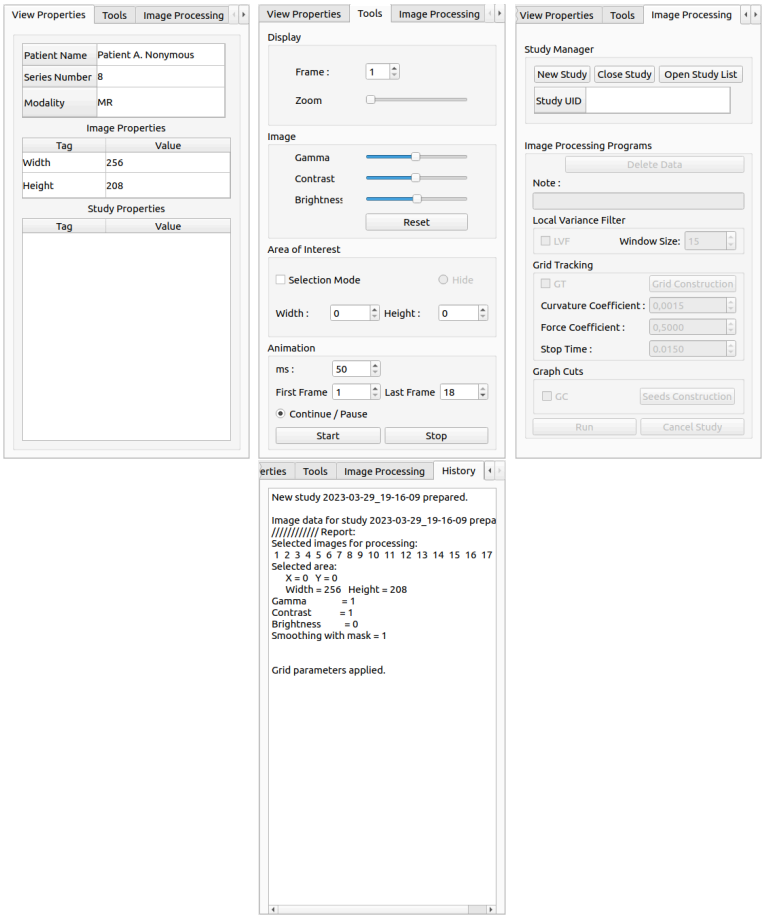


Obr. 3.2 Zobrazenie prvej snímky v DICOM Viewer aplikácii

Po zobrazení ľavého postranného panelu sa odhalí celková štruktúra používateľského rozhrania aplikácie. Tú môžeme rozdeliť na ľavý a pravý postranný panel, medzi ktorými sa nachádza čierna plocha zobrazujúca vybranú snímku. Pod touto snímkou sú zobrazené náhľady všetkých snímok.

Obsahom čiernej plochy, ktorá je dominantná v zobrazení aplikácie, je aktuálne vybraná DICOM snímka. Nad ňou môže byť tiež vykreslená používateľom definovaná mriežka. Náhľady všetkých snímkov, ktoré sa zobrazujú pod aktuálne vybranou snímkou, obsahujú taktiež zaškrtávacie políčko. Toto políčko reprezentuje možnosť, či má byť daný snímok spracovaný v rámci daných 3 podprogramoch.

Obsah pravého postranného panelu je rozdelený na nasledovné karty: „View Properties“, „Tools“, „Image Processing“ a „History“.



Obr. 3.3 Zobrazenie obsahu kariet na pravom postrannom paneli

„View Properties“ karta nie je interaktívna – zobrazuje informácie ako meno subjektu so zobrazenou snímkou, číslo série a modalitu snímkov. Taktiež je zobrazená výška a šírka aktuálne zobrazeného snímku.

Narozdiel od predchádzajúcej karty, „Tools“ karta obsahuje interaktívne prvky, ako napr. zobrazenie a zmena indexu zobrazeného snímku, či slider pre priblíženie snímku. Samotnému snímku je možné tiež zmeniť kontrast, jas a gammu pomocou sliderov. Nasleduje sekcia pre nastavenie oblasti záujmu, pri ktorej je možné nastaviť jej zobrazenie alebo zmeniť jej výšku/šírku – oblasť záujmu je na základe týchto nastavení vykreslená nad snímkou. Keďže aplikácia podporuje prehrávanie snímkov ako animáciu, aplikácia ponúka nastavenie jej rýchlosti v milisekundách (čo predstavuje čas, ktorý ubehne zobrazením jednej snímky), nastavenie počiatočnej snímky, od ktorej sa animácia spustí, a poslednej snímky, po ktorú animácia bude spustená. Okrem nastavenia parametrov animácie nesmú chýbať tlačidlá pre samotné spustenie a zastavenie animácie.

Na ďalšej karte „Image Processing“ sa nachádzajú tlačidlá ovládajúce manažéra štúdií. Tento „manažér“ zoskupuje rôzne štúdie, pod ktorými sa ukladajú rôzne parametre aplikácie. Po kliknutí na tlačidlo „Open Study List“ sa zobrazí nové okno so všetkými štúdiami a ich parametrami. Pre interakciu s ostatnými poliami na tejto karte je potrebné najprv vytvoriť novú štúdiu kliknutím na tlačidlo „New Study“.
Štúdiu je tiež možné ukončiť zvolením tlačidla „Close Study“. Každá štúdia je reprezentovaná jedinečným ID, ktoré sa skladá z dátumu a času jej vytvorenia. Vytvorením novej štúdie sa aktivujú polia „Image Processing Programs“ sekcie. Táto sekcia ponúka tri hlavné začiarkavacie políčka – prvé reprezentuje spustenie aplikovania filtra lokálnej variancie. Druhé z nich reprezentuje spustenie algoritmu pre zarovnanie vygenerovanej mriežky s mriežkou myokardu vytvorenou pomocou SPAMM techniky a tretie spustenie algoritmu segmentácie srdečných komôr pomocou grafových rezov. Zaškrtnutím daného políčka a kliknutím na tlačidlo „Run“ sa algoritmus príslušný danému políčku spustí. Pre „Grid Tracking“ algoritmus je v tejto sekcii možné definovať tri parametre, a to „Curvature Coefficient“, „Force Coefficient“ a „Stop Time“ (viď 3.3).

Účelom poslednej karty „History“ je výpis rozličných záznamov pre informovanie používateľa o prebiehajúcich krokoch aplikácie.

Obsah ľavého postranného panelu sa mení v závislosti na zvolenej možnosti z menu Aditional [sic] Tools. Momentálne sa zobrazuje obsah ľavého panelu po zvolení možnosti „Grid Tools“. V paneli je možné nájsť nasledujúce nastavenia:

- Grid Angle – nastavuje uhol mriežky,
- Grid Spacing – nastavuje rozpätie jednotlivých bodov,
- Grid Offset X – x pozícia od ľavého horného bodu,
- Grid Offset Y – invertovaná y pozícia od ľavého horného bodu,
- Points by Height – udáva počet bodov na úsečku mriežky na výšku,
- Points By Width – udáva počet bodov na úsečku na šírku,
- Show Grids – indikuje, či má byť zobrazená mriežka,
- Show Refining Points – indikuje, či majú byť zobrazené body, ktoré upresňujú pozíciu mriežky,
- Vector Field – spočíta rozdiel v pohybe mriežky medzi predchádzajúcou a aktuálnou snímkou,
- Recalculate – odošle dáta `grid-tracker` podprogramu pre opätovné zarovnanie mriežky voči SPAMM mriežke,
- a Apply Number of Points – uloží mriežku ako textový TNL multivektor.

Ostatné dve možnosti z menu „Aditional [sic] Tools“ nie je potrebné pre účely tejto práce popisovať.

Kapitola 4

Analýza a návrh webovej aplikácie

popísať kapitolu

4.1 Analýza požiadaviek

Táto sekcia sa venuje analýze požiadaviek, ktoré sa delia na dve hlavné kategórie. Týmito kategóriami sú funkčné a nefunkčné požiadavky. Na základe realizovania týchto požiadaviek bude možné implementovať novú webovú aplikáciu pre potrebu analýzy srdcového myokardu.

4.1.1 Funkčné požiadavky

Funkčné požiadavky sú požiadavky vymedzujúce rozsah funkcionality, ktorá by mala byť v danej aplikácii implementovaná.

4.1.1.1 FR1 – Spracovanie a zobrazenie MR snímkov

Do aplikácie by malo byť možné importovať snímky z magnetickej rezonancie vo formáte DICOM a tieto snímky taktiež zobraziť.

4.1.1.2 FR2 – Animácia MR snímkov

Aplikácia by mala umožniť animovať importované snímky pre jednoduchšiu analýzu pohybu myokardu. Parametre animácie ako jej rýchlosť a výber fotky, od/do ktorej snímky má animácia prebiehať by mali byť upraviteľné, napr. pomocou číselného vstupu.

4.1.1.3 FR3 – Zobrazenie a interaktívna úprava mriežky

Implementovaná aplikácia by mala vedieť zobraziť mriežku nad snímkou z MR, ktorá by sa mala dať vygenerovať tlačidlom v používateľskom rozhraní.

Mriežka by taktiež mala byť interaktívna, t.j. polohu jej bodov by malo byť možné interaktívne upravovať, najlepšie pomocou potiahnutím bodu myšou. Parametre mriežky (3.5) by sa taktiež mali dať upraviť podľa želania používateľa a ich zmena by mala byť ihneď viditeľná.

4.1.1.4 FR4 – Zadanie parametrov pre `grid-tracker` podprogram

Pre korektné spustenie `grid-tracker` podprogramu pre zarovnanie mriežky je nutné tomuto podprogramu podsunúť rozličné parametre. Tieto parametre by sa mali dať definovať v aplikácií pre ich neskoršie použitie v tomto podprograme. Výpis týchto parametrov je možné nájsť v 3.3.

4.1.1.5 FR5 – Spustenie `grid-tracker` podprogramu a zobrazenie jeho výsledkov

Aplikácia by mala umožniť spustiť `grid-tracker` podprogram, ktorý zarovná mriežku definovanú používateľom s mriežkou, ktorá bola vygenerovaná pomocou techniky SPAMM. Po jej zarovnaní by mala byť aplikácia schopná vykresliť upravenú mriežku.

4.1.2 Nefunkčné požiadavky

Požiadavky tohto typu síce nevymedzujú rozsah funkcionality danej aplikácie, avšak umožňujú určiť isté obmedzenia pre novú aplikáciu, ako napr. dôraz na podobu výslednej architektúry aplikácie.

4.1.2.1 NF1 – Webová aplikácia

Prvou nefunkčnou požiadavkou je vytvorenie webovej aplikácie, ktorá by mala byť prístupná zo všetkých moderných webových prehliadačov. Pre lekárov výber tejto architektúry zjednoduší jej prístupnosť, nakoľko k takejto aplikácii bude možné pristupovať z rôznych zariadení a platforiem bez nutnosti inštalácie aplikácie a jej následnej podpory na týchto zariadeniach.

4.1.2.2 NF2 – Používateľské rozhranie

Pre interakciu s aplikáciou je nutné navrhnúť a implementovať používateľské rozhranie, pomocou ktorého lekári budú môcť s aplikáciou interagovať. Lekári by preferovali používateľské rozhranie podobné iným aplikáciám z tejto oblasti.

4.1.2.3 NF3 – Ochrana pred únikom dát o pacientovi

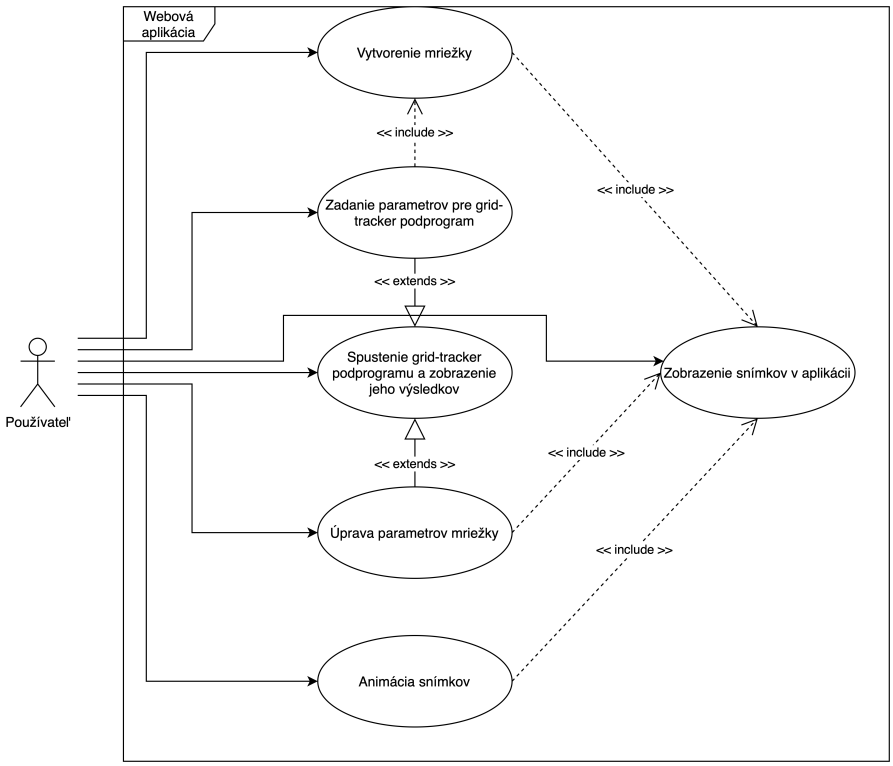
Práca s osobnými dátami by mala byť do maximálnej možnej miere naprieč aplikáciou minimalizovaná, aby sa predišlo únikom citlivých údajov o pacientovi. Týka sa to najmä práce s DICOM súbormi, nakoľko tie obsahujú citlivé dáta o pacientovi.

4.2 Používateľské role

V aplikácii sa bude nachádzať len aktér – používateľ, rovnako ako v súčasnej aplikácii. Tomuto aktérovi by mala byť aplikácia sprístupnená bez rôznych funkčných obmedzení.

4.3 Prípady použitia

Nasledujúce prípady použitia reprezentujú rôzne činnosti, ktoré môže používateľ s aplikáciou vykonávať. Tieto prípady použitia sú popísané pomocou scenárov, ktoré taktiež vychádzajú z funkčných požiadaviek kladených na novú aplikáciu. Pre lepšiu predstavu sú prípady použitia taktiež znázornené graficky pomocou Use Case diagramu nižšie.



Obr. 4.1 Use case diagram

4.3.1 UC1 – Zobrazenie snímkov v aplikácii

Zobrazenie snímkov magnetickej rezonancie v DICOM formáte je jedným z esenciálnych funkčných požiadaviek – „FR1 – Spracovanie a zobrazenie MR snímkov“. Nasledovný scenár túto požiadavku realizuje.

Scenár:

1. Používateľ klikne na jedno z tlačidiel pre import DICOM snímkov do aplikácie.
2. Prehliadač zobrazí systémové okno, v ktorom si používateľ vyberie snímky, ktoré by chcel mať zobrazené v aplikácii.
3. Následne potvrdí import želaných snímkov kliknutím na tlačidlo „Otvoriť“.
4. Aplikácia automaticky vykreslí prvý importovaný snímok a taktiež zobrazí náhľady ostatných importovaných snímkov.
5. V prípade, že sa medzi zvolenými snímkami nachádza súbor, ktorý nekorešponduje so štruktúrou DICOM súboru, aplikácia zobrazí notifikáciu o neúspešnom zobrazení snímky.

4.3.2 UC2 – Animácia snímkov

Nasledovný scenár realizuje funkciu prehrania série snímkov ako animáciu, ako bolo popísané vo funkčnej požiadavke „FR2 – Animácia MR snímkov“. Okrem iného taktiež zahŕňa prípad „UC1 – Zobrazenie snímkov v aplikácii“.

Scenár:

1. UC1 – Zobrazenie snímkov v aplikácii.
2. Kliknutím na tlačidlo reprezentujúce štart animácie sa spustí animácia importovaných snímkov.
3. Kliknutím na tlačidlo reprezentujúce koniec animácie sa animácia skončí.

Alternatívny scenár:

2. Používateľ si nastaví rýchlosť animácie, index snímku, od ktorého má animácia začínať alebo index snímku, ktorým má animácia končiť.

3. Kliknutím na tlačidlo reprezentujúce štart animácie sa spustí animácia importovaných snímkov.

4. Kliknutím na tlačidlo reprezentujúce koniec animácie sa animácia skončí.

4.3.3 UC3 – Vytvorenie mriežky

Vytvorenie mriežky nad snímkou z MR je potrebné pre účely analýzu pohybu myokardu. Nasledujúci scenár čiastočne realizuje funkčnú požiadavku – „FR3 – Zobrazenie a interaktívna úprava mriežky“. Taktiež zahŕňa prípad použitia „UC1 – Zobrazenie snímkov v aplikácii“.

Scenár:

1. UC1 – Zobrazenie snímkov v aplikácii.
2. Používateľ klikne na tlačidlo „Create grid“.
3. Aplikácia zobrazí výzvu pre kliknutie na oblasť snímky, kde má byť mriežka vytvorená.
4. Používateľ klikne na oblasť snímky, kde chce vytvoriť mriežku.
5. Aplikácia vygeneruje mriežku s predvolenými nastaveniami a zobrazí ju.

4.3.4 UC4 – Úprava parametrov mriežky

Medzi prípady použitia patrí aj úprava parametrov mriežky určenej pre analýzu pohybu srdcového svalu. Nakoľko je najprv potrebné mať mriežku pred jej úpravou vytvorenú, zahŕňa nasledovný scenár aj jej vytvorenie. Ten taktiež čiastočne realizuje funkčnú požiadavku „FR3 – Zobrazenie a interaktívna úprava mriežky“.

Scenár:

1. UC3 – Vytvorenie mriežky.
2. Používateľ upraví jeden alebo viacero parametrov uvedených v 3.5.
3. Aplikácia následne automaticky vykreslí mriežku na základe upravených parametrov.

4.3.5 UC5 – Zadanie parametrov pre `grid-tracker` podprogram

Pre spustenie algoritmu zodpovedného pre posun mriežky vytvorenej používateľom voči mriežke vygenerovanej SPAMM technikou je potrebné tomuto algoritmu poslať tri parametre definované v 3.5. Nasledujúci scenár tento prípad použitia realizuje spolu s funkčnou požiadavkou – „FR4 – Zadanie parametrov pre `grid-tracker` podprogram“.

Scenár:

1. UC1 – Zobrazenie snímkov v aplikácii.
2. Používateľ zadá číselné hodnoty parametrov „Curvature coefficient“, „Force coefficient“ a „Stop time“.

4.3.6 UC6 – Spustenie `grid-tracker` podprogramu a zobrazenie jeho výsledku

Spustenie podprogramu a zobrazenie jeho výsledku (súradnice bodov mriežok) vyžaduje nielen mať importované DICOM snímky v aplikácii, ale aj vytvorenú mriežku s upravenými parametrami a zadanými parametrami pre `grid-tracker` podprogram. To je dôvodom, prečo tento scenár použitia zahŕňa prípady „UC1 – Zobrazenie snímkov v aplikácii“, „UC3 – Vytvorenie mriežky“, „UC4 – Úprava parametrov mriežky“ a „UC5 – Zadanie parametrov pre `grid-tracker` podprogram“. Samotný scenár realizuje funkčnú požiadavku „FR5 – Spustenie `grid-tracker` podprogramu a zobrazenie jeho výsledkov“.

Scenár:

1. UC1 – Zobrazenie snímkov v aplikácii.
2. UC3 – Vytvorenie mriežky.
3. UC4 – Úprava parametrov mriežky.
4. UC5 – Zadanie parametrov pre `grid-tracker` podprogram.
5. Používateľ kliknutím na tlačidlo „Compute“ spustí výpočet pomocou `grid-tracker` podprogramu.
6. Po dokončení výpočtu aplikácia zobrazí mriežky upravené horeuvedeným podprogramom.

4.4 Návrh architektúry webovej aplikácie

Medzi prvými krokmi pred vývojom webovej aplikácie patrí analýza všetkých prípustných možností architektúry navrhovanej aplikácie. V tomto prípade návrh architektúry závisí najmä na prepojení webového rozhrania so súčasnou aplikáciou, a preto je potrebné najprv zanalyzovať všetky dostupné možnosti tohto prepojenia. Po porovnaní dostupných možností je nutné zvoliť jednu z nich. Následne bude možné pokračovať s analýzou technológií, ktoré budú použité pre vývoj aplikácie.

Čo sa týka prepojenia súčasnej aplikácie, resp. výpočetného podprogramu `grid-tracker` s novou webovou aplikáciou, existujú dve možnosti, ako môže daná integrácia prebehnúť. Prvou možnosťou je využitie tzv. C++ addons[31] technológie, druhou možnosťou sa naskytuje využiť relatívne novú technológiu – WebAssembly.[32]

[31] https://nodejs.org/docs/latest-v18.x/api/addons.html
[32] https://webassembly.org

4.4.1 C++ addons

C++ addons je technológia, ktorá poskytuje rozhranie medzi C/C++ knižnicami a JavaScriptom.[33] Táto technológia je implementovaná v rámci Node.js,[34] čo je runtime prostredie JavaScriptu, ktoré bude popísané v samostatnej sekcii. C++ addons umožňuje pristupovať k natívnym API operačného systému a taktiež pomáha integrovať C/C++ knižnice tretích strán pre ich priame použitie v Node.js. Doporučeným spôsobom písania takýchto addonov je pomocou technológie Node-API,[35] vďaka ktorej je možné vytvoriť Node-API addon v jazyku C. Pre písanie Node-API addonov v C++ je ale potrebné použiť modul `node-addon-api`,[36] nakoľko ten obsahuje hlavičkové súbory v C++ [20] (vlastný preklad).

Výhodou použitia Node-API technológie je jej nemennosť v rámci rôznych verzií Node.js, čo zaručuje použitie skompilovaného addonu v rôznych verziách Node.js bez nutnosti jeho prekompilovania pre rozličné verzie Node.js [20] (vlastný preklad).

Nástrojom pre zostavenie takéhoto modulu je build systém `node-gyp`.[37] Ten používa `binding.gyp` súbor, ktorý špecifikuje konfiguráciu zostavenia modulu. Táto konfigurácia zahŕňa okrem iného aj cestu k zdrojovým `.cpp` a `.h` súborom. Tie musia byť pred zostavením upravené tak, aby používali Node-API rozhranie [20] (vlastný preklad).

Tento krok zahŕňa vytvorenie metód, ktoré budú prijímať vstupné a vracať výstupné argumenty pretypované na Node-API typy.

Pred zostavením addonu je potrebné vygenerovať Makefile pre cieľový operačný systém pomocou príkazu `node-gyp configure`. Pre zostavenie addonu je následne potrebné exekuovať príkaz `node-gyp build`. Ten skompiluje želané súbory špecifikované v `binding.gyp` do jediného súboru s príponou `.node`. Ten je následne možné importovať ako modul do iného JavaScript modulu pomocou kľúčového slova `import`.[38] Po jeho importovaní je možné volať jeho metódy rovnakým spôsobom ako iné JS metódy [20] (vlastný preklad).

4.4.2 WebAssembly

WebAssembly je nový typ jazyku, ktorý je možné exekuovať vo všetkých moderných webových prehliadačoch. Jeho hlavnou výsadou je zvýšenie rýchlosti

[33] https://developer.mozilla.org/en-US/docs/Web/JavaScript
[34] https://nodejs.org/en
[35] https://nodejs.org/docs/latest-v18.x/api/n-api.html
[36] https://github.com/nodejs/node-addon-api
[37] https://github.com/nodejs/node-gyp
[38] https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/import

exekuovania kódu oproti JavaScriptu blížiaci sa k skoro natívnej exekúcii kódu naprogramovaného v jazykoch C, C++ alebo Rust[39] a iné. Je navrhnutý tak, aby mohol fungovať spoločne s JavaScriptom [21] (vlastný preklad).

Webová platforma sa dá vo všeobecnosti rozdeliť na dve časti:

- VM, pomocou ktorej sa exekuuje kód webovej aplikácie a
- webové API, ktoré je poskytnuté vývojárom pre kontrolu rozličných funkcionalít webového prehliadača, resp. zariadenia [21] (vlastný preklad).

Historicky, VM umožňovala načítať len kód napísaný v JavaScripte. Avšak postupom času sa ukázalo, že JS nie je určený pre aplikácie, ktoré potrebujú väčší výpočetný výkon, ako sú napr. 3D hry, VR/AR, editácia videa či obrázkov a iné. WebAssembly bolo navrhnutý tak, aby tieto problémy vyriešilo a prinieslo prostriedky pre vývoj takýchto aplikácií.

Pomocou WebAssembly JS API[40] je možné načítať WebAssembly moduly – čo sú moduly v binárnom formáte – do JavaScript aplikácie a zdieľať s touto aplikáciou funkcionalitu poskytovanú týmito modulmi [21] (vlastný preklad).

Možností, ako daný modul vytvoriť, je viacero:

- portovať C/C++ aplikáciu pomocou Emscripten[41] technológie,
- písať priamo vo WebAssembly,
- napísať aplikáciu v inom jazyku a kompilovať ju pomocou kompilátora podporujúci WebAssembly výstup, alebo
- použiť AssemblyScript,[42] ktorý je podobný TypeScript[43] jazyku a dá sa priamo skompilovať do WebAssembly [21] (vlastný preklad).

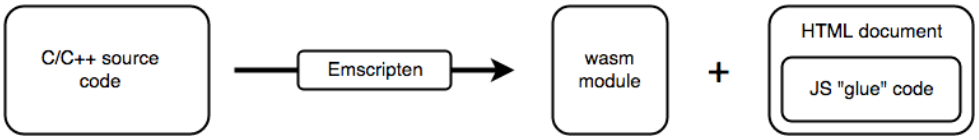
Nakoľko sa v tomto prípade jedná o C++ aplikáciu, bude bližšie analyzovaná prvá možnosť zo všetkých dostupných možností.

Najprv je potrebné stiahnuť a nainštalovať Emscripten kompilátor. Ten umožňuje skompilovať program v C/C++ do modulu vo formáte `.wasm`. Následne je možné daný C++ program skompilovať pomocou príkazu `em++ sample.cpp -o sample.html`. Výstupom tohto príkazu sú tri súbory –

[39] https://www.rust-lang.org
[40] https://developer.mozilla.org/en-US/docs/WebAssembly/Using_the_JavaScript_API
[41] https://emscripten.org
[42] https://www.assemblyscript.org
[43] https://www.typescriptlang.org

`a.out.js`, `a.out.wasm` a `sample.html`. Prvý zo súborov tvorí JS kód, ktorého úloha je prepojiť daný WASM modul (druhý súbor) s JS prostredím. Následne je možné tento modul spolu s JS kódom importovať do HTML[44] stránky, na ktorej daný modul pobeží. Takto importovaný kód v HTML stránke je možné vidieť v súbore `sample.html` [22] (vlastný preklad).

Celý proces je pre ilustráciu zobrazený na obrázku nižšie:



Obr. 4.2 Od zdrojového kódu k .wasm modulu [23]

4.4.3 Výsledná voľba prepojenia

C++ addons technológia je implementovaná v Node.js, čo by v prípade zvolenia tejto technológie znamenalo zvolenie architektúry klient-server. Klientom by bol v tomto prípade webový prehliadač, ktorý by posielal HTTP požiadavku serveru s potrebnými dátami pre `grid-tracker` podprogram. Po výpočte na strane servera by server poslal odpoveď s informáciami o bodoch a úsečkách, ktoré by mali byť vykreslené na všetkých importovaných snímkoch magnetickej rezonancie.

V prípade použitia technológie WebAssembly by všetky výpočetné operácie mohli byť implementované na úrovni klienta. Tým pádom by nebolo nutné posielať žiadne dáta serveru, čo hrá v prospech bezpečnosti. Avšak, ako z analýzy `grid-tracker` podprogramu vyplýva, `grid-tracker` a knižnica TNL pre zrýchlenie výpočetných algoritmov používajú OpenMP technológiu. Bohužiaľ, WebAssembly túto technológiu nepodporuje, čo by v tomto prípade znamenalo, že by celý výpočet musel prebiehať jednovláknovo alebo byť refaktorovaný, aby používal viacero vlákien pomocou technológie WebAssembly threads [24] (vlastný preklad).

Čo sa týka C++ addons, tá OpenMP technológiu podporuje. Taktiež by bol v rámci tejto technológie nutný menší zásah do zdrojového kódu `grid-tracker` podprogramu, nakoľko by stačilo vytvoriť jednu wrapper funkciu v C++, ktorá by bola zodpovedná za konverziu dát do potrebného formátu a taktiež za vrátenie výsledku. Doterajší kód by mohol zostať prakticky nezmenený, resp. s minimálnymi zmenami. Taktiež nie je možné spoľahnúť sa na to, že by webové prehliadače, na ktorých by bežala nová webová aplikácia, podporovala WebAssembly technológiu. Rovnakým spôsobom je možné

[44] https://developer.mozilla.org/en-US/docs/Web/HTML

argumentovať ohľadom výkonu zariadení, na ktorých by daný výpočetný algoritmus bežal, v prípade WebAssembly.

Na základe týchto dôvodov bude lepšou voľbou vybrať technológiu C++ addons, s ktorou by mala byť implementácia prepojenia `grid-tracker` podprogramu a webovej aplikácie nielen rýchlejšia, ale aj s podporou OpenMP technológie. Výpočty v rámci `grid-tracker` podprogramu by taktiež neboli závislé na dostupných výpočetných prostriedkov klienta ale serveru, čo umožňuje mať väčšiu kontrolu nad potrebným škálovaním výkonu pre `grid-tracker` podprogram.

4.5 Technológie pre vývoj webovej aplikácie

Po analýze, ako bude webová aplikácia prepojená s `grid-tracker` podprogramom, je nutné zvoliť potrebné technológie, ktoré budú použité pri vývoji webovej aplikácie. V rámci tejto sekcii nebudú popísané žiadne konkrétne balíčky a iné závislosti.

4.5.1 HTML5

Pre definovanie štruktúry webového dokumentu a jeho významu bude potrebné použiť značkovací jazyk HTML. Tento jazyk pozostáva zo série značiek (elementov) a k nim príslušných atribútov, pomocou ktorých je možné vytvorený obsah anotovať a významovo ho definovať. Týmto spôsobom je možné vytvoriť nadpisy, odstavce textu, číselné i nečíselné zoznamy, či importovať obrázky alebo sprostredkovať audio/video, atď.

Takto štruktúrovaný obsah definovaný pomocou jazyka HTML je možné zobraziť v ľubovoľnom webovom prehliadači podporujúcom tento jazyk. Webový prehliadač takýto dokument zanalyzuje a na základe použitých značiek vykreslí. Každá značka má definovaný predvolený štýl zobrazenia, ktorý sa môže líšiť od prehliadača k prehliadaču.

Nižšie je uvedený príklad základnej štruktúry HTML5 webového dokumentu:

```
<!doctype html5>
<html>
  <head></head>
  <body>
    <p>Hello world!</p>
  </body>
</html>
```

Značka `<!doctype>` definuje verziu použitého HTML dokumentu, čo je v tomto prípade HTML5. Ďalej nasleduje značka `<html>`, ktorej úloha je zoskupiť elementy `<head>` a `<body>`. V elemente `<head>` sa zvyčajne nachádzajú metadáta ako názov dokumentu, špecifikácia ďalších zdrojov pre načítanie v dokumente a iné. Na druhú stranu, element `<body>` zoskupuje obsah dokumentu, ktorý je zobrazený prehliadačom.

Prvá verzia tohto jazyka bola definovaná v roku 1993 samotným vynálezcom WWW, Timom Berners-Leeom. Momentálne najnovšou verziou HTML jazyka je tzv. HTML5 Living Standard,[45] vyvíjaný pracovnou skupinou WHATWG.[46] Najnovší štandard priniesol viacero nových značiek ako napr. značku `<audio>` pre prehrávanie audia, `<video>` pre prehrávanie videa, či `<picture>`, ktorá je určená pre definovanie viacero zdrojov pre zobrazený obrázok. HTML5 štandard okrem značiek poskytuje niekoľko API, ktoré sú implementované vo webových prehliadačoch. Pomocou týchto API je možné napr. geolokalizovať používateľa pomocou HTML5 Geolocation API alebo vykreslovať grafiku použitím HTML5 Canvas API, a iné.

4.5.2 CSS 3

CSS[47] – z anglického Cascading Style Sheets – je jazyk popisujúci vzhľad použitých HTML5 elementov vo webovom dokumente. Tento jazyk definuje súbor pravidiel, ktoré môžu byť aplikované na jednotlivé elementy webového dokumentu, na základe ktorých sa mení vzhľad pravidlami ovplyvnených elementov.

Samotné pravidlo sa skladá zo selektora, ktorý definuje rozsah elementov, ktoré budú ovplyvnené. Nasleduje zoznam vlastností s ich hodnotami, ktoré majú byť aplikované na samotný selektor. Týmto spôsobom je možné definovať vzhľad nielen jedného, ale aj viacerých elementov vo webovom dokumente pomocou jedného pravidla.

Nižšie je uvedený príklad pravidla, ktoré mení farbu textu vo všetkých elementoch `p` (`p` definuje odstavec textu) na červenú:

```
p {
    color: red;
}
```

[45] https://html.spec.whatwg.org
[46] https://whatwg.org/
[47] https://developer.mozilla.org/en-US/docs/Web/CSS

Zoskupené pravidlá sa väčšinou ukladajú do samostatného súboru s príponou `.css`. Tento súbor je následne nalinkovaný do HTML5 dokumentu pomocou značky `link`, ktorú prehliadač pri parsovaní dokumentu prečíta a následne aplikuje.

Definovanie samotného selektoru môže byť pre dané pravidlo sofistikovanejšie než ako bolo ukázané v príklade vyššie. Element môže byť špecifikovaný na základe jeho rôznych atribútov, ako ID, zoznam tried, či hodnotou jeho atribútu, atď.

Čo sa týka verzií CSS jazyka, u CSS sa nepoužívajú verzie ale tzv. levely. Prvým levelom bol CSS Level 1, ktorý sa stal odporúčanou špecifikáciou W3C konzorcia[48] v 1996. Tento level bol základom pre nasledujúce levely tohto jazyka. V súčasnosti najnovší level CSS jazyka je CSS Level 3, v ktorom sa narozdiel od predchádzajúcich levelov jednotlivé časti jazyka delia na moduly, z ktorých každý môže mať level vyšší než samotná špecifikácia. Z tohto dôvodu sa taktiež rozhodlo, že samotný level CSS jazyka sa už nebude zvyšovať [25] (vlastný preklad).

4.5.3 JavaScript

JavaScript je cross-platformový skriptovací programovací jazyk a treťou základnou technológiou pre vývoj webových stránok a aplikácií, spolu s HTML a CSS. Používa sa pre implementovanie funkcionality, kde kombinácia HTML a CSS nie je pre daný účel vhodná alebo určená, ako napr. dynamická interakcia používateľa s webovou stránkou/aplikáciou, riešenie rôznych výpočetných úloh, odosielanie dát na server a prijímanie odpovede, a iné.

Samotný jazyk bol vytvorený Brendanom Eichom, pracujúcom vo firme Netscape, ktorá taktiež vyvíjala webový prehliadač – Netscape Navigator. JavaScript bol zahrnutý už vo verzii 2.0 tohto prehliadača, ktorý bol vydaný v roku 1995. Následne sa začal objavovať aj v iných prehliadačoch – napr. vo všetkých prehliadačoch vytvorených firmou Microsoft počnúc Internet Explorerom 3.0 [26] (vlastný preklad).

JavaScript nasleduje ECMAScript špecifikáciu,[49] vytvorenú Ecma International organizáciou. Tá kontinuálne vydáva každý rok novú štandardizovanú ECMAScript špecifikáciu, ktorá slúži ako predloha pre vytvorenie všeobecného skriptovacieho jazyka. JavaScript je v tomto prípade jazyk spĺňajúci tento štandard, nakoľko sa ním riadi a implementuje ho. Momentálne najnovšou verziou štandardu je jeho 14. edícia, ktorá pridala najmä nové metódy pracujúce s poľom (`Array`).

[48] https://www.w3.org
[49] https://tc39.es/ecma262/

Čo sa týka vlastností samotného jazyka, JavaScript je dynamicky typovaným jazykom, čo znamená, že pri vytváraní premenných sa nedefinuje ich typ. To umožňuje do rovnakej premennej na jednom mieste uložiť číslo, a na inom zase reťazec. Taktiež sa jedná objektovo-orientovaný jazyk, kde dedičnosť je riešená mechanizmom prototypov, kde metódy a vlastnosti môžu byť za behu pridané do akéhokoľvek objektu. Taktiež používa primárne jedno hlavné vlákno pre všetky svoje operácie, avšak je možné vytvoriť tzv. pracovné vlána pomocou Web Workers technológie. Nakoľko JavaScript podporuje OOP, imperatívny a deklaratívny štýl písania kódu, jedná sa o tzv. multi-paradigmový jazyk [26] (vlastný preklad).

Samotný kód je potrebné uložiť do súboru s koncovkou `.js`, aby bol rozpoznaný prehliadačom ako súbor obsahujúci JavaScript kód.

4.5.3.1 TypeScript

Pri písaní kódu v JavaScripte sa často môže stať, že programátor napíše nevalidný kód, avšak IDE ho žiadnym spôsobom na túto skutočnosť neupozorní. Táto situácia vzniká najmä kvôli tomu, že sa v JavaScript kóde nevyskytujú žiadne informácie o typoch premenných, argumentov funkcií a iné. Pri väčšom codebase je následne väčšia šanca, že bude obsahovať chyby, na ktoré by mohli byť vývojári upozornení samotným IDE ešte počas vývoja tohto kódu.

Problém s neexistujúcimi typmi je možné vyriešiť pomocou písania kódu v TypeScript jazyku vyvíjanom od jeho počiatku (2012) firmou Microsoft.[50] Ten pridáva priamo do kódu podporu pre typy premenných, vstupných a výstupných argumentov funkcií či návratových hodnôt funkcií. TS je nadmnožinou JavaScriptu, čo znamená, že validný JavaScript kód je taktiež validným TypeScript kódom. Taktiež je garantované, že JavaScript kód prevedený na TypeScript nezmení správanie kódu, čo znamená pre vývojárov jednoduchšiu migráciu z JavaScriptu do TypeScriptu [27] (vlastný preklad). TypeScript v princípe funguje ako statický analyzátor kódu, ktorý analyzuje validitu napísaného kódu. V prípade, že kód nie je validný, alebo obsahuje chyby, TypeScript pomocou IDE upozorní vývojára na túto skutočnosť. Pre samotnú analýzu kódu je nutné písať kód do súborov s koncovkou `.ts`.

Nižšie je uvedený príklad funkcie, ktorej argument má typ `Number`. V prípade, že by bol argument nekompatibilného typu ako v uvedenom príklade, TS compiler pomocou IDE informuje vývojára o tejto skutočnosti. V prípade použitia JavaScriptu by IDE o tomto probléme vývojára neinformovalo.

```
1 function divideByThree(a: number): number {
```

[50] https://www.microsoft.com

```
2   return a / 3;
3 }
4
5 divideByThree('a');
6 // TS compiler v tomto prípade zobrazí na r. 5 nasledujúcu chybu:
7 // The left-hand side of an arithmetic operation
8 // must be of type 'any', 'number', 'bigint' or an enum type.
```

Nakoľko nie je možné TypeScript použiť priamo vo webovom prehliadači, je potrebné takýto kód konvertovať (transpilovať) do JavaScriptu. Podporu tejto funkcionality pridali vývojári TypeScriptu pomocou konzolového programu `tsc` [27] (vlastný preklad). Jeho vstupom sú TypeScript súbory, ktoré majú byť transformované do JavaScriptu. Výstupom sú už súbory v JavaScripte.

Počas transpilácie TypeScript kódu do JavaScriptu dochádza k vymazaniu všetkých informácií o typoch a iných konštruktoch nepodporovaných JavaScriptom, tak aby výsledný kód bol validným JavaScript kódom [27] (vlastný preklad).

Pre horeuvedené výhody tohto jazyka bude TypeScript použitý pri vývoji webovej aplikácie.

4.5.4 Node.js

Node.js je asynchrónne cross-platform runtime prostredie JavaScriptu, ktoré umožňuje vývojárom exekuovať JavaScript kód mimo prehliadača. Pomocou neho je možné vyvíjať nielen konzolové aplikácie, ale aj budovať škálovateľné webové služby. Node.js je open-source projektom a jeho vývoj zastrešuje nadácia OpenJS Foundation [28] (vlastný preklad). Podobne ako JavaScript v prehliadači, Node.js používa jedno hlavné vlákno pre exekúciu kódu. Súčasne je nad Node.js vyvíjaných mnoho backendových frameworkov, z ktorých najznámejšie sú napr. koa[51] alebo express.[52] Pomocou týchto frameworkov je možné vytvoriť API endpointy, pomocou ktorých vie klient komunikovať so serverom.

Tieto frameworky (a iné aplikácie postavené nad Node.js) sú poskytované formou balíčkov. Tieto balíčky zvyknú byť dostupné v registri balíčkov pre Node.js – npm. Pomocou rovnomenného terminálového programu je možné (globálne i lokálne) nainštalovať rôzne balíčky, ktoré môžu byť použité pri vývoji aplikácií. Pre inštaláciu balíčka stačí exekuovať príkaz `npm install`

[51] https://koajs.com/
[52] https://expressjs.com/

`packagename`. Okrem inštalácií balíčkov je možné tieto balíčky spravovať, aktualizovať na novšie verzie či odinštalovať, a iné. Počas inštalácie Node.js je automaticky nainštalovaný aj tento balíčkový manažér.

Okrem iného je vďaka Node.js možné používať len jeden programovací jazyk na vývoj oboch častí webových aplikácii – frontendu a backendu. Tento fakt predstavuje odpadnutie nutnosti ovládať ďalší programovací jazyk pre vývoj webových aplikácií.

Nakoľko je možné vyvíjať webovú aplikáciu v jednom jazyku, čo prináša komfort pre samotného vývojára, bude Node.js použitý na strane backendu pre komunikáciu s frontendom webovej aplikácie.

4.5.5 Docker

Docker je platforma určená pre vývoj a distribúciu aplikácií. Pomocou tejto platformy je možné separovať aplikáciu od infraštruktúry, čo umožňuje zrýchliť distribúciu softvéru. Docker poskytuje možnosť zabaliť aplikáciu a spustiť ju v izolovanom prostredí nazvanom „kontajner“. Tie sú vytvárané tak, aby obsahovali len nástroje potrebné pre beh aplikácie spolu s jej konfiguráciou, čo zrýchluje inicializáciu a spustenie kontajnerov. Kontajnery sú od seba predvolene izolované, avšak je možné dodatočne nastaviť sieťový interface pre komunikáciu medzi nimi [29] (vlastný preklad).

Vytvorenie kontajneru prebieha z objektu nazývanom „Image“. Image je inými slovami šablóna, ktorá definuje, ako má byť vytvorená zabalená aplikácia. Častokrát obsahuje príkazy, ktoré majú za úlohu nainštalovať aplikačné závislosti, nastaviť dodatočné bezpečnostné vlastnosti či otvoriť porty, cez ktoré je možné s danou aplikáciou komunikovať. Tieto príkazy sú následne uložené do súboru zvanom `Dockerfile`. Z jedného Docker Imagu je možné spustiť viacero kontajnerov, čím sa stáva škálovanie aplikácie ešte jednoduchším.

Vlastne vytvorený image používa väčšinou už image vytvorený inou osobou. Taký image je možné nájsť v registri imageov – v Docker Hube. Pomocou tejto platformy je možné okrem sťahovania rôznych imageov taktiež zdieľať vlastné obrazy s ostatnými.

Keďže použitie Dockeru prináša výhody ohľadom nasadenia aplikácie, bude v tomto prípade využitá táto technológia pre spustenie aplikácie. Použitím Dockeru je taktiež možné vyhnúť sa prípadnou nemožnosťou zostavenia aplikácie, čoho príkladom môže byť problematické zostavenie súčasnej desktopovej aplikácie.

4.6 Analýza frameworkov pre tvorbu webovej aplikácie

Webovú aplikáciu je možné od základov naprogramovať len pomocou vlastného kódu, avšak takýto vývoj by bol nie len zdĺhavejší, ale aj pracnejší, nakoľko by sa musel samotný vývojár zamerať na viac než len na implementáciu samotnej aplikácie. Riešením je použitie webového frameworku, ktorý vývojára od takejto práce odbremenia.

Pre vývoj webovej aplikácie by bolo vhodné využiť framework, ktorý je postavený nad Node.js technológiou, nakoľko by sa klientská a taktiež serverová časť aplikácie dala naprogramovať v jednom jazyku – JavaScripte. Plusom by v tomto prípade bolo, ak by daný framework natívne podporoval TypeScript, čo by mohlo zredukovať prípadné množstvo chýb v implementácii webovej aplikácie. Medzi ďalšími výhodami by patrilo použitie fullstack frameworku, vďaka ktorému by bolo potrebné orientovať sa len v jednom frameworku určenom pre obe časti webovej aplikácie – frontend aj backend.

V súčasnosti medzi najviac používané fullstackové frameworky, ktoré spĺňajú požiadavky uvedené vyššie, patria Nuxt.js[53] a Next.js.[54]

4.6.1 Nuxt.js

Nuxt.js je voľne dostupný open-source framework, pomocou ktorého je možné vytvárať fullstack webové aplikácie a stránky pomocou Vue.js.[55] Vue.js technológia bude popísaná v samostatnej podsekcii nižšie.

Nuxt.js ponúka automatický routing na základe štruktúry súborov v `/pages` zložke. Taktiež automaticky delí kód na menšie celky, čo môže pomôcť s prvým načítaním webovej aplikácie. Okrem renderovania obsahu až na klientovi je možné renderovať obsah už na serveri a takýto obsah poslať naspäť webovému prehliadaču. Pomocou automatických importov Vue.js komponentov nie je potrebné explicitne importovať použité Vue.js komponenty. Samotný framework je naprogramovaný v TypeScripte, čo znamená že je možné využívať type-hinty čo sa týka funkcionality Nuxt.js pri programovaní webovej aplikácie i bez nutnosti použitia TypeScriptu [30] (vlastný preklad).

Na pozadí používa Nuxt.js Nitro[56] server, ktorý generuje API endpointy na základe štruktúry súborov nachádzajúcich sa v zložke `server/api` [30] (vlastný preklad). Nitro je taktiež zodpovedné za zostavenie aplikácie, po-

[53] https://nuxt.com
[54] https://nextjs.org
[55] https://vuejs.org
[56] https://nitro.unjs.io/

mocou príkazu „nuxt build“ – jeho výstupom je `.output` zložka obsahujúca minifikované súbory zbavené všetkých nepoužitých závislostí. Túto zložku je následne možné nasadiť na server podporujúci Node.js a zostavenú aplikáciu spustiť pomocou príkazu `node .output/index.mjs` .

4.6.1.1 Vue.js

Vue.js je JavaScript framework určený pre budovanie používateľského rozhrania postavený na štandardných technológiach ako HTML, CSS a JavaScript. Tento framework poskytuje deklaratívny programovací model založený na znovupoužiteľných komponentoch, ktoré je možné použiť v rámci iných komponentov, čím pomáha zefektívniť proces vývoja znížením nutnosti použitia duplicitného kódu. Pod pojmom „komponent“ je možné predstaviť si samostatnú jednotku používateľského rozhrania (v HTML) s definovanými štýlmi (pomocou CSS) a stavom, ktorý je riadený pomocou JavaScriptu.

Nasleduje príklad využitia Vue.js frameworku – pomocou vytvorenia tzv. Single File Componentu (SFC), ktorý v rámci jedného súboru kombinuje použitie HTML, CSS a JS ako v popise uvádzanom vyššie.

ParagraphComponent.vue

```
<template>
  <p>{{ paragraphText }}</p>
</template>

<script setup lang='ts'>
import { computed, defineProps } from 'vue';

const props = defineProps({
  text: {
    type: String,
    default: '',
  },
});

const paragraphText = computed(() => {
  return props.text;
});
</script>

<style lang='scss' scoped>
p {
  color: red;
}
</style>
```

Horeuvedený príklad demonštruje dve hlavné funkcie Vue.js frameworku:

- deklaratívne vykreslovanie a
- reaktivitu [31] (vlastný preklad).

V prvom prípade sa jedná o rozšírenie štandardného HTML o template syntax – {{ }}, ktorý umožňuje dynamicky vykresliť obsah na základe JavaScript stavu. V uvedenom príklade sa jedná o zobrazenie paragrafu, kde zobrazený text pochádza z premennej `paragraphText`. Táto premenná obsahuje `computed` funkciu, ktorá vracia hodnotu `props.text`. Táto hodnota pochádza zo šablóny iného komponentu, kde bol tento komponent (`ParagraphComponent.vue`) importovaný. Nasledujúci príklad ukazuje práve tento prípad.

`ArticleComponent.vue`

```
<template>
  <paragraph-component text="Example text">
</template>

<script setup lang='ts'>
import { ParagraphComponent } from './ParagraphComponent.vue';
</script>
```

V súbore `ArticleComponent.vue` bol importovaný `ParagraphComponent.vue` komponent, ktorý definuje atribút `text` a nastavuje ho na hodnotu „Example text“. Hodnota tejto premennej sa tým pádom spropaguje do `ParagraphComponent.vue` komponentu do premennej `props.text`. `props` je vlastne objekt, v ktorom je možné nájsť všetky takto definované „properties“. Ak by sa namiesto fixného textu v atribúte „text“ nachádzala premenná, ktorá by zmenila hodnotu, funkcia `computed` zaistí, že sa jej vrátená hodnota (`paragraphText`) zmení na základe detekovanej zmeny jej hodnoty. Na základe tohto príkladu bola ukázaná sila reaktivity Vue.js frameworku.

Horeuvedeným spôsobom je možné modulárne vytvárať a zobrazovať rozličné komponenty podľa potreby. Taktiež je možné reagovať na rozličné eventy emitované prehliadačom, ako napr. na kliknutie myši na určitý element, posun po webovej stránke, atď.

Nakoľko webový prehliadač neumožňuje priamo importovať Vue.js komponenty, je nutné ich zostavením skonvertovať do JavaScriptu, napr. pomocou nástroja Vite.[57] Ten je nutné nainštalovať ako závislosť, napr. pomocou nástroja npm. Následne stačí vytvoriť konfiguračný súbor, v ktorom sa definuje zostavovanie Vue.js komponentov a následne pomocou príkazu `vite build` je možné zostaviť Vue.js komponenty do `.js` súborov, ktoré môžu byť následne importované do HTML šablóny.

4.6.2 Next.js

Next.js je voľne dostupným, a taktiež open-source fullstack frameworkom podobne ako Nuxt.js. Narozdiel od Nuxt.js nepoužíva Vue.js pre vytváranie znovupoužiteľných UI komponentov, ale React.js[58] framework.

[57] https://vitejs.dev
[58] https://react.dev/

Ponúka nástroje pre vytváranie API endpointov, ktoré musia byť vytvorené v zložke `pages/api`. Čo sa týka samotnej funkcionality, taktiež podporuje kompilovanie UI komponentov do spustiteľného JS kódu, jeho minifikovanie pre rýchlejší prenos dát medzi serverom a webovým prehliadačom, code-splitting (rozdelenie kódu pre zlepšenie výkonu aplikácie) až po server-side rendering (SSR, vykreslovaný obsah na serveri sa pošle klientovi). Samotný framework je naprogramovaný pomocou TypeScriptu, takže je možné využívať nápovedu pri programovaní webovej aplikácie pomocou tohto frameworku [32] (vlastný preklad).

Zostavenie aplikácie je možné príkazom `next build`. Tento príkaz vytvorí `.next` zložku obsahujúcu skompilovaný obsah aplikácie. Takto skompilovanú aplikáciu je následne možné spustiť príkazom `next start` [32] (vlastný preklad).

4.6.2.1 React.js

React.js je taktiež JavaScript framework, ktorý poskytuje možnosti pre budovanie používateľského rozhrania. Svojim účelom je podobný Vue.js frameworku a taktiež patrí medzi open-source nástroj, ktorý je avšak spravovaný firmou Meta.[59]

React.js sa od Vue.js líši spôsobom definovania komponentu – nepoužíva SFC pre definovanie štruktúry komponentu, jeho štýlu a stavu. Pre definovanie štruktúry komponentu používa React.js tzv. JSX, ktorý umožňuje písať HTML v JavaScripte.

JSX je oproti HTML striktnejší v tom, že:

- samotný komponent musí byť uzatvorený v značke,
- všetky značky musia mať uzatváraciu značku a
- atribúty značiek je potrebné písať v camelCase forme [33] (vlastný preklad).

Použitie CSS je v komponente možné pomocou importovania CSS stylesheetu napísaného pre daný komponent [34] (vlastný preklad), alebo importovania tzv. CSS modulu, ktorý je možné prepoužiť viacerými komponentmi [35] (vlastný preklad). Taktiež je možné CSS definovať priamo v JS a ten naviazať priamo na element v komponente. CSS naviazané týmto spôsobom je možné dynamicky meniť v závislosti od vnútorného stavu komponentu.

[59] https://about.meta.com/

Takto definovanému komponentu zostáva už len definovať jeho stav. Ten sa definuje pomocou funkcie `useState`, ktorého argumentom je inicializačná hodnota. Táto funkcia vracia pole, kde na nultom indexe sa nachádza aktuálna hodnota daného stavu a na prvom indexe sa nachádza funkcia, ktorú je možné exekuovať pre aktualizáciu stavu [36] (vlastný preklad). Nasledujúci príklad zobrazuje použitie tejto funkcie:

```
import { useState } from 'react';

const [answer, setAnswer] = useState('');
console.log(answer); // výstupom na konzole bude prázdny reťazec
setAnswer('foo');
console.log(answer); // výstupom na konzole bude reťazec 'foo'
```

Pre predstavu je na nasledujúcom príklade ukázaný rovnaký komponent ako v príklade pre Vue.js, avšak upravený pre React.js framework:

```
import 'paragraph.css';

export default function Paragraph({ paragraphText = '' }) {
  return (
    <>
      <p> {paragraphText} </p>
    </>
  );
}
```

V porovnaní s kódom pre Vue.js je horeuvedený kód kratší, nakoľko neobsahuje „boilerplate“ pre definovanie prijímaných properties ako vo Vue.js. Taktiež je automaticky zaistené prepojenie hodnoty „paragraphText“ s vyrenderovaním jeho obsahu v `<p>` tagu. Pre fungovanie importovania horeuvedeného komponentu je potrebné funkciu, ktorá obsahuje definíciu React.js komponentu, exportovať. Táto povinnosť vo Vue.js odpadá.

Samozrejme je rozsah rozdielov väčší než tie uvedené v tejto práci. Účelom ukážky je informovať o základných rozdieloch vytvárania komponentov v oboch frameworkoch.

4.6.3 Výsledná voľba frameworku

Na základe doterajších skúseností s Vue.js frameworkom by mal byť vývoj aplikácie pomocou Nuxt.js frameworku pre autora plynulejší a menej proble-

matický, keďže autor nemá skúsenosti nie len s React.js frameworkom ale ani s Nuxt.js frameworkom.

4.7 Analýza spracovania MR snímkov vo webovej aplikácii

Spracovávanie importovaných MR snímkov by v aplikácii malo prebiehať najmä na strane klienta – vo webovom prehliadači. Takto zvolený prístup zamedzí prípadnému útočníkovi preniknúť k snímkom a dátam o pacientoch, ktoré by v opačnom prípade museli byť uchovávané na strane servera. Z uvedeného vyplýva, že pre implementáciu aplikácie bude potrebné nájsť JavaScript knižnicu resp. knižnice, ktoré sú schopné spracovať DICOM súbory v prehliadači.

Pod pojmom „spracovať“ je myslené: čítanie hlavičky DICOM súborov, zobrazenie snímkov nachádzajúcich sa v týchto súboroch, či tieto snímky modifikovať. Medzi ďalšie požiadavky kladené na takúto knižnicu patrí jej aktívny vývoj, dostupná dokumentácia a taktiež použiteľnosť knižnice pre produkčné nasadenie. Je možné, že neexistuje daná knižnica spĺňajúca všetky požiadavky, ktoré sú na ňu kladené. V takom prípade by mal byť nájdený mix knižníc, ktoré spolu tieto podmienky spĺňajú.

Bohužiaľ, všetky požiadavky kladené na hľadanú knižnicu nespĺňa ani jedna nájdená knižnica, ale výber viacero knižníc, kde každá z nich implementuje určitú časť požiadaviek a dokopy podmienky kladené na knižnicu vyššie, spĺňajú.

Jedná sa o nasledovné knižnice:

- Cornerstone Core,
- Cornerstone WADO Image Loader a
- Dicom Parser.

4.7.1 Cornerstone Core

Cornerstone Core[60] je knižnica, ktorá má za úlohu zjednodušiť proces vývoja komplexnejších webových aplikácií, ktoré majú za úlohu zobrazovať snímky akéhokoľvek formátu, vrátane bežných medicínskych snímkových formátov. Taktiež poskytuje API, pomocou ktorého je možné zobrazovať DICOM snímky a meniť ich vlastnosti, ako napr. zvýšiť alebo znížiť jas, priblížiť snímku alebo ju oddialiť, a iné.

[60] https://github.com/cornerstonejs/cornerstone

Táto knižnica neimplementuje import DICOM súborov a ich spracovanie. Túto funkcionalitu deleguje na tzv. ImageLoaders. Tie po spracovaní DICOM súborov posunú DICOM dáta cez spoločné rozhranie Cornerstone Core knižnici, ktorá ich nakoniec vykreslí. Cieľom tohto prístupu Cornerstone Core knižnice je jej dôraz na minimalizmus a poskytnutie flexibility pri spracovávaní rôznych typov obrazových dát. Použitie Cornerstone Core knižnice pre vývoj špecializovaných aplikácií tohto typu je de-facto štandardom.

V súčasnosti sa pripravuje nová „Cornerstone Core“ knižnica, ktorej názov sa zmení na „Cornerstone3D“.[61] Keďže je táto knižnica momentálne v beta berzii a stabilná verzia tejto knižnice ešte nebola vydaná, vývoj aplikácie bude postavený na doterajšej Cornerstone Core knižnici. Momentálne neexistuje alternatíva tejto knižnice, ktorá by sa špecifikovala na túto oblasť.

4.7.2 Cornerstone WADO Image Loader

Cornerstone WADO Image Loader[62] je tzv. ImageLoader, ktorý je zodpovedný za načítanie a spracovanie DICOM súborov. Použitie tejto knižnice je vynútené Cornerstone Core knižnicou. Táto knižnica podporuje nielen načítanie DICOM súborov cez HTTP protokol, ale aj z lokálneho súborového systému pomocou File API[63] implementovaného webovými prehliadačmi.

Po načítaní DICOM súborov je ich parsovanie prenechané knižnici Dicom Parser. Nakoľko sa veľkosť týchto súborov môže pohybovať v rádoch megabajtov (MB), samotné parsovanie súborov beží tiež pomocou webovej technológie zvanej Web Workers.[64] Pre začiatok priblížim technológiu Web Workers a následne knižnicu Dicom Parser.[65]

4.7.2.1 Web Workers

JavaScript je v prehliadači implementovaný ako jednovláknový jazyk využívajúci jedno hlavné vlákno a exekúcia skriptov tohto jazyka prebieha zvyčajne v tomto vlákne. Výpočetne náročné úlohy by avšak mohli vyústiť do zablokovania tohto vlákna, ktoré sa prejavuje nereagovaním prehliadača na rozličné používateľské akcie alebo nevykreslovaním aktualizácií na webovej stránke. Dôvodom zablokovania hlavného vlákna by v tomto prípade bolo využitie všetkých dostupných prostriedkov prioritne pre danú výpočetne náročnú úlohu.

Web Workers technológia je štandardom, ktorý je implementovaný a poskytovaný webovými prehliadačmi umožňujúci exekúciu takýchto úloh, ktoré

[61] https://github.com/cornerstonejs/cornerstone3D-beta
[62] https://github.com/cornerstonejs/cornerstoneWADOImageLoader
[63] https://developer.mozilla.org/en-US/docs/Web/API/File_API
[64] https://developer.mozilla.org/en-US/docs/Web/API/Web_Workers_API
[65] https://github.com/cornerstonejs/dicomParser

by inak pri dlhšom spracovávaní mohli dané hlavné vlákno zablokovať. Pomocou Web Workers je možné predísť zablokovaniu hlavného vlákna jednoduchým vytvorením nového pracovného vlákna pomocou konštruktu `new Worker(url)`, kde `url` je adresa skriptu, ktorý má bežať v novom pracovnom vlákne. Takéto pracovné vlákno môže exekuovať JS skript bez zablokovania hlavného vlákna, keďže je od neho nezávislé [37].

4.7.3 Dicom Parser

Dicom Parser je knižnica implementujúca parsovanie všetkých známych validných DICOM súborov. Knižnica je navrhnutá pre beh vo všetkých moderných HTML5 prehliadačoch a k svojej funkčnosti nezávisí na iných knižniciach. Dicom Parser poskytuje globálny objekt `dicomParser`, ktorý obsahuje viacero metód, z ktorých je najzaujímavejšia metóda `parseDicom`. Argumentom tejto metódy je `Uint8Array` pole obsahujúce nespracovaný (raw) obsah DICOM súboru. Výsledkom volania tejto metódy spolu s `Uint8Array` poľom je `DataSet` objekt obsahujúci vyparsovaný obsah DICOM súboru.

Alternatívou Dicom Parser knižnice by mohla byť knižnica dcm.js,[66] avšak vývoj tejto knižnice nie je stále dokončený (nebola zatiaľ vydaná jej stabilná verzia) a sami vývojári varujú pred použitím tejto knižnice v produkčnom prostredí.[67]

Pre zhrnutie informácií v tejto sekcii – knižnica Cornerstone WADO Image Loader využíva Web Workers pre vytvorenie nových pracovných vlákien, ktorých úloha je parsovanie DICOM súborov pomocou metódy `parseDicom` objektu `dicomParser` pochádzajúceho z Dicom Parser knižnice uvedenej vyššie. Metódou vrátený `DataSet` objekt je následne poslaný knižnici Cornerstone Core, ktorá sa postará o vykreslenie vyparsovaného DICOM snímku z tohto objektu.

Nasledujúca sekcia sa bude zaoberať analýzou a návrhom, ako vykresliť mriežku na zobrazenú DICOM snímku pomocou knižnice Cornerstone Tools.

4.8 Analýza vykreslenia mriežky nad MR snímkami

Keďže bude nielen potrebné MR snímky zobrazovať, ale aj nad týmito snímkami vykreslovať používateľom generovanú mriežku (ako bolo uvedené vo funkčnej požiadavke 4.1.1.3), je nutné nájsť knižnicu, ktorá vykresľovanie takejto mriežky nad určitou oblasťou MR snímky podporuje.

[66] https://github.com/dcmjs-org/dcmjs
[67] https://github.com/dcmjs-org/dcmjs

4.8.1 Cornerstone Tools

Tou knižnicou je Cornerstone Tools[68] knižnica, ktorá asistuje nielen pri vytváraní rôznych anotácií pre DICOM snímky načítané pomocou Cornerstone Core, ale aj pri ich segmentácii či rôznych meraní. Táto knižnica ponúka široký počet nástrojov, ktoré môžu dané snímky modifikovať alebo nad týmito snímkami vykreslovať rôzne informácie či lomené čiary.

Prednosťou tejto knižnice je API, pomocou ktorého je možné vytvárať nové nástroje, manažovať ich, či importovať/exportovať ich stav. Pre využitie tejto knižnice je potrebná nielen Cornerstone Core knižnica, nakoľko je s ňou úzko previazaná, ale aj knižnica Hammer.js[69] a Cornerstone Math.[70]

Cornerstone Tools využíva Cornerstone Core knižnicu pre reagovanie na rôzne eventy, ktoré Conerstone Core knižnica emituje. Na základe týchto eventov môžu nástroje Cornerstone Tools knižnice meniť svoj stav. Hammer.js knižnica implementuje podporu rozhrania založeného na dotyku namiesto myši. Túto knižnicu je potrebné importovať bez ohľadu na to, či sa plánujú využívať gestá na báze dotyku alebo nie, nakoľko niektoré nástroje sú od tejto knižnice závislé. Cornerstone Math, ako už názov napovedá, poskytuje rôzne matematické operácie prevažne týkajúce sa vektorovej matematiky. Niektoré nástroje z Cornerstone Tools knižnice ju používajú napr. pre výpočet vzdialenosti medzi rôznymi bodmi.

Nakoľko Cornerstone Tools neobsahuje mriežku ako nástroj, ktorý vie knižnica zobraziť a s ňou ďalej pracovať, bude nutné túto mriežku od základov implementovať. Implementácia tejto mriežky môže využívať API poskytované knižnicou, pomocou ktorej by bolo možné danú mriežku implementovať bez zásahu do knižnice alebo bude nutné danú mriežku naprogramovať priamo do tejto knižnice.

Obe možnosti implementácie majú svoje výhody a nevýhody. Pri implementácii mriežky pomocou dedikovaného API by nebolo potrebné udržiavať vlastnú kópiu Cornerstone Tools knižnice. V tomto prípade by pre využitie rôznych funkcií implementovaných v samotnej knižnici bolo možné vyžadovanú funkcionalitu importovať pomocou metódy `importInternal(moduleName)`. Nevýhodou tohto spôsobu je nedostatočná flexibilita spojená s nemožnosťou importovania všetkej funkcionality, ktorá by mohla byť pri vývoji potrebná. Ďalším negatívom zvolenia tohto spôsobu by bola nemožnosť upravenia akéhokoľvek kódu v Cornerstone Tools knižnici.

[68] https://github.com/cornerstonejs/cornerstoneTools
[69] https://github.com/hammerjs/hammer.js
[70] https://github.com/cornerstonejs/cornerstoneMath

Na druhú stranu, ak by mala byť mriežka implementovaná priamo v Cornerstone Tools knižnici, odpadol by problém s importovaním teoreticky potrebnej funkcionality, nakoľko by sa dal importovať akýkoľvek modul knižnice priamo pomocou JS konštruktu `import`, bez nutnosti využitia `importInternal` metódy.

Nakoľko v tomto momente nie je jasné, či bude výsledná implementácia mriežky potrebovať zmenu niektorého zo súborov Cornerstone Tools knižnice, je vhodnejšie začať implementovať mriežku priamo v knižnici. Keď bude implementácia tejto mriežky dokončená, bude nutné posúdiť, či je možné celú funkcionalitu mriežky migrovať do riešenia využívajúceho iba dedikované API pre svoju funkcionalitu. .

v implementácii zhodnotiť že to nie je možné, nakoľko boli potrebné zmeny v samotných súboroch cornerstone tools

Ako pri Cornerstone Core, tak aj Cornerstone Tools knižnica bude mať čoskoro svojho nástupcu, knižnicu Cornerstone Tools v2,[71] ktorá bude určená pre Cornerstone3D. Táto nová knižnica je momentálne v aktívnom vývoji a jej stabilná verzia ako v prípade Cornerstone3D nebola stále vydaná. To je dôvodom, prečo bude pri implementácii aplikácie použitá doterajšia verzia Cornerstone Tools knižnice.

4.9 Návrh komunikácie webového rozhrania so serverom

pridať diagram komunikácie

4.9.1 Anonymizácia DICOM dát

4.10 Návrh používateľského rozhrania

pridať wireframe

[71] https://github.com/cornerstonejs/cornerstone3D-beta/tree/main/packages/tools

Kapitola 5

Implementácia

5.1 Klientská časť

5.1.1 Používateľské rozhranie

5.1.2 Spracovanie DICOM dát

5.1.3 Interaktívna úprava mriežky

5.2 Serverová časť

5.2.1 Výpočet umiestnenia mriežky

Kapitola 6

Testovanie

Kapitola 7

Zhodnotenie aplikácie a odporúčania pre jej ďalší vývoj

Kapitola 8

Záver

Bibliografia

1. MACKIEWICH, Blair. *Intracranial boundary detection and radio frequency correction in magnetic resonance images*. Burnaby, 1995. Dostupné tiež z: https://summit.sfu.ca/item/6770. Diplomová práca. Simon Fraser University.
2. LAM, Peter et al. What to know about MRI scans. In: Healthline Media. *Medical News Today* [online]. Júl 24, 2018 [cit. 2023-03-24]. Dostupné tiež z: https://www.medicalnewstoday.com/articles/146309#what-is-an-mri-scan.
3. MAZÁNKOVÁ, Jitka. *Kontrastní látky a jejich nežádoucí účinky*. Brno, 2011. Dostupné tiež z: https://is.muni.cz/th/rsk3f/. Bakalárska práca. Masarykova univerzita, Lekárska fakulta.
4. JÍRALOVÁ, Tereza. *MR - nové trendy a význam v moderní diagnostice*. České Budějovice, 2021. Dostupné tiež z: http://wstag.jcu.cz/StagPortletsJSR168/CleanUrl?urlid=prohlizeni-prace-search&studentSearchPrijmeni=j%C3%ADralov%C3%A1. Bakalárska práca. Jihočeská univerzita v Českých Budějovicích, Zdravotně sociální fakulta.
5. ELSTER, Allen D. What is SPAMM? *Myocardial tagging/SPAMM - Questions and Answers in MRI* [online]. 2023 [cit. 2023-03-24]. Dostupné tiež z: https://mriquestions.com/taggingspamm.html.
6. BRYANT, David. About Qt. In: *Main* [online]. 18. júla 2022 [cit. 2023-03-29]. Dostupné tiež z: https://wiki.qt.io/index.php?title=About_Qt.
7. THE QT COMPANY. Qt Core 5.15.13. *doc.qt.io* [online]. 2023 [cit. 2023-03-29]. Dostupné tiež z: https://doc.qt.io/qt-5/qtcore-index.html.

8. THE QT COMPANY. Qt Widgets 5.15.13. *doc.qt.io* [online]. 2023 [cit. 2023-03-29]. Dostupné tiež z: `https://doc.qt.io/qt-5/qtwidgets-index.html`.
9. THE QT COMPANY. Qt GUI 5.15.13. *doc.qt.io* [online]. 2023 [cit. 2023-03-29]. Dostupné tiež z: `https://doc.qt.io/qt-5/qtgui-index.html`.
10. THE QT COMPANY. Qt Test 5.15.13. *doc.qt.io* [online]. 2023 [cit. 2023-03-29]. Dostupné tiež z: `https://doc.qt.io/qt-5/qttest-index.html`.
11. THE QT COMPANY. qmake Manual. *doc.qt.io* [online]. 2023 [cit. 2023-03-29]. Dostupné tiež z: `https://doc.qt.io/qt-5/qmake-manual.html`.
12. VARMA, Dandu Ravi. Managing DICOM images: Tips and tricks for the radiologist. *Indian Journal of Radiology and Imaging*. 2012, roč. 22, č. 01, s. 4–13.
13. DICOM LIBRARY. About DICOM format. *DICOM Library* [online]. 2023 [cit. 2023-03-29]. Dostupné tiež z: `https://www.dicomlibrary.com/dicom/`.
14. THE MEDICAL IMAGING TECHNOLOGY ASSOCIATION. History. *dicomstandard.org* [online]. [2023] [cit. 2023-03-29]. Dostupné tiež z: `https://www.dicomstandard.org/history`.
15. DICOM LIBRARY. Transfer Syntax. *DICOM Library* [online]. 2023 [cit. 2023-03-29]. Dostupné tiež z: `https://www.dicomlibrary.com/dicom/transfer-syntax/`.
16. OFFIS E. V. DCMTK. *dicom.offis.de* [online]. [2023] [cit. 2023-03-29]. Dostupné tiež z: `https://dicom.offis.de/dcmtk.php.en`.
17. LANGR, Daniel et al. *Úvod do OpenMP* [prednáška]. 2023. Dostupné tiež z: `https://courses.fit.cvut.cz/NI-PDP/media/lectures/NI-PDP-Prednaska02-OpenMP.pdf`. Praha: ČVUT v Praze, Fakulta informačních technologií, 27.02.2023.
18. *Homepage - TNL - Template Numerical Library* [online]. 2023 [cit. 2023-03-29]. Dostupné tiež z: `https://tnl-project.org`.
19. KAFKA, Jiří. *Vývoj aplikace pro analýzu pohybu srdečních komor*. Praha, 2015. Diplomová práca. České vysoké učení technické v Praze, Fakulta jaderná a fyzikálně inženýrská.
20. OPENJS FOUNDATION. C++ addons In: *Node.js v18.16.0 documentation* [online]. 2023 [cit. 2023-04-13]. Dostupné tiež z: `https://nodejs.org/docs/latest-v18.x/api/addons.html`.
21. MDN CONTRIBUTORS. WebAssembly Concepts. In: *MDN Web Docs* [online]. Mar 23, 2023 [cit. 2023-04-13]. Dostupné tiež z: `https://developer.mozilla.org/en-US/docs/WebAssembly/Concepts`.

22. MDN CONTRIBUTORS. Compiling a New C/C++ Module to WebAssembly. In: *MDN Web Docs* [online]. Mar 24, 2023 [cit. 2023-04-13]. Dostupné tiež z: `https://developer.mozilla.org/en-US/docs/WebAssembly/C_to_wasm`.

23. MDN CONTRIBUTORS. Emscripten Diagram. In: *MDN Web Docs* [online]. Mar 23, 2023 [cit. 2023-04-13]. Dostupné tiež z: `https://developer.mozilla.org/en-US/docs/WebAssembly/Concepts/emscripten-diagram.png`.

24. STEPANYAN, Ingvar. Using WebAssembly threads from C, C++ and Rust. In: *web.dev* [online]. Júl 12, 2021 [cit. 2023-04-13]. Dostupné tiež z: `https://web.dev/webassembly-threads/`.

25. TAB ATKINS JR. Elika J. Etemad, Florian Rivoal. CSS Snapshot 2022. In: *w3.org* [online]. 22. November 2022 [cit. 2023-04-17].

26. SHU-YU GUO Michael Ficarra, Kevin Gibbons et al. ECMAScript® 2024 Language Specification. In: *tc39.es* [online]. 2023 [cit. 2023-04-16].

27. TYPESCRIPT DOCS CONTRIBUTORS. TypeScript for the New Programmer. In: *typescriptlang.org* [online]. Apr 12, 2023 [cit. 2023-04-16].

28. OPENJS FOUNDATION AND NODE.JS CONTRIBUTORS. About Node.js®. *node.js* [online]. 2023 [cit. 2023-04-17].

29. DOCKER INC. Docker overview. In: *docs.docker.com* [online]. 2023 [cit. 2023-04-17].

30. NUXT.JS CONTRIBUTORS. Introduction. In: *nuxt.com*. 2023 [cit. 2023-04-18]. Dostupné tiež z: `https://nuxt.com/docs/getting-started/introduction`.

31. VUE.JS CONTRIBUTORS. Introduction. In: *vuejs.org*. 2023 [cit. 2023-04-18]. Dostupné tiež z: `https://vuejs.org/guide/introduction.html`.

32. INC., Vercel. What is Next.js? In: *nextjs.org*. 2023 [cit. 2023-04-18]. Dostupné tiež z: `https://nextjs.org/learn/foundations/about-nextjs/what-is-nextjs`.

33. META OPEN SOURCE. Writing Markup with JSX. In: *react.dev*. 2023 [cit. 2023-04-18]. Dostupné tiež z: `https://react.dev/learn/writing-markup-with-jsx`.

34. LLOBERA, Lewis. Adding a Stylesheet. In: *create-react-app.dev*. 2/13/2020 [cit. 2023-04-18]. Dostupné tiež z: `https://create-react-app.dev/docs/adding-a-stylesheet`.

35. HARDO. Adding a CSS Modules Stylesheet. In: *create-react-app.dev*. 1/4/2019 [cit. 2023-04-18]. Dostupné tiež z: `https://create-react-app.dev/docs/adding-a-css-modules-stylesheet`.

36. META OPEN SOURCE. State: A Component's Memory. In: *react.dev*. 2023 [cit. 2023-04-18]. Dostupné tiež z: `https://react.dev/learn/state-a-components-memory`.

37. MDN CONTRIBUTORS. Using Web Workers. In: *MDN Web Docs* [online]. Mar 16, 2023 [cit. 2023-04-16]. Dostupné tiež z: `https://developer.mozilla.org/en-US/docs/Web/API/Web_Workers_API/Using_web_workers`.

Dodatok A

Zoznam použitých skratiek

API Application Programming Interface

AR Augmentovaná realita

CSS Cascading Style Sheets

CT Počítačová tomografia

ČVUT České vysoké učení technické

DICOM Digital Imaging and Communications in Medicine

FID Free Induction Decay

FJFI Fakulta jaderná a fyzikálně inženýrská

GCC GNU Compiler Collection

HTML HyperText Markup Language

HTTP Hypertext Transfer Protocol

ID Identifikátor

IDE Integrated Development Environment

JS JavaScript

MB Megabajt

MR Magnetická rezonancia

NEMA National Electrical Manufacturers Association

SPAMM Spatial Modulation of Magnetization

TE Time to Echo

TNL Template Numerical Library

TR Repetition Time

UI User Interface

VM Virtuálny počítač

VR Virtuálna realita

W3C World Wide Web konzorcium

WADO Web Access to DICOM Objects

WWW World Wide Web

Dodatok **B**

Obsah priloženého média

- `src` ..... adresár so zdrojovými kódmi
  - `app` ..... adresár so zdrojovými kódmi aplikácie
  - `thesis` ..... adresár so zdrojovým kódom práce vo formáte LaTeX
- `text` ..... text práce
  - `DP_Tomáš_Taro_2023.pdf` ..... text práce vo formáte PDF