Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Diplomová práce

Návrh systému měření tremorových dat

Bc. Miroslav Holeček

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Miroslav Holeček

Systémové inženýrství a informatika
Informatika

Název práce

Návrh systému měření tremorových dat

Název anglicky

A proposal of a tremor data measuring system

Cíle práce

Cílem práce je návrh a implementace systému měření tremorových dat pro mobilní zařízení za účelem následné diagnostiky. Data z mobilních zařízení budou synchronizována na centrální databázový server.

Metodika

Práce se bude skládat ze dvou částí. Teoretická část bude založena na rešerši odborné literatury a konzultaci s odborníkem. Poskytne přehled k dané problematice a bude výchozím bodem pro tvorbu praktické části.

Praktická část bude obsahovat návrh a implementaci aplikace pro mobilní zařízení. Ta zpracuje data z měřícího zařízení, umožní uživateli práci s těmito daty a následně je odešle na centrální databázový server. Při transferu dat budou využity kryptovací funkce k zajištění integrity a bezpečnosti dat. Při návrhu a řešení budou použity standardní softwarové nástroje. Dokumentace bude provedena ve standardu UML.

Doporučený rozsah práce

a

Klíčová slova

Měření lidského tremoru, sběr dat, akcelometrie, gyroskop

Doporučené zdroje informací

EBERHART, Russell C.; HU, Xiaohui. Human tremor analysis using particle swarm optimization. In: Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406). IEEE, 1999. p. 1927-1930.

GOYAL, Dinesh, et al. (ed.). Design and Analysis of Security Protocol for Communication. John Wiley & Sons, Incorporated, 2020.

GRIFFITHS, Dawn a David GRIFFITHS. Head first Android development. 2nd edition. Beijing: O'Reilly, 2017. ISBN 9781491974056.

Předběžný termín obhajoby

2020/21 LS – PEF

Vedoucí práce

Ing. Martin Pelikán, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 16. 10. 2020

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 16. 10. 2020

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 22. 03. 2021

Čestné prohlášení

Prohlašuji, že svou diplomovou práci "Návrh systému měření tremorových dat" jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze, dne 31.3.2021_____

Poděkování

Rád bych touto cestou poděkoval Ing. Martinu Pelikánovi, Ph.D. za vedení mé práce a Mgr. Josefu Zemanovi Ph.D. za poskytnuté konzultace. Dále bych rád poděkoval své manželce a dětem, za jejich trpělivost a shovívavost při zpracovávání. Nesmím také zapomenou poděkovat panu Ing. et Ing. Markovi Drápalovi, Ph.D.,za jeho pomoc s konfigurací webového serveru a SQL databáze pro účely ostrého testování.

Návrh systému měření tremorových dat

Abstrakt

Tato diplomová práce je zaměřena na návrh a implementaci mobilní aplikace vytvářející systém registru pacientů, jejich anamnéz a čtveřice souborů naměřených tremorových dat. Tato data je následně nutno odesílat na centrální server, kde mohou být dále zpracovávány, a kde k těmto datům musí být vytvořen přístup přes webové rozhraní. Práce se skládá z teoretické části, ve které se můžeme blíže seznámit s problematikou snímání třesu, s probíhající studií diagnostiky roztroušené sklerózy z klidového třesu, s použitými technologiemi a se základy programování aplikací v systému Android včetně standardu UML. Praktická část je tvořena dvěma částmi. V první části jsou představeny návrhy samotné aplikace, webového rozhraní a centrální databáze SQL pomocí UML diagramů, popisů tříd a funkcí. Ve druhé části je popsána implementace celého řešení včetně návrhu designu webové i aplikační části a včetně ukázek stěžejních částí kódu. V závěru práce jsou zhodnoceny výsledky, shrnuto testování prostředí v ostrém provozu a představen ukázkový případ použití.

Klíčová slova: Měření lidského tremoru, sběr dat, akcelerometrie, gyroskop, Android, vývoj mobilní aplikace

A proposal of a tremor data measuring system

Abstract

This thesis focuses on design and implementation of a mobile application creating a system of patients, their anamnesis and four sets of measured tremor data. It is then necessary to send this data to the central server, where it can be further processed and where access to the data through a web interface must be created. The thesis consists of a theoretical part featuring a closer look at the issue of tremor scanning, ongoing studies of diagnosing multiple sclerosis from resting tremor, used technologies, basics of application development in the Android system including the UML standard. The practical part consists of two parts, the first part features the design of the application itself, the web interface and the central SQL database using UML diagrams, descriptions of classes and functions. In the second part the implementation of the whole solution is introduced, including the design proposal of the web and application section with examples of the core parts of the code. The conclusion of the thesis evaluates the results of testing the environment.

Keywords: Human tremor measurement, data collection, accelerometery, gyroscope, Android, mobile application development

Obsah

Seznam obrázků
Odkazovaný seznam obrázků

Seznam tabulek
Odkazovaný seznam tabulek

Seznam zdrojových kódů
Odkazovaný seznam zdrojových kódů

1 Úvod

Cílem diplomové práce je navrhnout a implementovat systém pro registr pacientů a jejich měření, který bude uživatelsky přívětivý, nebude vyžadovat technické vzdělání, nebude vyžadovat trvalé připojení k internetu a bude bezpečně odesílat svá data na centrální SQL server za účelem pořízení třesu pacientů trpících roztroušenou sklerózou (RS). V druhé části pak bude vytvořen webový přístup k centrální databázi, pomocí kterého budou data zpřístupněna v hierarchii pacientů a jejich měření. Jako volitelná součást diplomové práce je zpřístupnit přes rozhraní Bluetooth snímací zařízení MetaWear, pomocí něhož bude možno naměřit požadovanou čtveřici souborů, jenž se budou spolu s daty odesílat na centrální server.

V úvodní části je provedena rešerše odborné literatury, která vychází z odborných článků a konzultací s odborníkem. Dále pak jsou v teoretické části přiblíženy postupy a metody k návrhu a implementaci aplikací.

V kapitole vlastní práce jsou nejprve specifikovány požadavky na software. Dále jsou popsána navrhnutá řešení, a ty jsou podrobena analýze. V následující kapitole se nachází dokumentace ve standardu UML. Před samotnou implementací jsou představeny návrhy prezentačních struktur a designů mobilní aplikace i webového rozhraní.

Implementace se skládá z přehledu aktivit a fragmentů, jejich stručným popisem a ukázek kódu. Dále pak jsou popsány stěžejní třídy včetně ukázek řešení daných problémů. Implementace webové části je rozdělena na přijímání souborů od aplikace a na zobrazení pacientů a měření. Obě části obsahují popisy důležitých částí i ukázky implementace. Na závěr kapitoly vlastní práce je představena výsledná grafická podoba obou částí.

V kapitole výsledky a diskuse jsou nejprve popsány komplikace při návrhu a implementaci, následně je zde popsáno testování, a poté je zpracováno shrnutí dosažených výsledků. Na závěr kapitoly bude představeno vzorové použití výsledné aplikace a webového rozhraní a shrnutí možností rozšíření.

2 Cíl práce a metodika

2.1 Cíl práce

Cílem práce je návrh a implementace systému měření tremorových dat pro mobilní zařízení za účelem následné diagnostiky. Data z mobilních zařízení budou synchronizována na centrální databázový server.

2.2 Metodika

Práce se bude skládat ze dvou částí. Teoretická část bude založena na rešerši odborné literatury a konzultaci s odborníkem. Poskytne přehled k dané problematice a bude výchozím bodem pro tvorbu praktické části.

Praktická část bude obsahovat návrh a implementaci aplikace pro mobilní zařízení. Aplikace zpracuje data z měřícího zařízení, umožní uživateli práci s těmito daty a následně je odešle na centrální databázový server. Při transferu dat budou využity kryptovací funkce k zajištění integrity a bezpečnosti dat. Při návrhu a řešení budou použity standardní softwarové nástroje. Dokumentace bude provedena ve standardu UML.

3 Teoretická východiska

3.1 Lidský třes a jeho měření

Třes je způsoben rytmickými oscilacemi vyvolanými nedobrovolnými kontrakcemi svalů. I v klidovém stavu jsou lidé vystaveni stěží vnímatelným třesům v důsledku svalových kontrakcí vycházejících z podvědomí. Některé formy třesu mohou být fyziologické, jiné formy mohou být patologické, jako je tomu u Parkinsonovy nemoci, nebo u RS. (Comby, 2008)
Často se jedná o symptom pohybové poruchy a k jeho léčbě dochází v interním medicíně, urgentní medicíně a samozřejmě v neurologii. Příčiny však mohou být různé. Při hodnocení pacienta s třesem, fenomenologie třesu, se zjišťuje přítomnost nebo nepřítomnost dalších neurologických příznaků, příznaků spojených s užíváním léků, alkoholu nebo příznaků spojených s vnějším okolím. K diagnostice příčiny třesu obvykle postačí anamnéza pacienta a následně cílené neurologické vyšetření. (Puschmann, 2011) Skutečná definice třesu se však často liší dle různých autorů a obecný popis lze určit jen stěží.

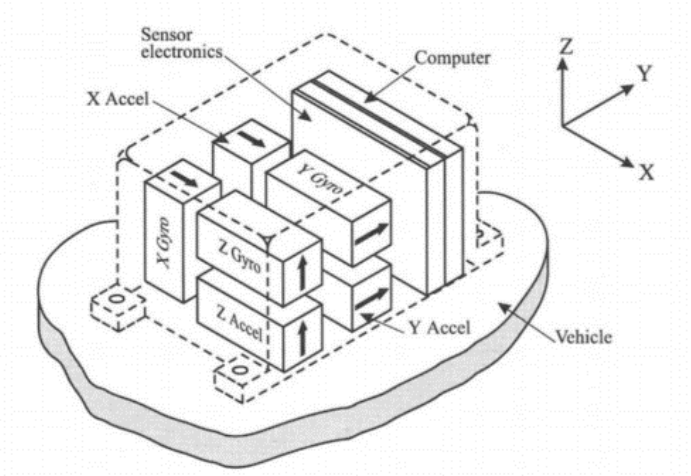
Lidský třes se často dělí do dvou skupin.

Klidový třes
Klidový třes je třes přítomný v určité části těla. Není aktivován dobrovolně a je vyvolán v klidové poloze. (Alusi, 2001)

Akční třes
Akční třes je jakýkoli třes, který je způsoben dobrovolnou kontrakcí svalu. Dělí se dále na posturální, izometrický, kinetický a intenční třes. Posturální třes je vyvolán při dobrovolném udržování polohy proti gravitaci. Kinetický třes je třes, ke kterému dochází při jakémkoli dobrovolném pohybu. K izometrickému třesu dochází v důsledku svalové kontrakce proti pevnému stacionárnímu předmětu. (Alusi, 2001)

3.1.1 Měření třesu

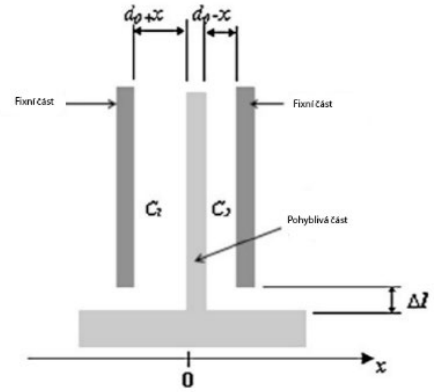
Třes se měří podle pohybů části těla, která jej produkuje. Pohyby lze zaznamenat senzory pro určování zrychlení a polohy, jako jsou právě akcelerometry a gyroskopy. V práci bude využito zařízení, které obsahuje MEMS sensor BMI160 od společnosti Bosch Sensortec. Jeho schématické zobrazení je zobrazeno na obrázku 3.1. Jedná se o šestiosou inerciální měřicí jednotku, která se skládá z 16bitového akcelerometru a gyroskopu. Toto zařízení se vyznačuje nízkou spotřebou energie a přesným výpočtem. (Bosch-sensortec, 2021)



Obrázek 3.1 Schématické zobrazení BMI160 (Varol, 2019)

3.1.2 **Akcelerometr**

Akcelerometry měří zrychlení a mírné náklony ve třech osách. Používají se v moderní elektronice například pro detekci orientace jako stabilizátor obrazu nebo zjištění vibrací. Akcelerometry typu MEMS jsou založeny převážně na technologiích kapacitního snímání. Funkce akcelerometrů spočívá v měření zrychlení, jež se provádí podobným způsobem jako u gyroskopů. Princip jejich funkce je realizován pomocí umístění závaží namontovaného na pružinách viz obrázek 3.2. Jedna část pružin je připevněna k deskám hřebenového kondenzátoru C_2, C_3 a druhá část k instalovanému závaží. Vlivem síly působící na senzor se závaží přesouvá na pružinách a způsobuje změnu vzdálenosti d_0 mezi kondenzačním elementem a hmotou. Tím ovlivňuje změnu kapacity. (Tme.eu, 2020)



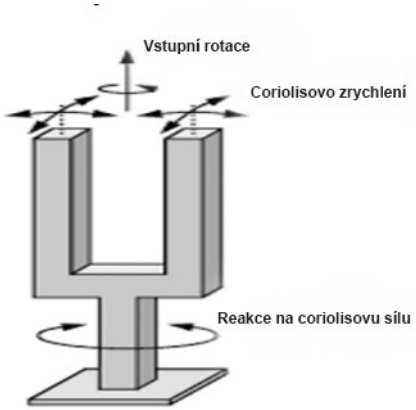
Obrázek 3.2 Princip akcelerometru (Varol, 2019)

K identifikaci třesu jsou klíčové tři parametry akcelerometru: rozsah, rozlišení a vzorkovací frekvence. U senzoru BMI160 je možnost nastavení rozsahu ±2, ±4, ±8, ±16 g, rozlišení je

pevně dané na 16 bitů a vzorkovací frekvenci je možné nastavit v rozsahu 0,001 – 100Hz v režimu stream a až 800Hz v režimu logování. (Bosch-sensortec, 2021)

3.1.3 Gyroskop

Gyroskopy oproti akcelerometrům neměří zrychlení, ale pouze náklony. Ve spolupráci s akcelerometrem tedy dokáží zefektivnit informaci o poloze a náklonu daného zařízení. Gyroskopy MEMS jsou podobné optickým gyroskopům. Oproti mechanickým gyroskopům, které ve skutečnosti indikují orientaci, gyroskopy MEMS udávají rychlost změny úhlu v čase. To se provádí podobným způsobem jako u akcelerometrů MEMS integrací změn úhlové rychlosti. Proto se jim také někdy říká rychlostní gyroskopy. Výhoda takového řešení je v ceně výroby a v celkové velikosti sensoru. Vzhledem k velmi nízké spotřebě se jich hojně využívá v mobilním průmyslu. (Varol, 2019)



Obrázek 3.3 Princip gyroskopu - triaxiální gyroskop (Varol, 2019)

Pro měření třesu rukou jsou významné především tři parametry gyroskopu: citlivost, rozlišení a vzorkovací frekvence. U senzoru BMI160 je možnost nastavení rozsahu ±125, ±250, ±500, ±1000, ±2000°/s, rozlišení je opět pevně dané na 16 bitů, vzorkovací frekvenci je možné opět nastavit v rozsahu 0,001 – 100Hz v režimu stream a až 800Hz v režimu logování. (Bosch-sensortec, 2021)

3.2 Roztroušená skleróza

Roztroušená skleróza (RS) je nejčastější netraumatické invalidizující onemocnění postihující především mladé dospělé. Výskyt RS se celosvětově zvyšuje a je častější u žen než u mužů. Příčina onemocnění RS a důvod nárůstu nemocných zatím není znám, i když je téměř jisté, že významnou roli hrají genetické vlivy a vlivy prostředí. Výzkumy naznačují, že nízká hladina vitaminu D, kouření a dětská obezita pravděpodobně hrají velikou roli ve vývoji onemocnění. Možnost dřívější diagnostiky onemocnění může silně ovlivnit následující průběh nemoci. (Dobson, 2018)

3.2.1 Diagnostika roztroušené sklerózy z klidového třesu

V této části bude přiblížena studie, která se zabývá diagnózou RS z tremoru horních končetin, ze které vychází požadavek na tvorbu systému, jenž je předmětem diplomové

práce. Vycházím z konzultací s vedoucím projektu panem Mgr. Josefem Zemanem Ph.D., neboť se danou problematikou dlouhodobě zabývá. Dále čerpám z doposud nepublikovaného článku pana doktora Zemana "Diagnostika RS z klidového tremoru".

3.2.1.1 Metoda

Třes byl testován na obou horních končetinách, obě paže byly střídány po jednominutových měřeních s otevřenýma a zavřenýma očima dle schématu PO (pravá ruka, otevřené oči), LO (levá ruka, otevřené oči), PZ (pravá ruka, zavřené oči), LZ (levá ruka, zavřené oči). Během měření osoba stála vzpřímeně s nohama rozkročenýma do šířky boků s měřenou rukou předpaženou a druhou volně podél těla.
Studie se účastnili dobrovolníci ve věku 20-70 let a věkový průměr činil 29 let. Zdravá kontrolní skupina neměla diagnostikovány žádné neurologické onemocnění v době měření. RS pozitivní pacienti byli vybráni na základě několika kritérií: jednoznačně klinická diagnóza RS na základě kritérií McDonald, neurologem zjišťována Expanded Disability Status Scale v rozmezí 2 – 6,5 a dále musel být pozitivní RS pacient měsíc bez recidivy nemoci a beze změn farmakoterapie.
Pro měření zrychlení a rotací byl použit čip LSM6DS0 s tříosým akcelerometrem a tříosým gyroskopem. Pro akcelerometr byl použit rozsah h ±2g s 16 bitovým rozlišením a gyroskop s rozsahem ±245dps rovněž s 16 bitovým rozlišením. Vzorkovací frekvence byla u obou typů snímačů 119 Hz. Data byla přenesena po sběrnici USB a logována v počítači. Hmotnost zařízení připevněného na ruce činila 36 g. USB kabel byl fixován k paži a trupu měřeného, jako fixační pásek byl použit koženkový pásek k hodinkám. (Zeman, 2021)

3.2.1.2 Zpracování dat

Ke zpracování dat byla zvolena patnácti sekundová část z naměřeného signálu na počátku záznamu. Ta byla přenásobena Gaussovým oknem, dále byla aplikována Fourierova transformace, čímž bylo zjištěno jeho frekvenční spektrum. Okno bylo přesunuto o jeden vzorek dále v čase a opět byla aplikována Fourierova transformace na úsek přenásobený Gaussovým oknem. Spektrum, které vzniklo, bylo přičteno k předchozímu. Stejný postup byl aplikován na všechny následující vzorky naměřeného signálu. Výsledkem byl zisk průměrného spektra. Nejvhodnějším řešením bylo porovnání stejně pořízených záznamů.
Identifikace spektra obsahující informaci o přítomnosti RS bylo následující. Cílem byla snaha o rozdělení získané množiny šestkrát 4n, kde n byl počet pacientů, spekter podle jejich vlastností tak, aby výsledné množiny co nejpřesněji odpovídaly požadavku rozřazení na pacienty s RS a na ostatní zdravé probandy. Identifikace spektra obsahující informaci o přítomnosti RS byla prováděna několika způsoby. Bylo zjištěno, že použití clusterové analýzy s euklidovskou metrikou ward.D algoritmem není pro diagnostiku RS vhodné. Dále byla provedena shluková analýza, ale nejlepších výsledků bylo dosaženo hledáním vzájemně podobných m-tic metodou maximální věrohodnosti, u které mírou podobnosti byl Pearsonův korelační koeficient. (Zeman, 2021)

3.2.1.3 Výsledky

Výsledky pomocí algoritmu x2 testem byly ve shodě s pracemi (Carpinella, 2015). Nejvýhodnější je testovat přítomnost RS gyroskopickým měřením. Přesnějších výsledků je dosazeno při použití měření torzního třesu levé horní končetiny při zavřených očích. Algoritmem nebyla chybně zařazena ani jedna zdravá žena, a to ani pro jednu z m-tic. Populační vzorek obsahoval ve zdravých kontrolách původně dva muže, které algoritmus přiřazoval do skupiny pacientů, a to ve 26 m-ticích u jednoho a v 36 m-ticích druhého. U prvního probanda byla diagnostikována RS tři měsíce po měření a druhý proband testy na RS nikdy nepodstoupil.
Navrhovaný postup je poměrně složitý na výpočet, avšak je zajímavou alternativou ke konvenčním metodám měření RS, a to díky rozšíření chytrých mobilních telefonů či chytrých hodinek, jejichž gyroskopy jsou dostatečně citlivé a rychlé k pořízení záznamu pro analýzu tremoru. Nabízí se zde tedy možnost využití pro screening populace. Studie byla prozatím provedena na malém vzorku. (Zeman, 2021)

3.3 **Android**

Android je celosvětově využívaný open-source operační systém pro mobilní zařízení, tablety a chytré hodinky. Je vyvíjen společností Google a různí výrobci instalují do svých zařízení svoje upravené verze, které doplňují o název prostředí.
Při vývoji byl kladen důraz na možnost spouštět systém na různém hardwaru, aby bylo využití operačního systému co nejširší. Vznikl tedy systém, jenž může být použit na zařízeních s různou čipovou sadou, velikostí obrazovky a hardwarovou platformu.

3.3.1 **Java**

Javu vyvinula společnost Sun Microsystems jako objektově orientovaný jazyk pro podnikové aplikace a pro interaktivní aplikace na internetu. Mezi výhody, díky nimž se Java v posledních letech stala tak populární, patří její bezpečnostní prvky a skutečnost, že je architektonicky neutrální. To znamená, že lze pomocí Javy napsat program, který běží na jakékoli platformě (operačním systému). Java může být spuštěna na široké škále počítačů, protože neprovádí pokyny přímo v počítači, ale na virtuálním stroji, kterému se říká Java Virtual Machine (JVM). Java se v JVM nespouští přímo, nejprve je třeba ji přeložit do tzv. Java bajtkódu (Java bytecode).
Pojem Java se však používá ve dvou významech: programovací jazyk a programovací rozhraní. Programovací jazyk určuje pravidla pro zápis programů. Oproti tomu programovací prostředí tzv. aplikační programové rozhraní (API) je kód, který je k dispozici při psaní programů. Javovské API se skládá z tříd, které jsou uspořádány do balíků. (Herout, 2007)
Do roku 2019 byla Java preferovaným jazykem pro programování Android aplikací.

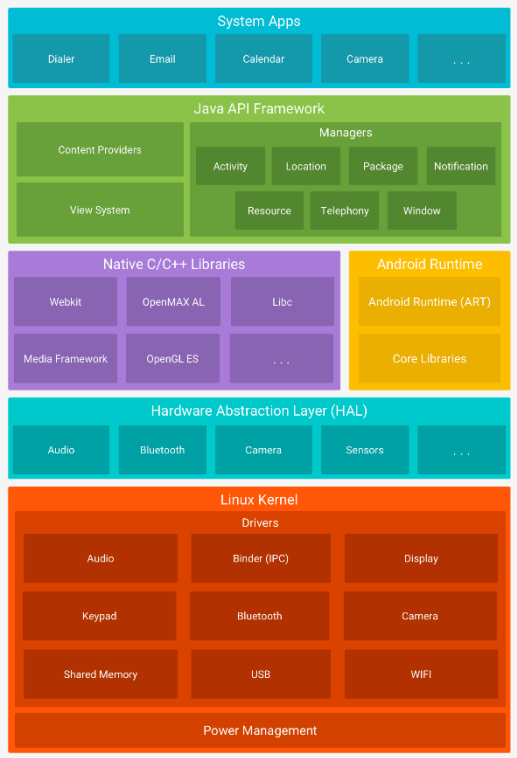
3.3.2 **Kotlin**

Kotlin je pragmatický programovací jazyk pro JVM, Android a JavaScript. Jedná se o open-source jazyk, vytvořený společností JetBrains. Google od roku 2019 považuje Kotlin jako preferovaný jazyk pro programování Android aplikací. Jako jedna z jeho předností se uvádí především interoperabilita s Javou. To znamená, že lze v Kotlinu použít jakýkoli Java kód

nebo knihovnu, a proto je také umožněno využívat v jednom projektu jak Javu, tak Kotlin. Druhou zásadní změnou oproti Javě je, že je tzv. null safety jazykem, což znamená, že během kompilace pracuje s možnými nulovými situacemi a nemůže tedy dojít k výjimkám při nulových hodnotách proměnných. (Subramaniam, 2019)

3.3.3 Architektura systému Android

Operační systém Android se skládá z pěti částí. Linuxové jádro, Hardwarová vrstva, Android Runtime, Nativní knihovny C / C ++, Java Api a Systémové aplikace viz obrázek 3.4.



Obrázek 3.4 Diagram zásobníku softwaru Android (Developer.android.com, 2020)

Linuxové jádro
Operační systém Android je postaven na linuxovém jádře. Je tedy využito mnoho vlastností z tohoto systému, jako je podpora správy paměti, správa sítí, běh souběžného počtu aplikací, práce s oprávněními nebo zabudované ovladače. (Developer.android.com, 2020)

Hardwarová vrstva
Hardwarová vrstva (Hardware Abstraction Layer) poskytuje standardní rozhraní, které umožňuje práci s hardwarem na vyšší úrovni v rámci Java API. Skládá se z několika knihoven, kde každá implementuje rozhraní pro konkrétní typ hardwarové komponenty, jako je např. Bluetooth nebo kamera. Když je zavolán přístup k danému hardwaru, systém Android načte příslušnou knihovnu pro tuto komponentu. (Developer.android.com, 2020)

Android Runtime
Od Android verze 5.0 běží každá aplikace ve svém vlastním procesu a se svou vlastní instancí Android Runtime (ART). Ta umožňuje, aby bylo spuštěno více virtuálních strojů na zařízeních s nízkou pamětí spuštěním souborů DEX, což jsou formáty bytecode navržené speciálně pro Android. Pomocí nich lze nejprve kompilovat zdroj Java do bytecodu a následně kompilovat do kódu DEX speciálním. Výsledkem je vyšší výkon a menší spotřeba baterie. (Developer.android.com, 2020)

Mezi tři hlavní rysy ART patří:

- Ahead-of-time (AOT) a just-in-time (JIT) kompilace
- Vyladění výkonu pomocí garbage collection
- Lepší podpora ladění

Nativní knihovny C / C ++
Mnoho komponent a služeb systému Android jsou napsány v jazycích C / C++. Platforma Android poskytuje rozhraní pro implementování některých funkcí z těchto knihoven. Pomocí nich je možno například přistupovat k OpenGL knihovnám prostřednictvím rozhraní Java OpenGL API a přidat podporu pro kreslení a manipulaci s 2D a 3D grafikou do dané aplikace. (Developer.android.com, 2020)

Java API
V operačním systému Android je k dispozici celá řada funkcí prostřednictvím rozhraní API, napsaných v jazyce Java. Tato rozhraní tvoří stavební bloky, jenž jsou potřeba k implementaci aplikací pro Android.

Mezi základní komponenty patří:

- Rozšiřitelný systém zobrazení (View system), jenž je možno použít k vytvoření uživatelského rozhraní aplikace, včetně textových polí, seznamů, mřížek a tlačítek.
- Manažer prostředků (Resource Manager) poskytuje přístup ke zdrojům, jako jsou lokalizované řetězce, grafiky nebo k uspořádání souborů.
- Manažer notifikací (Notification manager) umožňuje přístup k upozorněním aplikací a možnost jejich zobrazení na displeji nebo ve stavovém řádku.
- Správce aktivit (Activity manager) spravuje životní cyklus aplikací.
- Poskytovatel obsahu (Content Providers) umožňuje přístup k datům z jiných aplikací např. z nativní aplikace Kontakty.

Vývojáři mají plný přístup ke stejným rozhraním API, která používají systémové aplikace pro Android. (Developer.android.com, 2020)

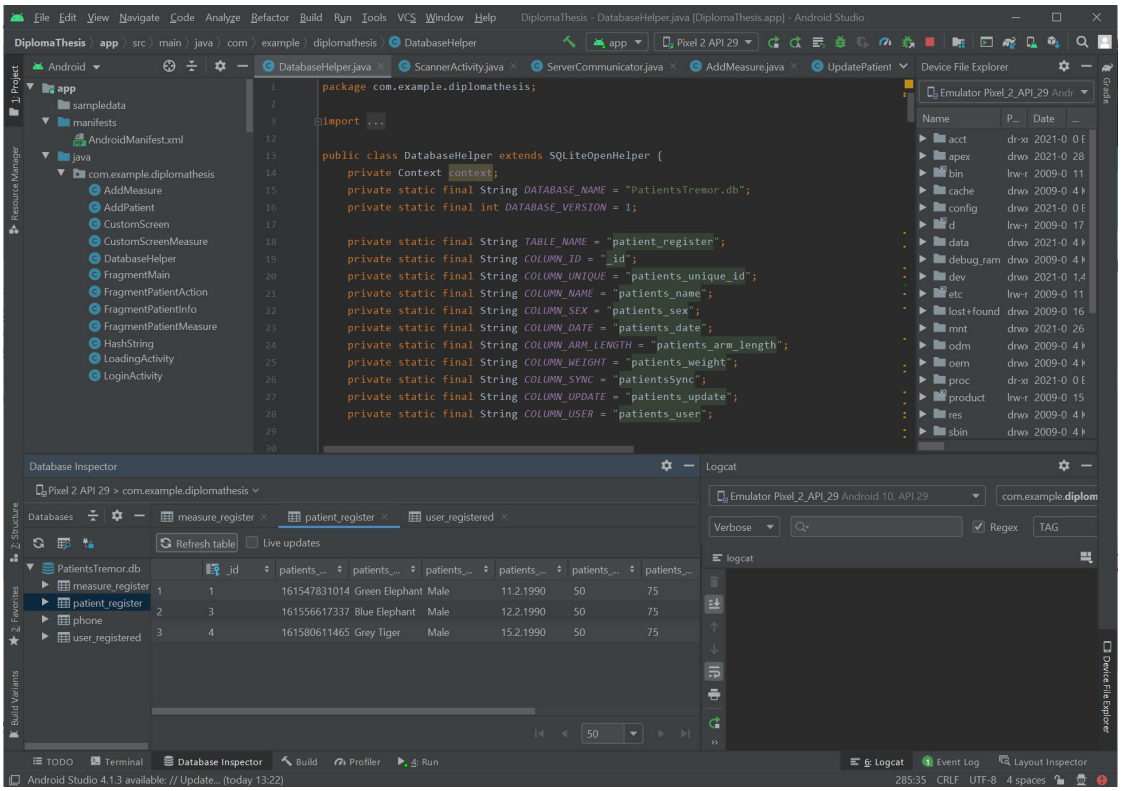
Systémové aplikace
Android vyvíjí a spravuje vestavěné aplikace, jenž je možno využít naskrz všemi verzemi systému. Jedná se o aplikace Email, Odesílání SMS zpráv, Kalendář, Internetový prohlížeč, Kontakty a další.
Systémové aplikace fungují jako aplikace pro uživatele i jako důležité funkce, díky nimž mají vývojáři přístup ze své vlastní aplikace k funkcím, které by jinak museli složitě implementovat.

3.3.4 **Android studio**

Android studio je v současnosti nejvhodnější IDE pro vývoj aplikací pro Android. Je k dispozici zdarma a je založeno na softwaru JetBrains IntelliJ IDEA. První stabilní verze Android studio byla vydána v prosinci 2014 a nahradila Eclipse (s Android Development Tools) jako primární IDE pro vývoj Androidu. Od té doby se Android studio stalo stabilním, rychlým a moderním IDE. Jeho velkou výhodou oproti konkurenci je nadstavba Gradle. Jedná se o nástroj pro automatizaci projektů a k vytvoření konfigurace projektu. Dále pak nástroje pro práci s layouty a s fragmenty, propojený emulátor nebo možnost spouštět aktualizované aplikace přímo v mobilním telefonu. (Van Drodgelen, 2015) V diplomové práci jsem především ocenil vestavěný nástroj Database inspector pro živý náhled lokální sqlLite databáze jak v emulátoru, tak v připojeném mobilním telefonu.

3.3.4.1 Uživatelské rozhraní

Uživatelské rozhraní Android studia se skládá z několika oken, která si může uživatel poskládat podle svých preferencí. V základním rozložení se jedná o okno struktury projektu a zobrazení otevřených souborů. Volitelně nastavená rozložení však umožňují mít současně otevřeny přehledy lokání databáze aplikace, emulátor Android systému, souborový strom přímo v systému, terminál apod. Příklad rozložení v programu Android studio můžeme vidět na obrázku 3.5.



Obrázek 3.5 Rozložení oken v programu Android studio (Zdroj: vlastní zpracování)

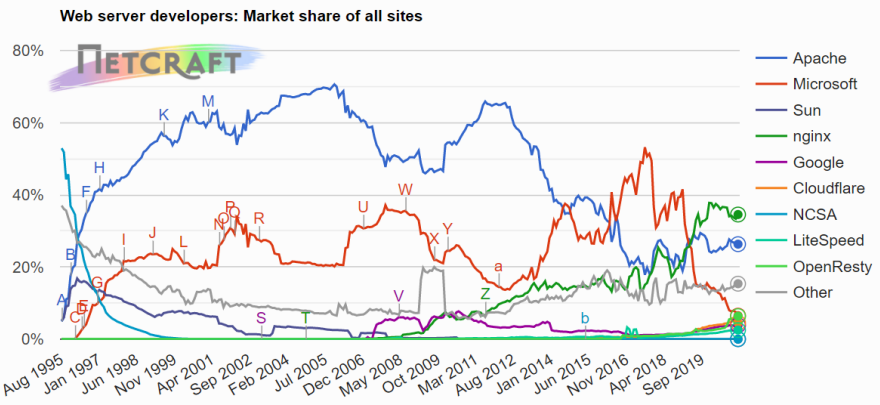
3.4 Webové servery

3.4.1 Apache

Apache Server je aplikace webového serveru, která poskytuje obsah jako stránky HTML, multimédia a šablony stylů CSS přes internet. Apache je komunitní webová aplikace publikovaná Apache Software Foundation. Je historicky nejpopulárnější software webového serveru, který je k dispozici. Nejčastěji se vyskytuje v operačních systémech založených na Unixu, jako jsou Linux, OSX, Solaris a FreeBSD. Je dobře optimalizován a zvládne poměrně velké zatížení při přenosu dat při minimálních hardwarových požadavcích.

3.4.2 NGINX

NGINX je open-source software pro webové služby, reverzní proxy, ukládání do mezipaměti, vyvažování zátěže a streamování médií. Původně byl vyvinut jako software pro vyřešení problému c10k, což je problém, který měly webové servery při větším počtu souběžných připojení. Díky jeho asynchronní architektuře založené na událostech však způsobil revoluci ve fungování webových serverů, a stal se nejrychleji se rozvíjejícím webovým řešením. V roce 2021 se stal, co do používání, lídrem na trhu viz obrázek 3.6. Často se také využívá jako frontend proxy pro Apache, a díky tomu lze využívat výhody obou řešení.



Developer	January 2021	Percent	February 2021	Percent	Change
nginx	399,330,927	33.33%	415,900,479	34.54%	1.20
Apache	316,046,149	26.38%	316,992,638	26.32%	-0.06
Microsoft	89,781,136	7.49%	78,331,379	6.50%	-0.99
OpenResty	74,385,487	6.21%	76,623,440	6.36%	0.15

Obrázek 3.6 Využívání webových serverů dle technologie (Netcraft.com, 2021)

3.4.3 **PHP**

PHP (Hypertext Preprocessor) je široce používaný open source univerzální skriptovací jazyk, jenž je zvláště vhodný pro vývoj webových aplikací, a který může být vložen i do HTML souborů.
PHP se liší od HTML a JavaScriptu především v tom, že kód je spuštěn na serverové části, kde generuje HTML, které je až poté odesláno klientovi. Výhoda tohoto řešení je, že se klient po obdržení výsledků provedeného skriptu nedozví, jaký byl základní kód, ale vidí pouze jeho HTML část. PHP se zaměřuje především na skriptování, lze ale pomocí něj i shromažďovat data formulářů, generovat dynamický obsah nebo odesílat a přijímat soubory cookies. (Bakken, 2000) PHP se používá ve třech hlavních oblastech:

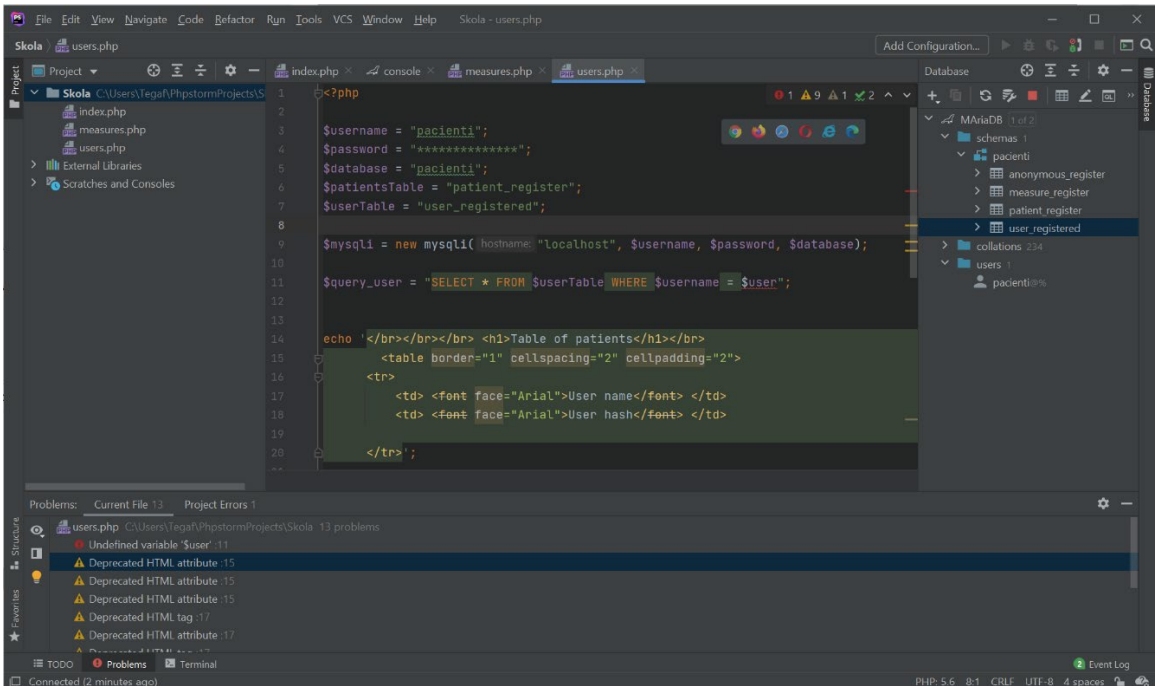
Skriptování na straně serveru
Jedná se o nejčastější využití. K tomu, aby vše fungovalo, je zapotřebí třech faktorů: analyzátor PHP (CGI nebo modul serveru), webový server a webový prohlížeč. K výstupu PHP je možno přistupovat pomocí webového prohlížeče nebo prohlížet stránku PHP přes server.

Skriptování z příkazového řádku
Lze také vytvořit skript PHP, jenž se následně bude spouštět bez webového serveru nebo prohlížeče. K využití tohoto způsobu je zapotřebí pouze parser PHP. Tento typ použití je ideální pro skripty pravidelně spouštěné pomocí Cronu (na Linuxových systémech) nebo Plánovače úloh (na systému Windows). Vytvořené skripty lze také použít pro jednoduché úlohy zpracování textu.

Psaní desktopových aplikací
Pomocí PHP-GTK lze také psát v PHP desktopové programy. Takovéto využití jazyka PHP je ovšem trochu nestandardní, a proto se přistupuje se k tomuto řešení jen při výjimečných situacích.

3.4.3.1 PHPstorm

PHPStorm je integrované vývojové prostředí založené na Javě IDE. Je vytvořeno společností JetBrains pro PHP a webové vývojáře. Podporuje PHP od verze 5.3 a poskytuje prevenci chyb za chodu, automatické dokončování, ladění nulové konfigurace a rozšířený editor HTML, CSS a JavaScript. Mezi jeho výhody patří, že podporuje integraci s mnoha redakčními systémy jako je Drupal, Wordpress apod. Dokončování kódu se netýká pouze PHP ale také HTM, CSS a JavaSkriptu, což usnadňuje práci při psaní webových aplikací. PHPStorm má také integrovanou podporu QGL databází, což umožňuje se k databázi nejen připojit, ale upravovat tabulky, spouštět příkazy, a dokonce analyzovat schémata pomocí UML diagramů.
Vzhledem k tomu, že patří do stejné skupiny jako Android studio, sdílí spolu podobný vzhled, rozložení a klávesové zkratky. Rozhraní programu viz obrázek 3.7.



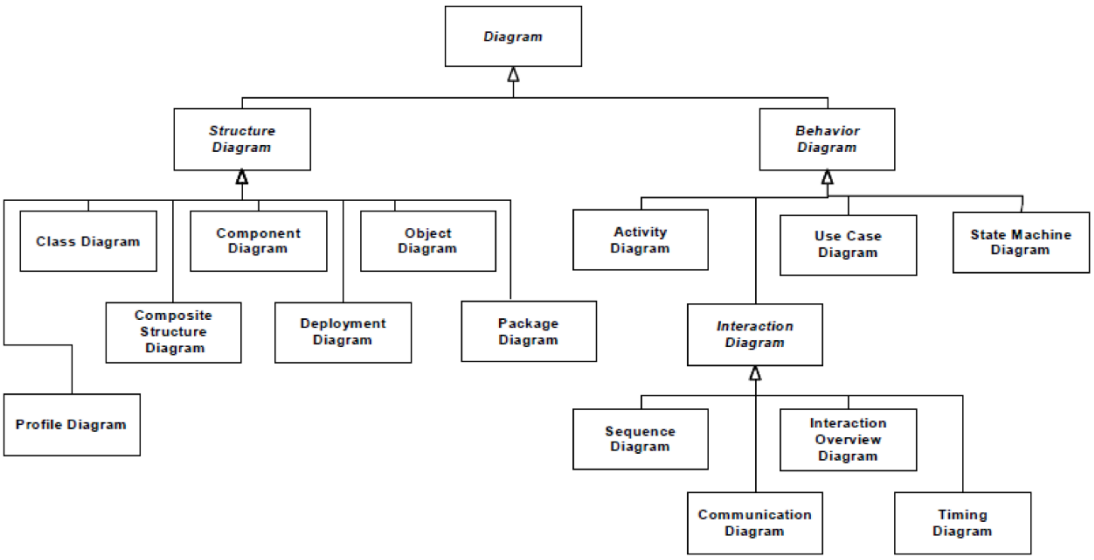
Obrázek 3.7 Rozložení oken v programu PHPStorm (Zdroj: vlastní zpracování)

3.5 UML

Unified Modeling Language (UML) je obecný modelovací jazyk. Hlavním cílem UML je definovat způsob vizualizace návrhu systému. Nejedná se o klasický programovací jazyk, jde spíše o soubor grafických notací. Výstupem návrhu systému je UML diagram, jenž pomáhá softwarovým inženýrům, programátorům a systémovým architektům s modelováním, designem a analýzou systémů. UML byl mezinárodní organizací pro normalizaci (ISO) schválen jako norma v roce 2005. Od té doby byl revidován a je pravidelně aktualizován. (Omg.org, 2017)

3.5.1 UML diagramy

UML diagramy jsou obecně kategorizovány do dvou skupin na diagramy struktur (Structure diagram) a diagramy chování (Behavior diagram). Strukturní diagramy se skládají ze sedmi statických diagramů, jako jsou například diagramy tříd (Class diagram), objektů (Object diagram), komponent (Component diagram) atd. Oproti tomu diagramy chování se skládají také ze sedmi, avšak dynamických diagramů, jako jsou sekvenční diagram (Sequence diagram), diagram případů užití (Use case diagram), aktivity diagram (Activity diagram) atd. Úplné rozdělení diagramů viz obrázek 3.8 .



Obrázek 3.8 Strukturní rozdělení UML diagramů podle kategorií (Omg.org, 2017)

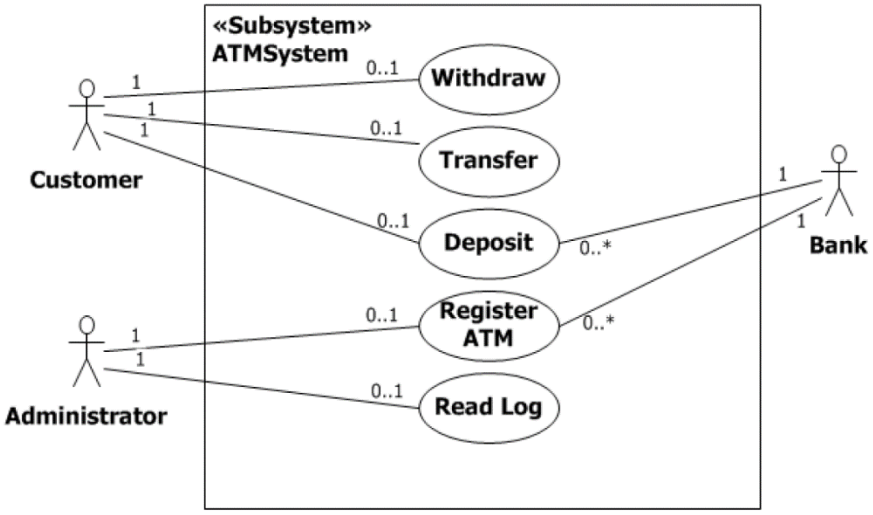
3.5.1.1 Use Case

Use Case, v češtině někdy nazýván jako diagram případů užití, je druh diagramu, který představuje deklaraci sady nabízeného chování systému a jeho interakci s okolím (s Aktéry). Základními kameny diagramu jsou funkční požadavky, které systém musí splňovat. Diagramy případů užití se používají k analýze požadavků na vysoké úrovni systému. Tyto požadavky jsou vyjádřeny prostřednictvím různých Use Case. Diagram se skládá ze tří hlavních komponent:

- Use case - představují funkční požadavky. Jedná se o sloveso, popisující akci.
- Aktéři - interagují se systémem. Aktérem může být člověk, organizace nebo interní / externí aplikace.
- Vztahy mezi aktéry a use case – znázorňují se pomocí přímých čar.

Každá use case určuje určité chování, které může subjekt provádět ve spolupráci s jedním nebo více aktéry. Use case definují nabízená chování subjektu bez odkazu na jeho vnitřní strukturu. Tato chování, zahrnující interakce mezi aktéry a subjektem, mohou mít za následek změny stavu subjektu a komunikace s jeho prostředím. Use case může zahrnovat možné varianty základního chování včetně výjimečného chování a zpracování chyb. (Omg.org, 2017)

Use case je zobrazena jako elipsa, která buď obsahuje název Use Case nebo s názvem umístěným pod elipsou. Volitelná stereotypní klíčová slova mohou být umístěna nad jménem Use Case. Hranice systému je zobrazena jako obdélník se svým názvem v levém horním rohu, s use case elipsami vizuálně umístěnými uvnitř tohoto obdélníku, viz obrázek 3.9. (Omg.org, 2017)



Obrázek 3.9 Struktura use case diagramu (Omg.org, 2017)

3.5.1.2 Use case specifikace

Use case specifikace je připojena k diagramu případů užití, nemá předepsanou formu, ale bývá formou tabulky nebo prostého textu. Obsahuje podrobnosti k jednotlivým případům užití. Části specifikace jsou:

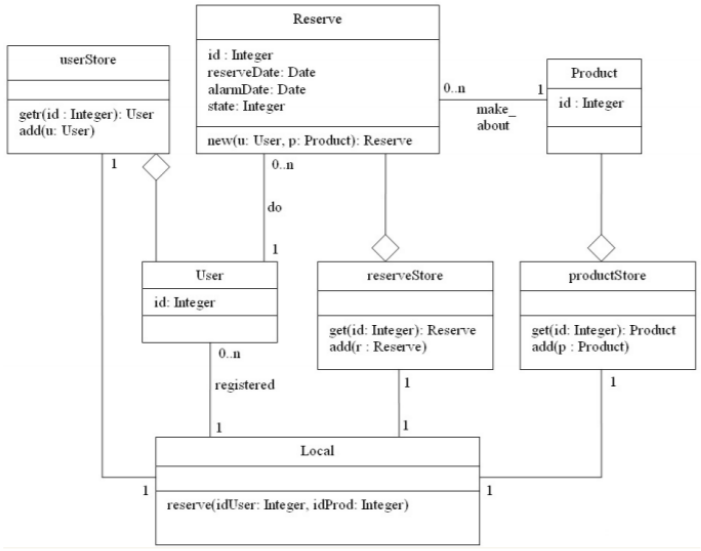
- Stručný popis v několika větách vysvětlující funkce poskytované případem užití a hodnotou pro uživatele, který jej aktivuje.
- Výpis všech aktérů, kteří se na případu užití podílejí.
- Podmínky před spuštěním případu užití, bez kterých by spuštění ztrácelo smysl.
- Základní tok, který v bodech vypisuje interakci mezi aktéry a případy užití.
- Jeden až několik alternativních toků (scénářů), které umožňují reagovat na odchylky od základního toku. Jedná se např. o chyby nebo chyby způsobené uživatelem.
- Podmínky, které je nutno provést ve chvíli, kdy bude případ užití považován za hotový.

3.5.1.3 Diagram tříd

Diagram třídy (Class diagram) UML je nejběžnějším typem diagramu pro dokumentaci softwaru. Jelikož je většina softwaru, který se v dnešní době vytváří stále založen na objektově orientovaném programování (OOP), a diagramy tříd jsou také založeny na třídách a vztazích mezi nimi, umožňují některé programy export diagramu přímo z IDE softwaru. (Martínez, 2005)

Diagramy tříd obsahují třídy spolu s jejich atributy (datovými typy) a jejich metodami. Třídy reprezentují pole s názvem třídy, jež by měly být vycentrovány a pojmenovány tučným písmem. Každá třída má dvě další pole: atributy třídy přímo pod názvem a metody třídy

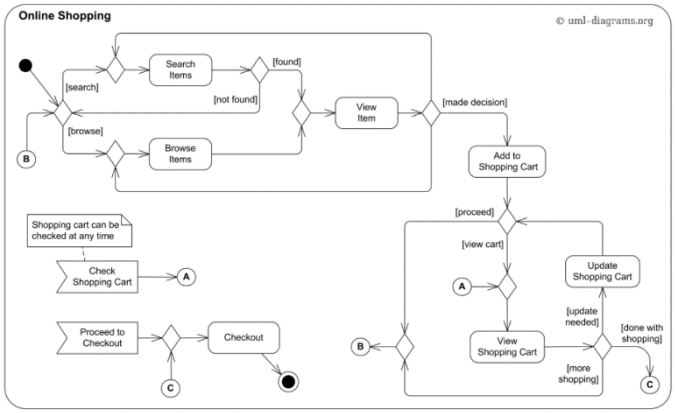
v dolní části. Vztah mezi různými třídami (představovaný spojovací čárou) tvoří diagram tříd. Struktura diagramu tříd viz obrázek 3.10.



Obrázek 3.10 Struktura Class diagramu (Martínez, 2005)

3.5.1.4 Aktivity Diagram

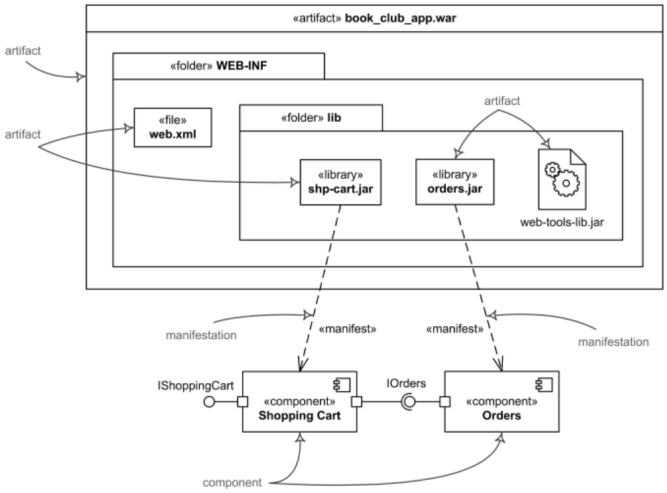
Aktivity Diagram se využívá k zobrazení sledu činností. Diagramy aktivit ukazují pracovní postup od počátečního bodu do konečného bodu s popisem různých cest, kterými se aktivita může dobrat ke svému cíli. Během nich dochází k rozhodování, která jsou v aktivitě popsány. Pokud nám tedy Use case diagram popisoval, co od systému očekáváme, aktivity diagram nám ukazuje cestu, jak se k výslednému kroku můžeme dostat. Uvnitř aktivity se vykonávají akce, které lze vykonávat jak pomocí systému, tak uživatelů. (Fakhroutdinov, 2020) Struktura aktivity diagramu viz obrázek 3.11.



Obrázek 3.11 Struktura aktivity diagramu (Fakhroutdinov, 2020)

3.5.1.5 Diagram nasazení

Diagram nasazení (Deployment diagram) je strukturní diagram, který ukazuje architekturu systému jako nasazení softwarových artefaktů do cílů nasazení. Artefakty představují konkrétní prvky ve fyzickém světě. Příkladem artefaktů jsou spustitelné soubory, knihovny, archivy, databázová schémata, konfigurační soubory atd. Cíl nasazení je obvykle reprezentován uzlem (node), kterým je buď hardwarové zařízení nebo některé prostředí pro provádění softwaru. Uzly lze spojovat prostřednictvím komunikačních cest a vytvářet síťové systémy libovolné složitosti.
Standard UML má předdefinované stereotypy UML jako například <<file>>, <<library>> atd., ale lze si nadefinovat i vlastní. (Fakhroutdinov, 2020) Strukturu diagramu nasazení viz obrázek 3.12 .



Obrázek 3.12 Struktura diagramu nasazení (Fakhroutdinov, 2020)

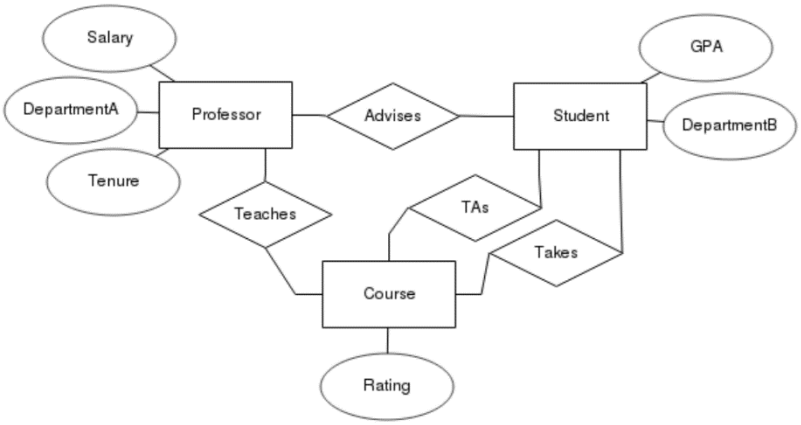
3.6 **E-R diagram**

Diagram vztahu mezi entitami, známý též jako ERD nebo ER model, je typ vývojového diagramu, který ilustruje vzájemně souvisejí „entity“ jako jsou lidé, objekty nebo koncepty. ER diagramy se nejčastěji používají k návrhu relačních databází v oblastech softwarového inženýrství, podnikových informačních systémů, vzdělávání a výzkumu. Používá definovanou sadu symbolů, jako jsou obdélníky, diamanty, ovály a spojovací čáry, které zobrazují vzájemnou propojenost entit, vztahů a jejich atributů. E-R diagram zrcadlí gramatickou strukturu s entitami jako podstatná jména a vztahy jako jsou slovesa. Databáze je možno modelovat jako množinu entit, nebo jako vztah mezi entitami. (Hayes, 2017) Strukturu E-R diagramu viz obrázek 3.13.

Základní pojmy E-R diagramu jsou:

- Entita - objekt reálného světa rozlišitelný podle jiných objektů.
- Entitní množina – množina entit téhož typu, které sdílejí stejné vlastnosti.
- Atribut – vlastnost entity, která je předmětem zájmu.

- Vztah – asociace mezi několika entitami.
- Množina vztahů - množina vztahů stejného typu sdílící tytéž vlastnosti.



Obrázek 3.13 Struktura E-R diagramu (Hayes, 2017)

3.7 **SQL databáze**

SQL databáze je kolekce tabulek, která se ukládá do zvláštního souboru strukturovaných dat. Umožňuje přístup k několika záznamům z jediného příkazu, který nevyžaduje specifikaci, jak se lze k němu dostat. K vytváření dotazů, manipulaci s daty, definování dat a k řízení výstupu se využívá SQL programovací jazyk.

3.7.1 **SQL programovací jazyk**

SQL (Structured Query Language) se používá ke komunikaci s databázemi. Podle ANSI (American National Standards Institute) se jedná o standardní jazyk pro správu relačních databází. Příkazy SQL se používají k provádění úkolů, jako je aktualizace dat v databázi, načítání dat z databáze, mazání dat nebo jejich úprava. Některé běžné systémy pro správu relačních databází, které používají SQL, jsou: Oracle, Sybase, Microsoft SQL Server, MariaDB atd. Ačkoli většina databázových systémů používá SQL, mnoho z nich mají také své vlastní další proprietární rozšíření, která se obvykle používají pouze v jejich systému. Standardní příkazy SQL jsou „SELECT", "INSERT", "UPDATE", "DELETE", "CREATE", a "DROP". Ty se vyskytují ve všech variantách databází, a poskytují dostatečný prostor na splnění základních procedur.

3.7.2 **MariaDB server**

MariaDB Server je jedním z nejpopulárnějších databázových serverů na světě. Na jeho tvorbě se podílejí původní vývojáři MySQL serveru. Je vydáván pod licencí open-source a jako relační databáze poskytuje přístup k datům pomocí jazyka SQL. Tento software

využívají například společnosti Wikipedia, WordPress.com a Google. Původně byl navržen jako alternativa k MySQL. Podle tvůrců bylo zapotřebí vytvořit rychlý, škálovatelný a robustní systém, s možností vytvářet a přidávat moduly. Vznikl tak velmi univerzální nástroj pro širokou škálu využití. (Mariadb.org, 2015)

3.7.3 SQLite

SQLite je knihovna, která implementuje samostatný, bezserverový, transakční databázový nástroj s nulovou konfigurací. Kód SQLite knihovny je open-source, je tedy zdarma pro jakékoli účely, jak komerční, tak i soukromé. Jedná se o nejrozšířenější databázovou knihovnu na světě.
Na rozdíl od většiny ostatních databází SQL nemá databáze SQLite na serveru samostatný proces, ale čte a zapisuje data přímo do jednoho souboru umístěného na disku. Ten obsahuje veškeré potřebné informace k práci s databází. Formát databázového souboru je multiplatformní a lze jej libovolně kopírovat mezi systémy nebo mezi architekturami a je velmi pečlivě testován před vydáváním nových verzí. (Sqlite.org, 2018)

3.8 Zabezpečení

3.8.1 OpenVPN

OpenVPN je software, pomocí kterého je možné vytvářet šifrované VPN tunely mezi hostitelskými stanicemi. Je založen na architektuře klient-server a je schopný zajistit přímou komunikaci mezi počítači bez možnosti odposlouchávání. Byl vyvinut v roce 2001 jako open-source projekt, kolem kterého vznikla komunita programátorů, kteří protokol pravidelně testují, vylepšují a aktualizují. Jelikož se nejedná o proprietární kód (vlastnění konkrétní společností), mají bezpečnostní experti po celém světě ke kódu přístup a mohou kontrolovat jeho bezpečnost.

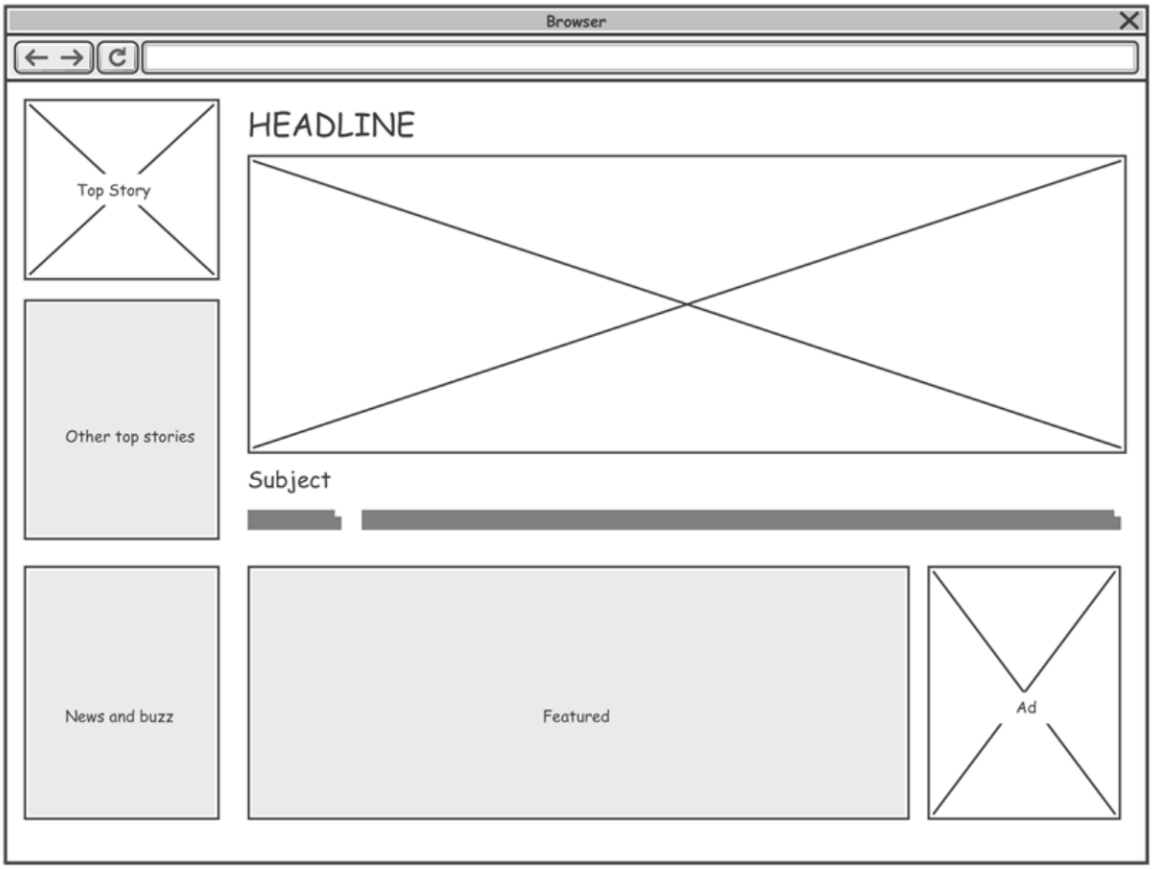
3.9 Návrh designu

3.9.1 Wireframe

Wireframe webu (skica webu) je dvourozměrný obrys webové stránky nebo aplikace, poskytující jasný přehled o struktuře stránky, rozložení, informační architektuře, toku uživatelů, funkčnosti a zamýšleném chování. Wireframe obvykle představuje počáteční koncept produktu. Styl, barva a grafika jsou omezeny na minimum. Wirframe lze kreslit ručně nebo vytvářet digitálně podle toho, kolik podrobností je vyžadováno. Většinou jej dělíme do čtyř skupin:

Textový wireframe
Jedná se o slovně popsaný obsah popisující funkční prvky webu nebo aplikace.

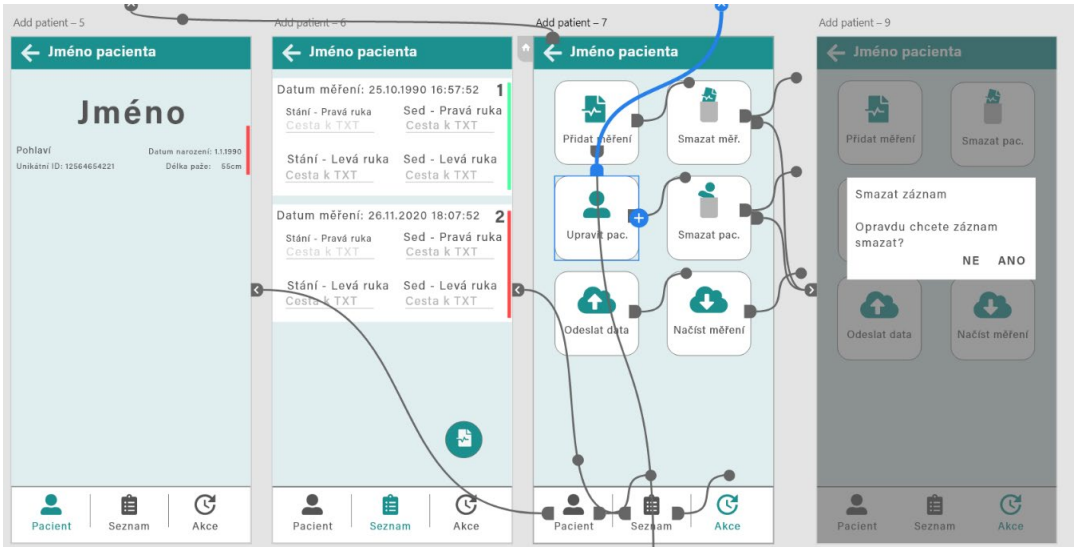
Stručný wireframe
Stručný wireframe může být nakreslen ručně nebo digitálně a ukazuje rozmístění prvků na webu nebo v aplikaci, viz obrázek 3.14 . Používá se u jednodušších řešení a jeho výroba není technicky náročná.



Obrázek 3.14 Stručný wireframe (Visual-paradigm.com, 2016)

Podrobný wireframe
Jedná se o propracovaný návrh webové stránky nebo aplikace. Obsahuje již ostré texty, obrázky a tabulky. Podrobně popisuje chování a napojení různých částí. Používá se u komplikovanějších řešení, které je potřeba zpracovat podrobně.

Aktivní wireframe
Jde o speciální případ, kde jsou různé wireframe propojeny navzájem a tvoří funkční celek s klikatelnými odkazy, prvky reagujícími na pohyb myší, animacemi a dalšími interaktivními prvky, viz obrázek 3.15. V dnešní době je již velká řada softwaru nabízejícího jednoduché zpracování prototypů jako AdobeXD, UXPin, Axure RP nebo Sketch.

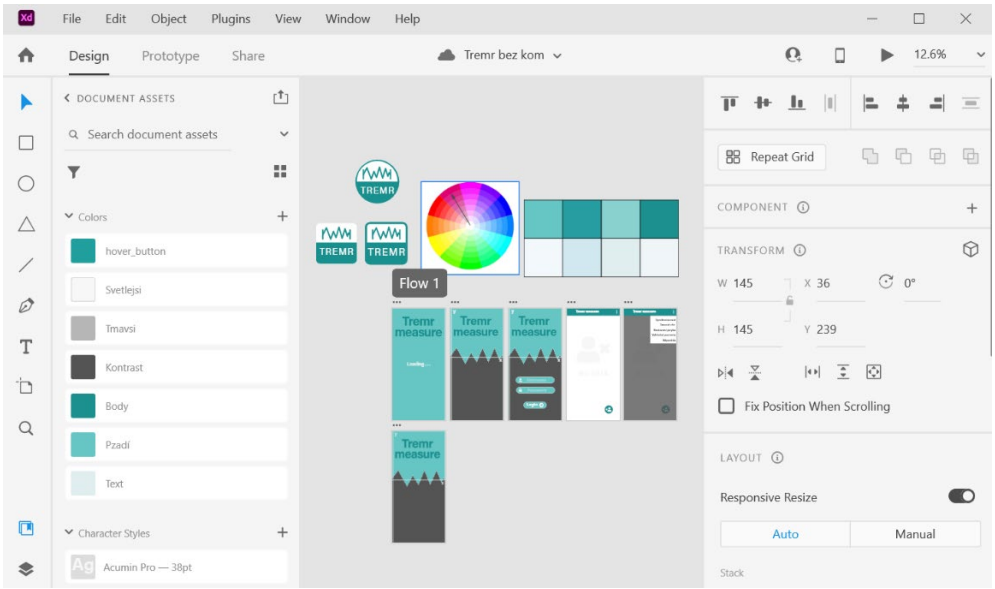


Obrázek 3.15 Aktivní wireframe (Zdroj: vlastní zpracování)

Wireframing je praxe, kterou nejčastěji používají návrháři UX. Tento proces umožňuje všem zúčastněným stranám se dohodnout, kde budou různé komponenty umístěny dříve, než vývojáři vytvoří rozhraní s kódem.

3.9.1.1 Adobe XD

Adobe XD je vektorový editor pro návrh wireframe, uživatelských UX a designu webových stránek aplikací, to vše v jednoduchém rozhraní viz obrázek 3.16. Je vyvíjen společností Adobe od roku 2015 a od roku 2018 jej společnost Adobe uvolnila pro bezplatné využívání. Mezi jeho přednosti patří kopírování komponent, široká škála doplňků, prototype mód, možnost sdílet návrh odkazem a spouštění prototypů přímo v mobilních zařízeních.



Obrázek 3.16 Rozložení programu Adobe XD (Zdroj: vlastní zpracování)

4 Vlastní práce

Cílem diplomové práce je vytvořit komplexní registr pacientů, jenž bude ukládat data pacientů a jejich diagnózy do lokální databáze mobilního telefonu a zároveň bude mít možnost odeslat tato data na centrální server. Jako součást práce musí být vytvořeno webové rozhraní pro přístup k datům na centrálním serveru. Pro snímání třesu může být implementováno měřící zařízení MetaWear od společnosti MbientLab.
V této části bude popsán průběh návrhu a vytvoření aplikace pro snímání třesu. Nejprve budou popsány požadavky na software a popis navrhovaného řešení. Navrhované řešení bude podrobeno analýze, ve které budou zvoleny vhodné postupy a nástroje, které budou k vytvoření aplikace využívány.
Následně budou zpracovány behaviorální, strukturální a datové diagramy, které budou vycházet z provedené analýzy. Pro přehled chování systému bude využit Use Case diagram, který bude následně upřesňován Use Case specifikací s příslušnými Aktivity diagramy. Dále bude vymodelován doménový diagram a diagram tříd, který bude doplněn přehledem aktivit a fragmentů.
V další části bude popsán návrh grafické podoby aplikace a webového rozhraní.
Následovat bude samotná implementace, kde budou ukázány různé části kódu a popsáno jejich použití. Na závěr této kapitoly bude ukázána reálná podoba aplikace a webového rozhraní.

4.1 Požadavky na software

Abychom mohli definovat požadavky na software, musíme si nejprve přiblížit okolnosti těchto požadavků. Dosavadní technické řešení sběru dat je určeno pro malý počet dobrovolníků v jednom odběrovém místě. Je tedy nutné vytvořit komplexní registr pacientů, jenž bude ukládat data pacientů a jejich diagnózy do lokální databáze mobilního telefonu nezávisle na prostředí a počtu pacientů. Zároveň je zapotřebí umožnit snadné měření třesu a následně umožnit automaticky odeslat tato data na centrální server. Navrhovaný systém musí být uživatelsky přívětivý a nesmí k obsluze vyžadovat technické vzdělání.

Obecné požadavky
Aby se obsluze nekomplikovala práce, je jedním z požadavků vytvoření profilu pacienta bez delších pasáží, které by zdravotnický personál musel vypisovat ručně. Při tvorbě systému je také zásadní otázka zpracování dat a jejich následné ukládání. Po konzultaci s panem doktorem Zemanem bylo rozhodnuto, že přestože budou k transferu dat použity kryptovací funkce, není nutné ukládat informace podléhající obecnému nařízení na ochranu osobních údajů, tedy směrnici GDPR. Data pacientů se budou uchovávat pouze v anonymizované podobě. Součástí požadavku taktéž je, aby bylo možno k datům na centrálním serveru jednoduše přistupovat pomocí webového rozhraní. (Zeman, 2020)

Mobilní aplikace
Mobilní aplikace bude vytvořena pro operační systém Android. Prototyp bude otestován na reálném zařízení. Aplikace by měla mít dvě jazykové mutace. Webové rozhraní může být pouze v češtině. Návrh designu bude předem konzultován.

Pacient
Každý pacient bude mít své unikátní ID, pod kterým budou uloženy jeho měření. Pacient bude mít v systému svoji přezdívku a budou o něm nasbírány následující údaje: datum narození, pohlaví, délka paže a hmotnost. Přezdívka bude náhodně generována. Po vytvoření pacienta musí existovat možnost pacienta upravit, odstranit nebo odeslat na centrální server. Všechny políčka budou jednoduše nastavitelná (např. posuvníky), ale zároveň bude poskytnuta možnost každý údaj upravit ručně.

Měření
Každé měření bude mít unikátní ID a bude přiděleno k pacientovi. Ke každému pacientovi musí být možnost přidat i více měření. Měření bude obsahovat následující informace: datum měření a čtveřici naměřených souborů. Soubory se budou přidávat ze souborové struktury mobilního zařízení. Pro měření bude existovat možnost průběžného ukládání v off-line režimu. Po připojení k internetu bude možnost data hromadně odesílat na server. Jako volitelná součást požadavku je prozkoumání možnosti připojit k aplikaci snímací zařízení přes bezdrátovou technologii, a případně implementovat ukládání souborů z takovéhoto zařízení přímo do aplikace.
K tomuto účelu je možno využít zařízení MetaWear od společnosti MbientLab. Pokud bude zapracována komunikace se zařízením, je zde požadavek na diodovou odezvu při měření, aby měl pacient i obsluha se zařízením kontakt.

Přihlašování uživatelů
Dalším požadavkem je vytvořit blíže nespecifikovaný systém přihlašování a kontroly uživatelů. Ti se pomocí přihlašovacích údajů budou moci do aplikace připojovat a odesílat data na centrální server.

Požadavky na webové rozhraní jsou:

Webové rozhraní
Požadavkem je možnost zobrazení seznamu pacientů a jejich měření včetně možnosti stažení souborů u jednotlivých měřeních. Dále bude vytvořen systém umožňovat přidání a smazání uživatelů, kteří budou mít k aplikaci přístup. Tento přístup bude zajištěn pouze pro pana doktora Zemana a jeho spolupracovníky, tudíž není nutné konzultovat design. (Zeman, 2020)

4.2 **Popis řešení**

Na základě požadavku bude vytvořena mobilní aplikace pro systém Android, do které se uživatel přihlásí pomocí jména a hesla. Není potřeba, aby byli uživatelé rozdělení podle práv, protože uživatelé nebudou mít možnost přidávat jiné uživatele. Uživatel bude mít možnost vytvořit pacienta, upravit pacienta, smazat pacienta, vytvořit měření, upravit měření, smazat měření, odeslat pacienta na centrální server a odeslat měření na centrální server.
Pacienti nebudou vázáni na uživatele, ale budou vázáni na mobilní zařízení. Je to z toho důvodu, že zařízení přítomné na pracovišti nebude sloužit pouze pro jednoho uživatele, ale často pro více uživatelů, kteří ale mají přístup ke stejným pacientům.

Uživatelům bude umožněno vyhledávat pacienty na centrální databázi a následně přidávat do zařízení. Tím se zajistí, že pokud by uživatel chtěl do svého zařízení přidat nějakého pacienta z jiného zařízení, bude mu to umožněno.
Dle zadání aplikace nemá podléhat GDPR, proto jméno pacienta musí být vždy pouze přezdívkou a nikoli reálným jménem. Tím zajistíme anonymizaci a lze dodržovat pouze standardní postupy při zabezpečení databáze a zařízení a není nutné přistupovat k vyšším formám zabezpečení (šifrování zařízení, šifrování serveru, monitorovat přístup k datům, pacientovi umožnit výmaz z databáze atp.). Přesto bude při tvorbě zabezpečení dbán důraz na to, aby případná modifikace systému na GDPR kompatibilní byla co nejjednodušší.
Měření budou také vázána na mobilní zařízení, ale oproti pacientům nebudou přenositelná. Z důvodu logistiky a synchronizace není žádoucí, aby se naměřené soubory přenášely z centrálního serveru zpět na různá mobilní zařízení. Proto je zvolen model, kde lze měření odeslat na centrální server až poté, kdy bude obsahovat čtyři naměřené soubory. Získání souborů bude umožněno buďto z lokálního souborového systému nebo pomocí technologie Bluetooth ze zařízení MetaWear.

Centrální server bude přístupný pouze přes VPN spojení. Pro prvotní přístup do aplikace bude vyžadováno spojení s centrálním SQL serverem, díky čemuž nebude muset být v aplikaci přítomno žádné univerzální přihlášení. Z tohoto důvodu ale bude nutné před prvním spuštěním již mít nainstalován VPN přístup na server.

V průběhu návrhu bylo lehce pozměněno zadání, a nově bude na webovém rozhraní zřízen přístup i pro uživatele, a nejen pro správce databáze. Přihlašování tedy bude rozdělené podle uživatelských práv, aby bylo možné odlišit uživatele od správců. Uživatelé budou mít přístup na centrálním serveru jen ke svým datům, zatímco správce databáze bude mít přístup ke všem měřením a pacientům. Zároveň bude správce databáze moci přidávat a mazat uživatele a mazat pacienty. Poslední zmiňvaná funkcionalita bude sloužit především pro mazání chybných záznamů. Uživatel nebude mít žádné další funkcionality.

4.3 **Analýza problému**

Součástí analýzy bylo provést průzkum, zda se podobné řešení již na trhu nevyskytuje. Jediné řešení, které bylo při analýze testováno, využívalo aplikaci MetaBase (od společnosti MbientLab), přes kterou se naměřené soubory posílaly na cloud, kde byly dále zpracovávány. Personál však musel vytvářet registraci pacientů zvlášť nebo soubory pojmenovávat dle aktuálního pacienta a cíle měření. Toto řešení bylo tedy označeno jako ne zcela vyhovující, protože obsahovalo množství úkonů, které by musel personál provádět, a tím by docházelo k jeho zahlcování.

Z popisu řešení vyplývá, že jde o velice komplexní a poměrně složitý systém, u kterého je potřeba dodržet obvyklé standardy práce s právy uživatelů, práv aplikace a se zabezpečením dat proti úniku. Z tohoto důvodu je nutné si před samotným návrhem zvolit správné prostředky, díky nímž bude možné všechny výše uvedené standardy dodržet. Zároveň je nutné si předem rozvrhnout, do kterých částí bude mít systém přístup.

4.3.1 Použitý software

- Software, ve kterém se bude programovat aplikační část projektu, bude Android studio.
- Programování bude probíhat v jazyku Java, především pro rozsáhlou databází návodů a řešení.
- Na programování PHP a HTML bude využit program PHPStorm.
- Webový server bude spuštěn na službě NGINX.
- Diagramy budou vytvářeny v aplikaci draw.io.
- O centrální SQL server se postará služba MariaDB.
- Pro náhled a editaci dat bude využit přímý přístup do databáze.
- K vytvoření E-R diagramu bude použit software DBeaver.
- K návrhu wireframes bude využit program Adobe XD.

4.3.2 Práva uživatelů webového rozhraní

	User	Superuser
Zobrazit svá měření	✔	✔
Zobrazit všechna měření	✘	✔
Přidávat uživatele	✘	✔
Mazat uživatele	✘	✔
Mazat pacienty	✘	✔

Tabulka 4.1 Přehled práv uživatelů webového rozhraní (Zdroj: vlastní zpracování)

4.3.3 Dodatečná práva aplikace vůči zařízení

- Přístup k externímu úložišti
- Přístup k internímu úložišti
- Přístup k síti
- Přístup k bluetooth

4.3.4 Použitá externí rozhraní

Získávání dat ze zařízení MetaWear
Konektivita se zařízením MetaWear se bude provádět pomocí externího rozhraní firmy MbientLab, vytvořené pro tyto účely. Jmenuje se MetaWear Android API a umožňuje snadný přístup pro programátory k deskám MetaWear. (Tsai, 2016)

Přenos souborů z aplikace na webový server
K přenosu souborů bude využit projekt OkHTTP. Pomocí tohoto nástroje je možné nahrávat soubory s využitím HTTP příkazů na vzdálený server. (Schimke, 2019)

Transformace souborů a parametrů
Pro transformaci souboru a parametrů do jednoho souboru, který bude reprezentován jako JSON bude využito open-source knihovny GSON. Tato knihovna slouží k převodu objektu Java do jejich JSON reprezentace. (Volkhart, 2016)

Přístup k interní a externí souborové struktuře
V roce 2020 byl společností Google výrazně omezen přístup externích aplikací do souborové struktury. Nyní jsou zpřístupněny pouze části struktury (složka Stažené) a následně jen různé kategorie dle povahy hledaného souboru (média, dokumenty, eknihy apod.). Ihned po tomto omezení začaly vznikat náhradní řešení, které tento problém řeší tak, že podle určité lokality, ve které chce aplikace hledat daný soubor, vybere správnou metodu (správnou adresu souboru) a tu následně použije k načtení souboru. V tomto projektu bude použita třída UriUtils.java, jejímž autorem je Asad Ali Choudhry a vyhovuje potřebám aplikace. (Choudhry, 2020)

Grafické zpracování webového rozhraní
Pro jednodušší zpracování grafického rozhraní bude využit open-source projekt Bootstrap 5. (Otto, 2020)

Odesílání souborů na webový server
Informace pro návrh a implementaci odesílání souborů na webový server byly čerpány z projektu „Image or any File Upload to Server using Retrofit in Android App“. (Abdullah, 2017)

4.4 Diagramy

4.4.1 Use Case model aplikace

Nejprve byl vymodelován Use Case diagram aplikace, který zobrazuje akce, které bude schopen uživatel provádět viz obrázek 4.1. Některé Use Case budou obsahovat stereotyp <<sqlite>>, který vyjadřuje možnost daného Use Case zápisu do lokální SQL databáze. Use Case diagram obsahuje šest aktérů:

Nepřihlášený uživatel
Tento aktér může zadat jméno a heslo do systému, a případně nechat uložit přihlašovací údaje do lokální databáze systému.

Přihlášený uživatel
Tento uživatel může přidávat a odebírat záznamy. Má veškerá práva, která jsou v aplikaci dostupná.

Centrální SQL server
Jedná se o pasivního aktéra, který poskytuje uložená data a přijmutá data zapisuje na vzdáleném serveru přes VPN spojení.

Web server
Pasivní aktér, který přijímá soubory naměřených dat od aplikace a potvrzuje jejich přijetí.

Časovač
Pasivní aktér, který se stará o dobu trvání měření.

MetaWear
Tento pasivní aktér poskytuje data o měření třesu.

Obrázek 4.1 Use Case diagram aplikace (Zdroj: vlastní zpracování)

4.4.2 Use case scénáře aplikace

V této kapitole budou popsány vybrané Use case scénáře a některé aktivity diagramy. Jedním z požadavků je možnost fungování aplikace v off-line režimu. V případě, že zařízení není připojeno k příslušné VPN síti, je vždy uživatel na tento stav upozorněn. Tento alternativní tok bude popsán jen v první specifikaci UC1 Přihlásit. Dále nebude ve specifikacích zmiňován.

4.4.2.1 UC10 Přihlásit

Stručný popis
Tato Use Case umožňuje se nepřihlášenému uživateli přihlásit. Přihlášením získá uživatel úplný přístup do aplikace.

Výpis aktérů

- Uživatel
- Systém
- Centrální SQL server

Podmínky před spuštěním

- Uživatel není přihlášen.

Základní tok

1. Systém vyzve uživatele k přihlášení do systému.
2. Uživatel vyplní přihlašovací údaje a odešle přihlašovací formulář.
3. Systém porovná uživatelské jméno s lokální databází.
4. Systém porovná heslo s lokální databází.
5. Systém přihlásí uživatele do aplikace.

Alternativní tok bodu 3

3.1 Pokud systém nenalezne uživatele v lokální databázi, připojí se k centrálnímu SQL serveru.
3.2 Systém provede validaci uživatelského jména s údaji na serveru a vyzve uživatele ke zkopírování přihlašovacích údajů do lokální databáze.
3.3 Uživatel potvrdí přidání údajů do lokální databáze. Tok pokračuje k bodem č. 4.

Alternativní tok bodu 3.1

3.1.1 Pokud systém nelze připojit k centrálnímu SQL serveru, vyzve systém uživatele k zapnutí VPN spojení.
3.1.2 Uživatel vyplní přihlašovací údaje a opět odešle. Tok pokračuje bodem č. 3.

Alternativní tok bodu 3.2

3.2.1 Pokud systém nenalezne uživatelské jméno, vyzve uživatele k zadání jiných přihlašovacích údajů.
3.2.2 Uživatel vyplní přihlašovací údaje a opět odešle, tok pokračuje bodem č. 3.

Alternativní tok bodu 3.3

3.2.1 Pokud uživatel nepotvrdí zkopírování přihlašovacích údajů, pokračuje se bodem č. 2.

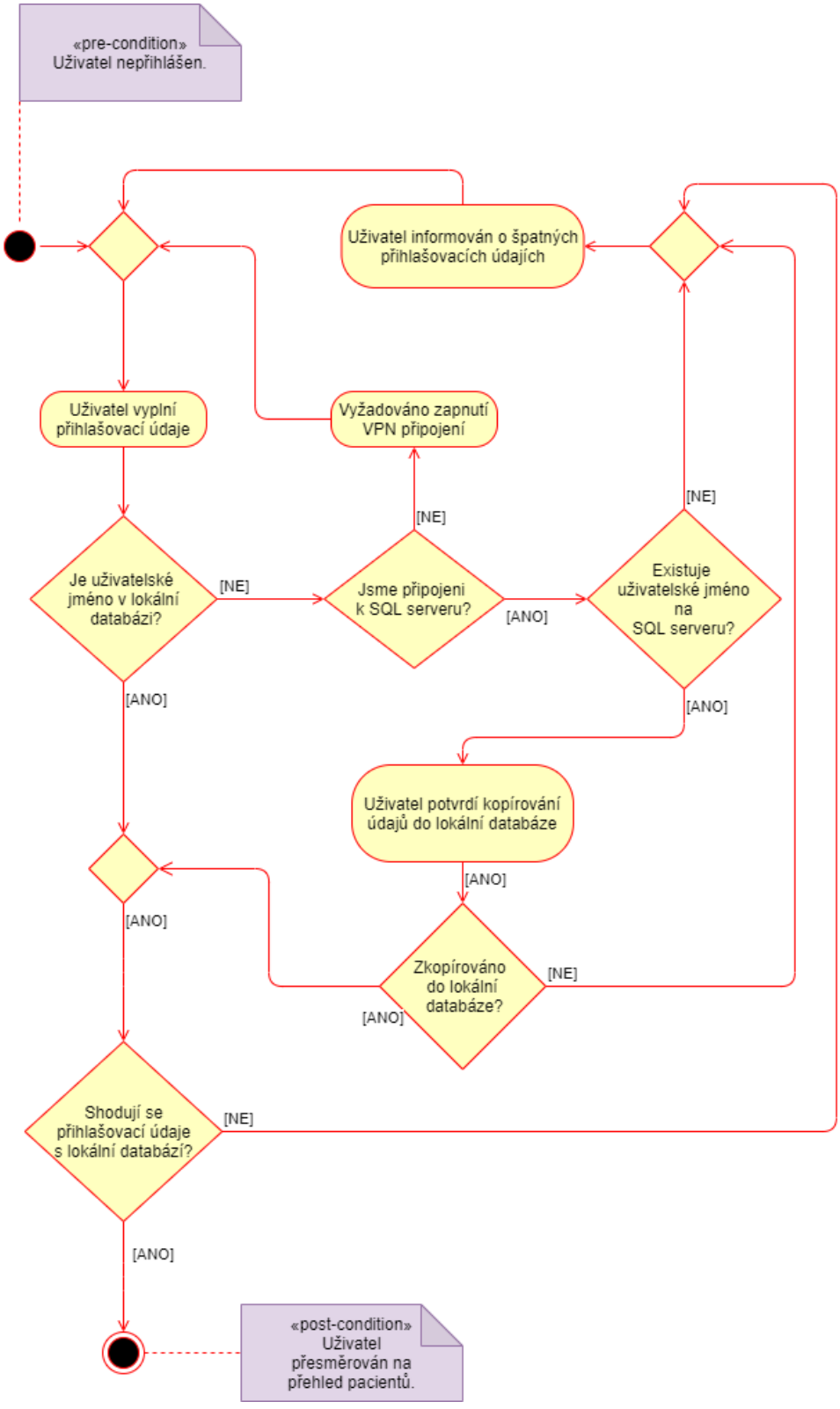
Alternativní tok bodu 4

4.1 Pokud nesouhlasí přihlašovací údaje s lokální databází, vyzve systém uživatele k zadání jiných přihlašovacích údajů.
4.2 Uživatel vyplní přihlašovací údaje a opět odešle tok pokračuje bodem č. 3.

Podmínky po spuštění

- Systém přihlásí uživatele do systému.

4.4.2.1.1 Aktivity diagram Přihlášení



Obrázek 4.2 Aktivity diagram Přihlášení (Zdroj: vlastní zpracování)

4.4.2.2 UC4 Přidat pacienta

Stručný popis
Use Case Přidat pacienta umožňuje přihlášenému uživateli přidat nového pacienta a uložit do lokální databáze.

Výpis aktérů
- Uživatel
- Systém

Podmínky před spuštěním
- Uživatel je přihlášen.

Základní tok
1. Uživatel spustí aktivitu přidání nového pacienta.
2. Uživatel vyplní příslušné kolonky a odešle formulář.
3. Systém zkontroluje vstupy a přidělí unikátní ID.
4. Systém uloží pacienta do lokální databáze a označí jako nesynchronizovaného.
5. Systém informuje pacienta o přidání do databáze.

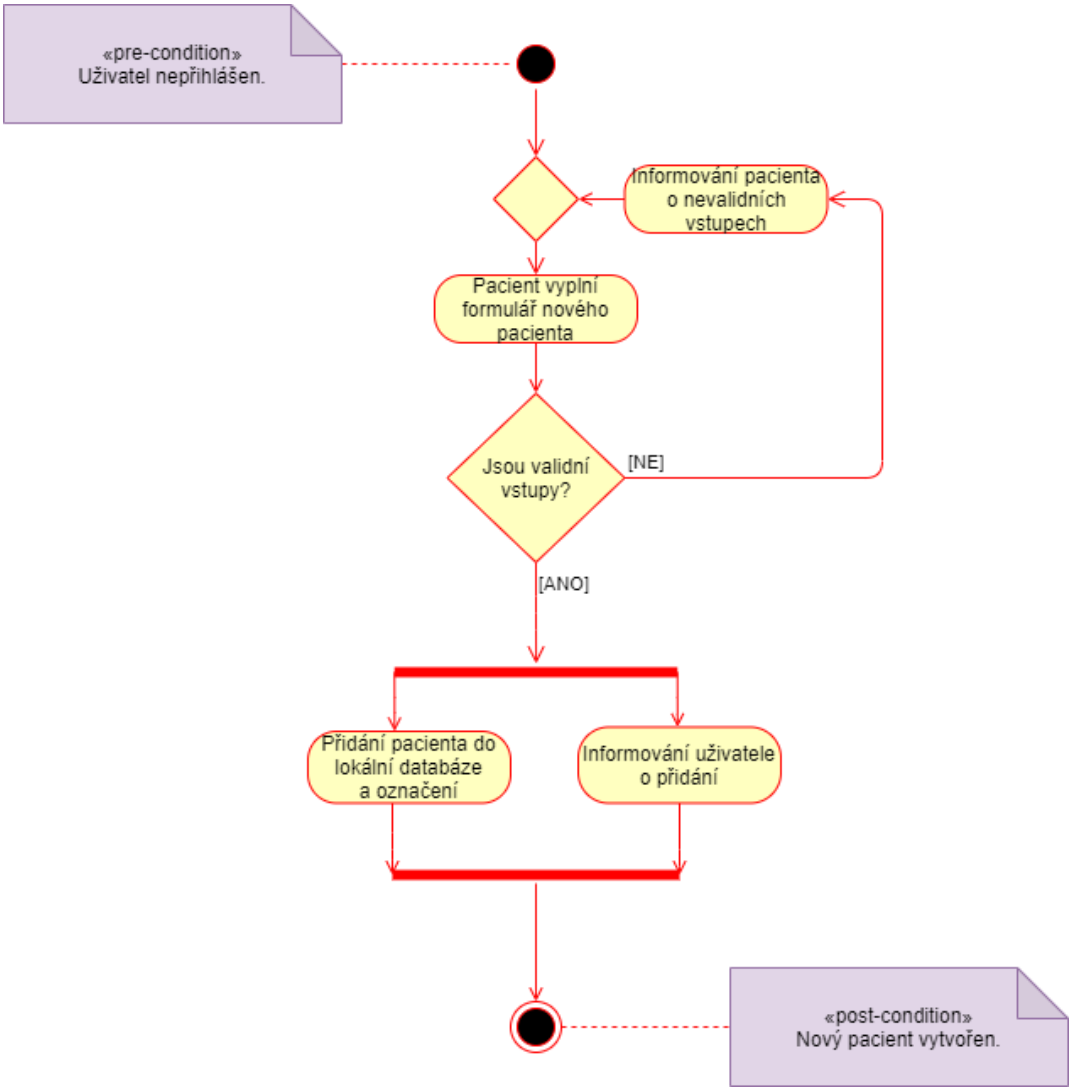
Alternativní tok bodu 3

3.1 Pokud systém nalezne nepovolené znaky, upozorní uživatele.

3.2 Tok pokračuje k bodem č. 2.

Podmínky po spuštění
- Systém přidá nového pacienta a označí ho jako nesynchronizovaného

4.4.2.2.1 Aktivity diagram Přidat pacienta



Obrázek 4.3 Přidat pacienta (Zdroj: vlastní zpracování)

4.4.2.3 UC4 Přidat měření

Stručný popis
Use Case Přidat měření umožňuje přihlášenému uživateli naměřit, nebo vybrat čtveřici souborů a tyto informace uložit do lokální databáze.

Výpis aktérů

- Uživatel
- Systém
- MetaWear
- Časovač

Podmínky před spuštěním

- Uživatel je přihlášen.
- Pacient již vytvořen.

Základní tok

1. Uživatel spustí aktivitu přidání nového měření.
2. Systém nabídne dva způsoby získání měření (souborová struktura, zařízení MetaWear).
3. Uživatel vybere připojení k zařízení MetaWear.
4. Systém nabídne různá zařízení MetaWear.
5. Uživatel vybere příslušné zařízení.
6. Uživatel vybere příslušný cíl, kam se bude měření ukládat.
7. Systém nastaví časovač.
8. Uživatel zahájí měření.
9. Časovač ukončí měření.
10. Systém přidělí měření správnému cíli.
11. Uživatel uloží měření.
12. Systém uloží měření do lokální databáze a označí měření jako nesynchronizováno.
13. Systém informuje uživatele o přidání do databáze.

Alternativní tok bodu 2

2.1 Pokud uživatel vybere měření ze souborového systému, systém nabídne souborový systém.
2.2 Uživatel vybere soubor ze systému.
2.3 Tok pokračuje od bodu č. 9.

Alternativní tok bodu 2.2

2.2.1 Pokud uživatel nevybere soubor ze souborového systému, tok pokračuje od bodu č. 2.

Alternativní tok bodu 5

5.1 Pokud uživatel nevybere zařízení, tok pokračuje od bodu č. 2.

Alternativní tok bodu 9

9.1 Pokud uživatel ukončí měření dříve, systém ukončí měření.
9.2 Tok pokračuje od bodu č. 10.

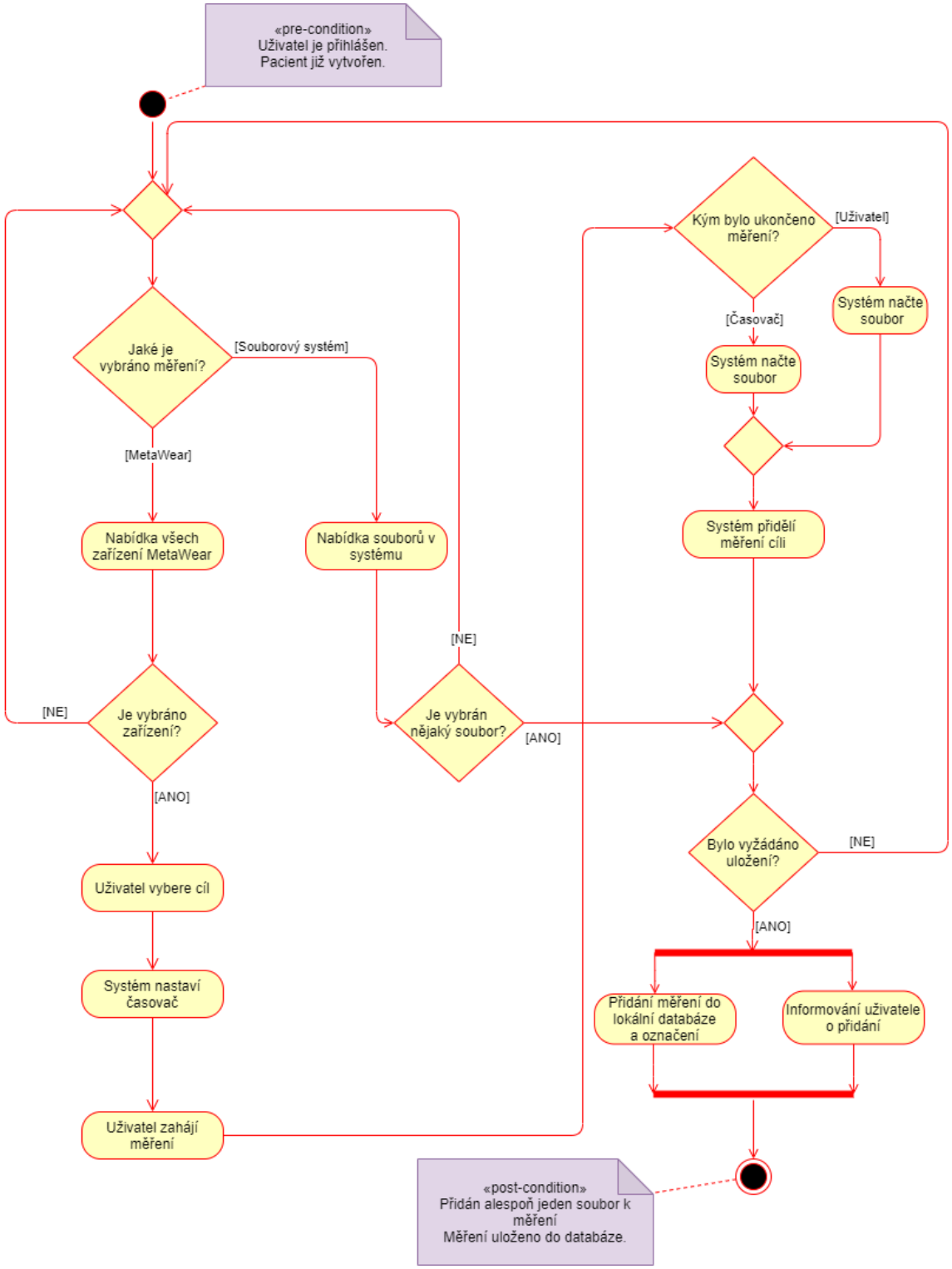
Alternativní tok bodu 11

11.1 Pokud uživatel neuloží měření, tok pokračuje od bodu č. 2.

Podmínky po spuštění

- Přidán alespoň jeden soubor k měření.
- Měření uloženo do databáze.

4.4.2.3.1 Aktivity diagram Přidat měření



Obrázek 4.4 Aktivity diagram Přidat měření (Zdroj: vlastní zpracování)

4.4.2.4 UC4 Odeslat měření

Stručný popis
Tento Use Case se stará o odeslání údajů pacienta, odeslání měření i naměřených souborů do daných destinací (QSL server a webový server).

Výpis aktérů
- Uživatel
- Systém
- Centrální SQL server
- Webový server

Podmínky před spuštěním
- Uživatel je přihlášen.
- Přidán alespoň jeden pacient.
- Přidáno alespoň jedno měření.

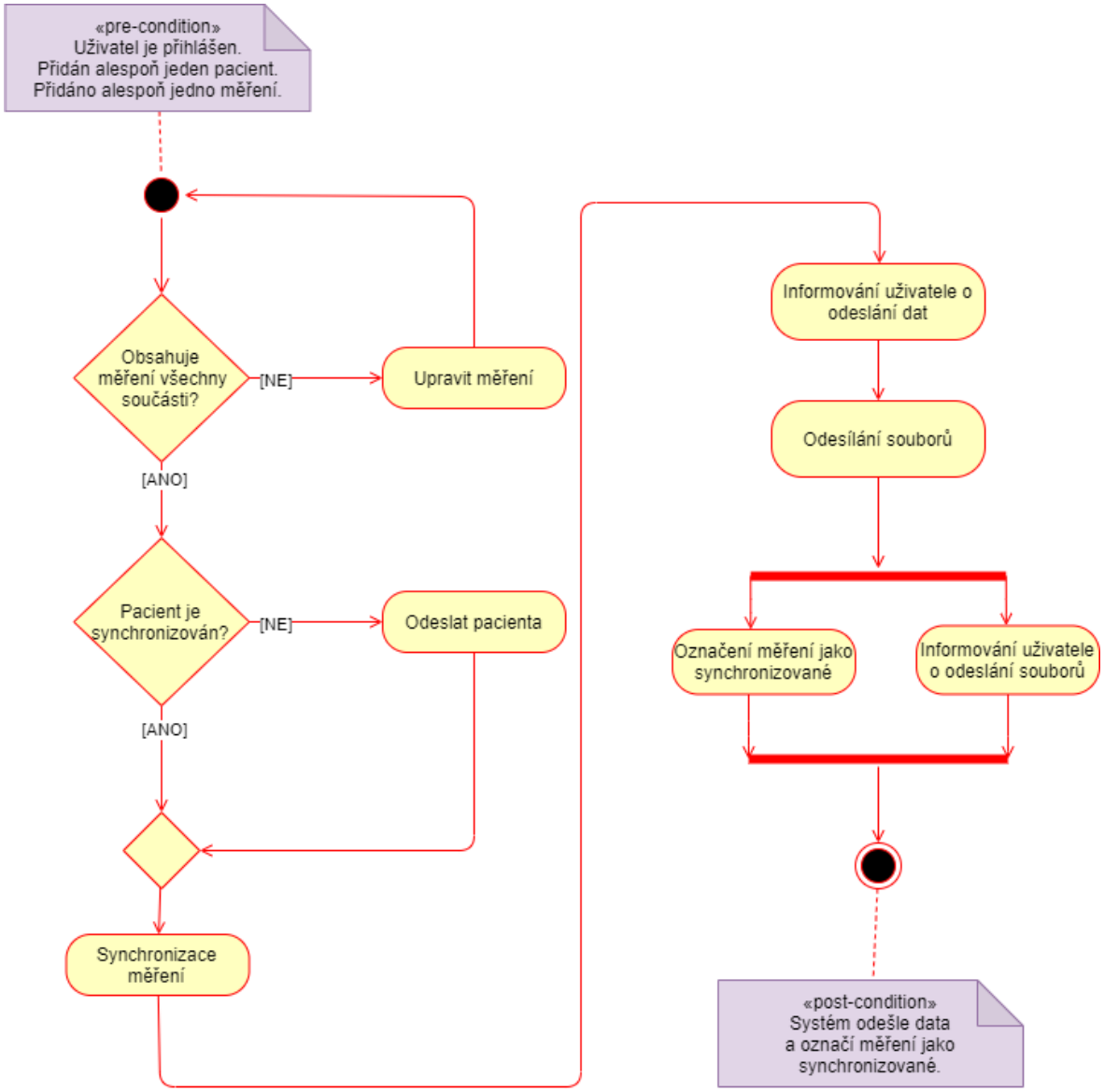
Základní tok
1. Uživatel vyvolá odeslání příslušného měření.
2. Systém zkontroluje, jestli měření obsahuje všechny součásti.
3. Systém zkontroluje, že je synchronizován pacient.
4. Systém pošle informace o měření do centrální databáze.
5. Systém informuje pacienta o přidání do databáze.
6. Systém odešle data webovému serveru.
7. Systém označí pacienta i měření jako synchronizované.

Alternativní tok bodu 2
2.1 Pokud systém nenalezne všechny součásti, upozorní uživatele.
2.2 Systém Spustí aktivitu Přidat měření.

Alternativní tok bodu 3
3.1 Pokud pacient není synchronizován, odešle pacienta.
3.2 Systém pokračuje bodem č. 5.

Podmínky po spuštění
- Systém odešle data a označí měření jako synchronizované.

4.4.2.4.1 Aktivity diagram Přidat měření



Obrázek 4.5 Aktivity diagram Odeslat měření (Zdroj: vlastní zpracování)

4.4.3 Use Case model webového rozhraní

Následně byl vymodelován Use Case diagram webového rozhraní. Ten má sloužit vedoucímu projektu, aby měl přímý přístup k datům bez nutnosti připojení k SQL databázi nebo přes FTP přístup. Use Case zobrazuje akce, které bude moct uživatel provádět, viz obrázek 4.6. Use Case diagram obsahuje 4 aktéry:

Nepřihlášený uživatel
Tento aktér může zadat uživatelské jméno a heslo do systému.

Přihlášený uživatel
Aktér „Přihlášený uživatel“ může zobrazit svá měření a své pacienty. Dále má přístup k jejich souborům.

Superuser
Aktér „Superuser“ má oproti aktéru „Přihlášený uživatel“ přístup nejen ke svým pacientům, ale zároveň ke zbylé databázi pacientů a jejich měření. Může stahovat jejich soubory, přidávat a mazat uživatele s nižšími právy a mazat pacienty.

Centrální SQL server
Jedná se o pasivního aktéra, který poskytuje data uložená a zapisuje přijatá data na vzdáleném serveru přes VPN spojení.

Časovač
Tento pasivní aktér se stará o časové přerušení toku.

Obrázek 4.6 Use Case Webové rozhraní (Zdroj: vlastní zpracování)

4.4.4 Use case scénáře webového rozhraní

V této kapitole bude popsány vybrané Use case scénáře a budou popsány některé Aktivity diagramy. Přístup přes webové rozhraní je předně navržen pro pana doktora Zemana. Ten má přístup do sítě, kde se server nachází, trvale z jeho pracoviště. Nebylo tedy potřeba přístup k serveru pomocí VPN do Aktivity diagramů zahrnovat a nebude ve specifikacích zmiňován.

4.4.4.1 UC1 Přihlásit

Stručný popis
Tento Use Case umožňuje nepřihlášenému uživateli se přihlásit. Přihlášením získá přístup k systému dle přidělených práv v SQL databázi.

Výpis aktérů
- Uživatel
- Systém
- Centrální SQL server
- Časovač

Podmínky před spuštěním
- Uživatel není přihlášen.

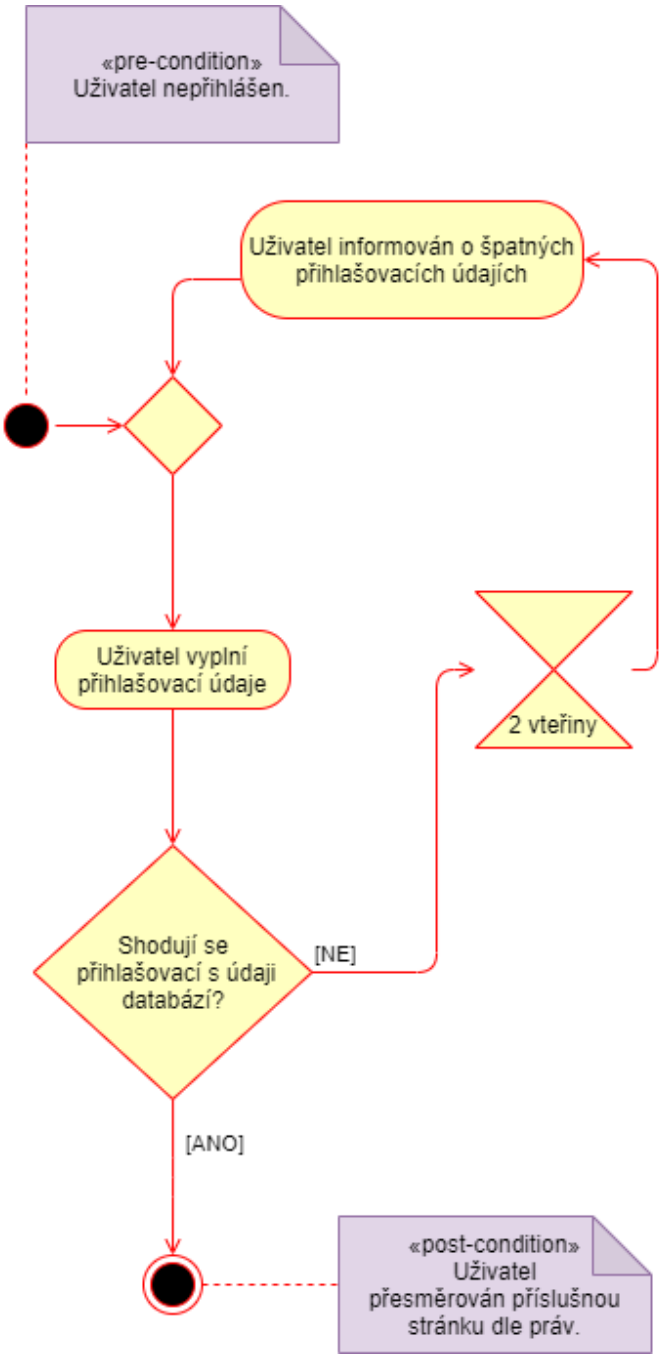
Základní tok
1. Systém vyzve uživatele k přihlášení do systému.
2. Uživatel vyplní přihlašovací údaje a odešle přihlašovací formulář.
3. Systém porovná uživatelské jméno a heslo s databází SQL serveru.
4. Systém přihlásí uživatele do webového rozhraní s příslušnými právy.

Alternativní tok bodu 3

3.1 Pokud systém nenalezne příslušné údaje, spustí časovač.
3.2 Po uplynutí doby časovače systém informuje uživatele o chybně zadaných přihlašovacích údajích.
3.3 Tok pokračuje bodem č. 2.

Podmínky po spuštění
- Uživatel přesměrován na příslušnou stránku dle práv.

4.4.4.1.1 Aktivity diagram Přihlásit

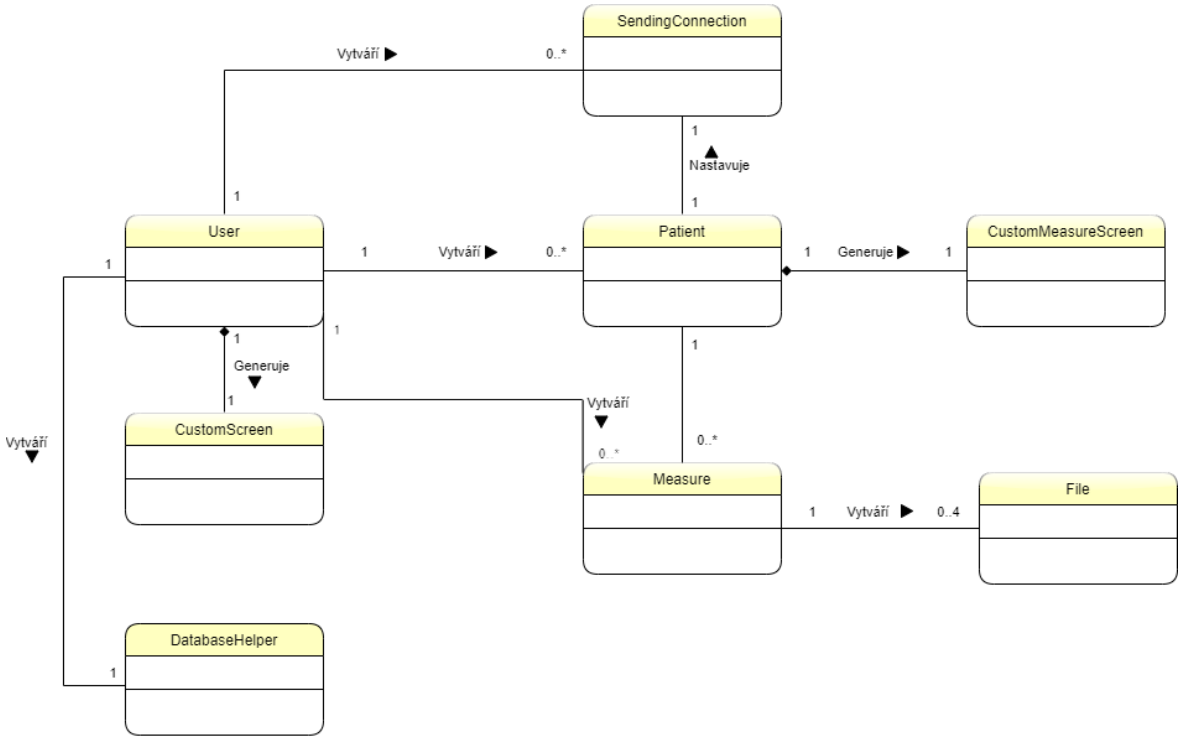


Obrázek 4.7Aktivity diagram Přihlásit - webové rozhraní (Zdroj: vlastní zpracování)

4.4.5 **Struktura aplikace**

4.4.5.1 Doménový diagram

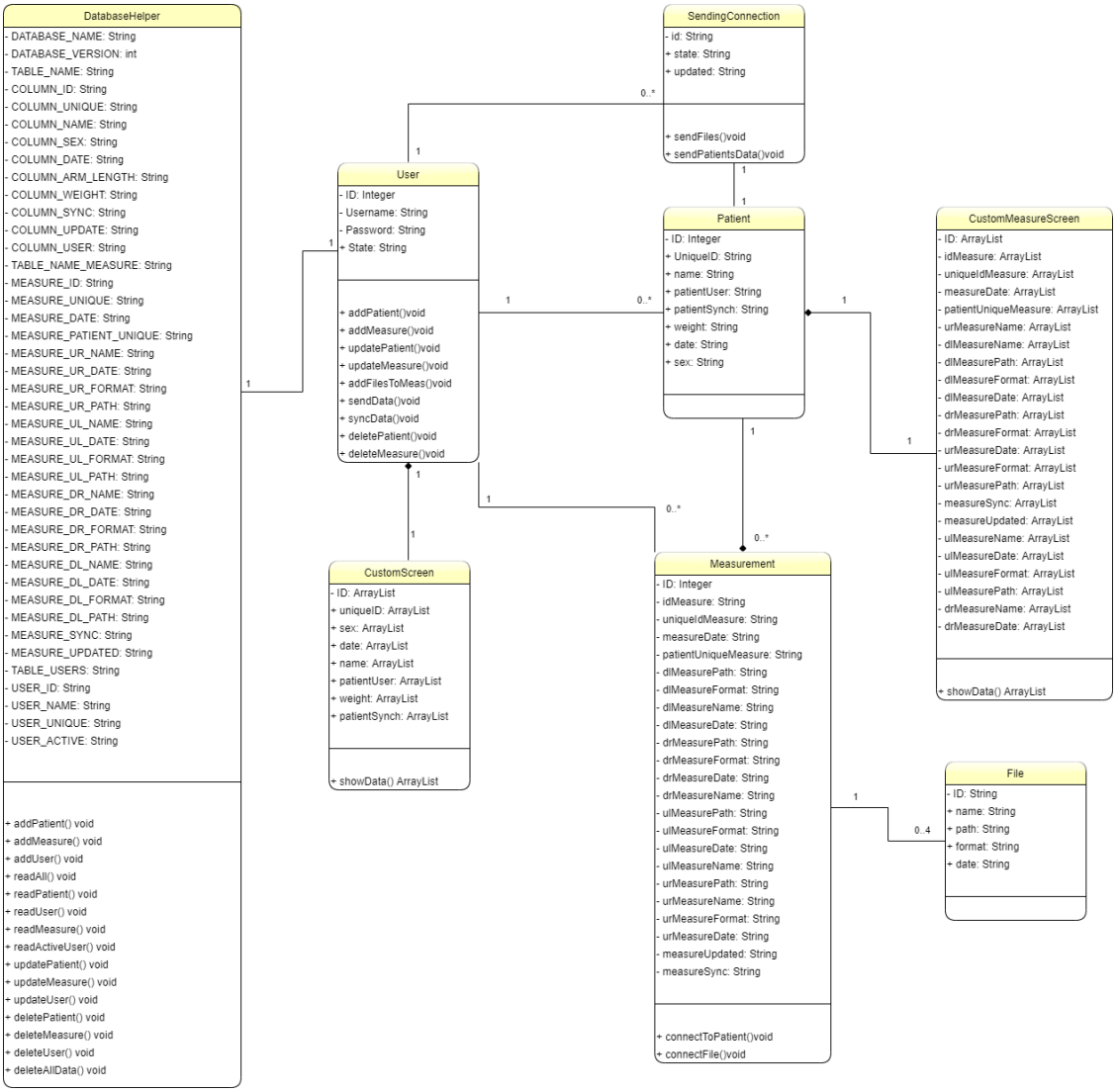
Před samotnou implementací byl vytvořen doménový diagram viz obrázek 4.8. Vzhledem k poněkud specifické struktuře Android aplikací byl vytvořen tento diagram především pro hrubou představu, jak bude aplikace koncipována.



Obrázek 4.8 Doménový diagram (Zdroj: vlastní zpracování)

4.4.5.2 Návrhový diagram tříd

Třídní diagram byl vytvořen pro hrubou představu o atributech a metodách viz obrázek 4.9



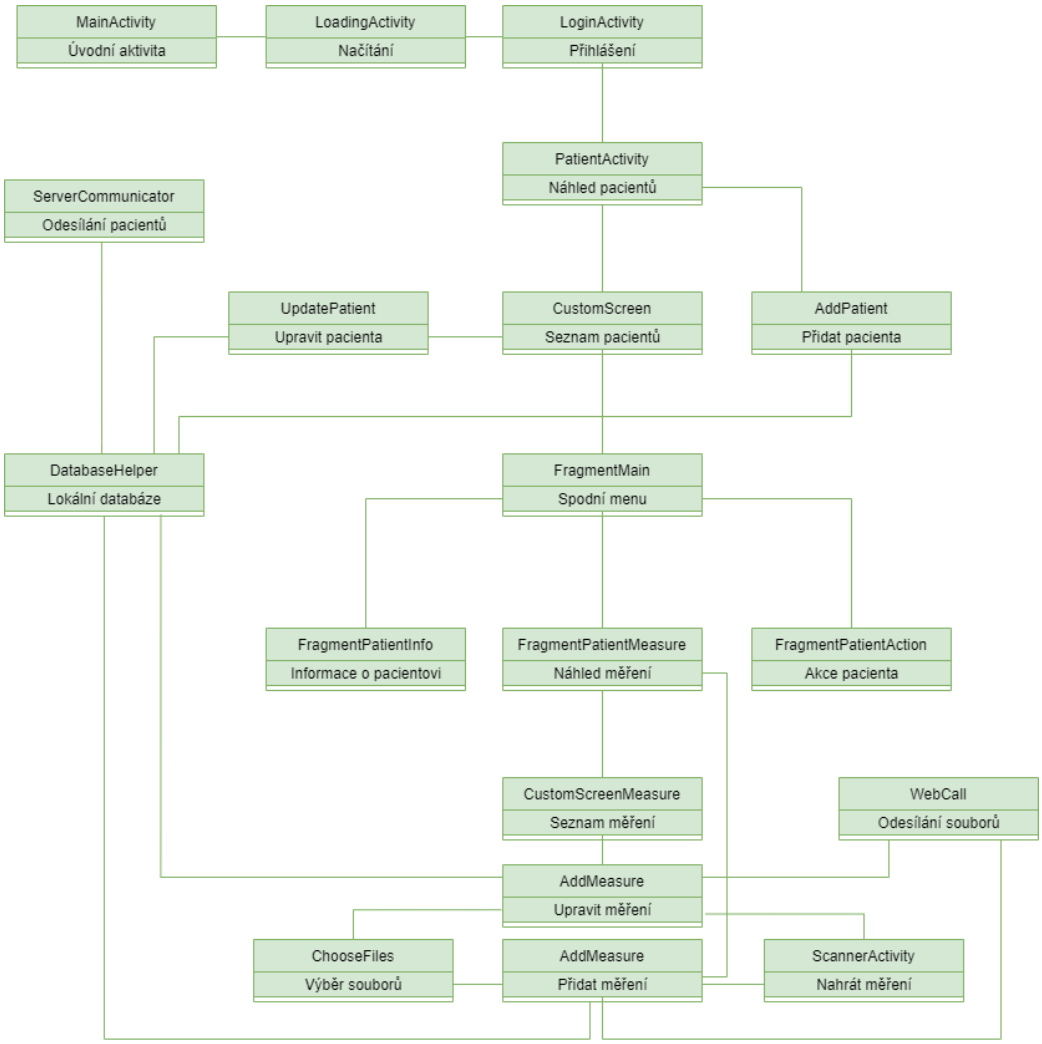
Obrázek 4.9 Třídní diagram (Zdroj: vlastní zpracování)

Třídní diagram (Zdroj: vlastní zpracování).

4.4.5.3 Přehled tříd, aktivit a fragmentů

Programování pro Android má svá specifika. Aplikace pro systém Android mají kromě klasických tříd i aktivity a fragmenty, čímž je struktura programu mnohem složitější než obecný návrh. Po hrubém návrhu designu byla zpracována předpokládaná struktura

fragmentů a aktivit do speciálního diagramu. Aktivita/Fragment obsahují název a popis funkce, čáry znázorňují vztahy mezi nimi, viz obrázek 4.10.



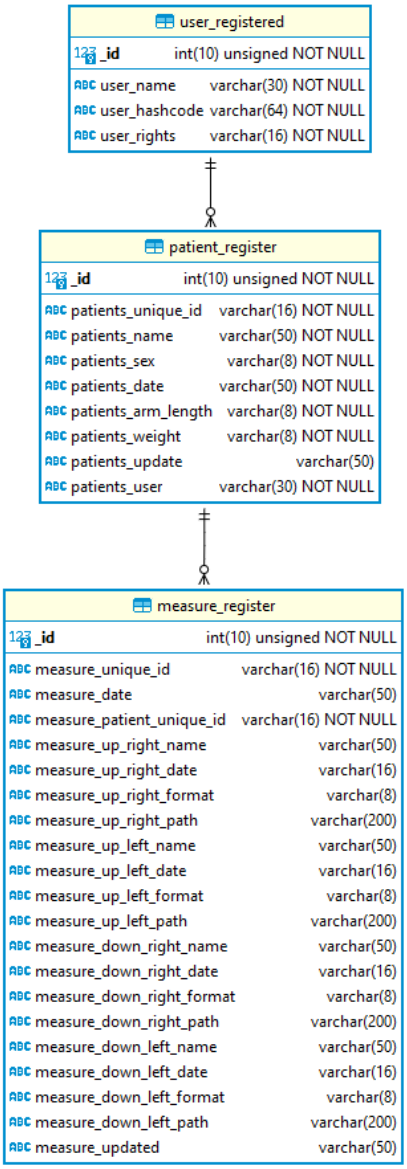
Obrázek 4.10 Diagram Aktivit a Fragmentů (Zdroj: vlastní zpracování)

4.4.6 **Datová struktura**

4.4.6.1 E-R diagram centrálního serveru

Tento diagram zobrazuje datovou strukturu centrálního SQL serveru, viz obrázek 4.11. Součástí návrhu jsou tři tabulky. Jedna pro správu uživatelů, druhá pro registraci pacientů a třetí pro registraci měření.

User_registered
Tato tabulka bude obsahovat uživatelská jména a hesla uživatelů. Uživatelské jméno je reprezentováno datovým typem String o maximální hodnotě třiceti znaků. Položka user_hashcode má nastaveno varchar(64), protože se v aplikaci využívá kryptovací funkce SHA256, která nabývá 64 znaků. Poslední sloupec obsahuje informaci o uživatelských právech.



Obrázek 4.11 E-R diagram centrálního SQL serveru (Zdroj: vlastní zpracování)

Patient_register

Tato tabulka se bude starat o registraci pacientů. Bude obsahovat unikátní číslo pacienta, jméno, a následně všechny potřebné informace jako jsou pohlaví, délka paže, datum narození a váha. Následují sloupce datum synchronizování pacienta a informace o uživateli, který pacienta na server nahrál.

Measure_registered

Tato tabulka bude obsahovat seznamy všech měření a její součástí budou unikátní ID měření a datum měření a následně pacientovo unikátní ID. Dále jsou všechny náležitosti, které měření dle zadání musí mít, jako jsou jméno, datum, formát a cesta k naměřenému souboru vždy pro pravou ruku s otevřenýma očima, levou ruku s otevřenýma očima, pro pravou ruku se zavřenýma očima a pro levou ruku se zavřenýma očima. Poslední sloupec je datum, kdy bylo měření synchronizováno.

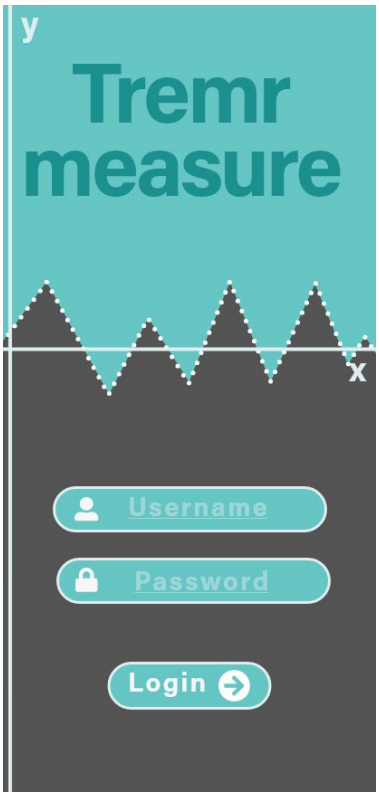
4.5 Návrh prezentační struktury a designu (Wireframes)

4.5.1 Návrh mobilní aplikace

Návrh designu mobilní aplikace vycházel z diagramu aktivit a fragmentů. Nejprve byly zpracovány ručně kreslené skici, které byly konzultovány a následně schváleny. Vzhledem ke konkrétnímu zadání, těžko představitelné struktuře aplikace a složitým úpravám v případě změny designu, bylo přistoupeno ke zpracování aktivního wireframe. Návrhy byly průběžně konzultovány. PDF soubor všech obrazovek z návrhu je přiložen v příloze C.
Aktivní wireframe byl navržen tak, aby veškeré části odpovídaly reálnému návrhu, aby každá obrazovka odpovídala příslušné aktivitě nebo fragmentu. Tím se návrh designu stal nejen důležitou komponentou ohledně vzhledu, ale také ověřením, že je strukturní diagram aktivit a fragmentů správně navržen. Zároveň došlo i k propojení atributů a metod z diagramu tříd s celkovým návrhem aplikace. Všechny níže představené návrhy jsou pouze klikatelný grafický soubor obrázků a nelze z něj generovat žádný negrafický obsah (kód, třídy, aktivity atp.).

4.5.1.1 Přihlášení do aplikace

Uživatel zde potřebuje pro přístup do aplikace zadat uživatelské jméno a heslo a potvrdit odeslání formuláře. Jsou tedy vyžadovány 3 akce: zadání jména, zadání hesla a tlačítko pro odeslání formuláře. Na obrázku 4.12 vidíme klikatelný návrh přihlašovací obrazovky.



Obrázek 4.12 Návrh přihlášení do aplikace návrh (Zdroj: vlastní zpracování)

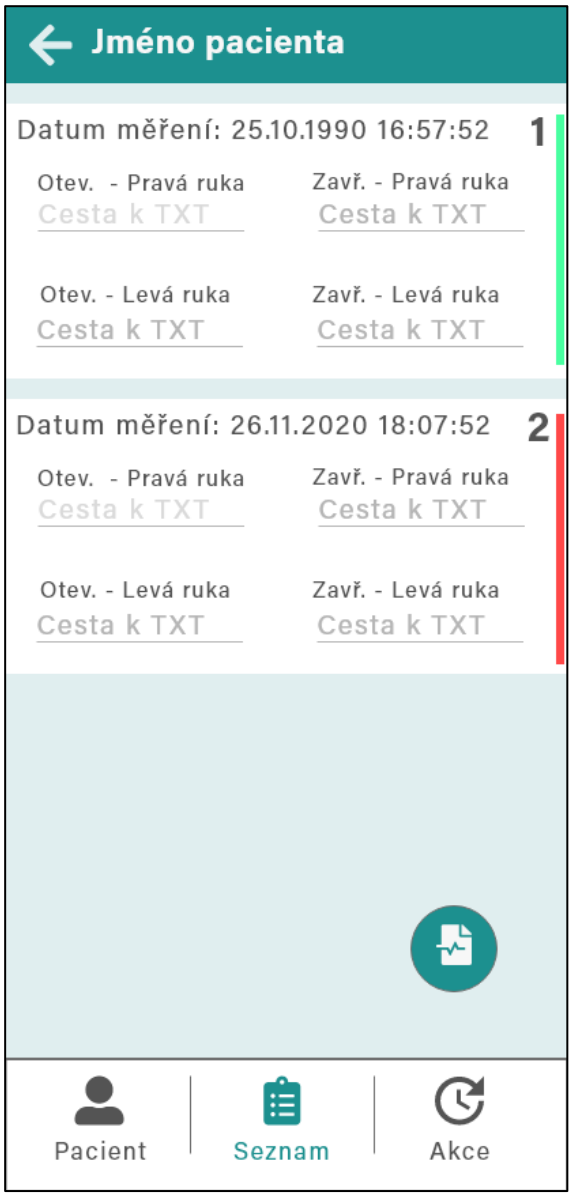
4.5.1.2 Přehled pacientů

V této aktivitě uživatel potřebuje kompletní přehled o všech svých pacientech a zároveň mít možnost pacienty přidat nebo upravit. Nachází se zde tedy tlačítko přidání pacienta a klikatelný seznam všech pacientů, viz obrázek 4.13. Dále se zde nachází informace o synchronizaci pacienta v podobě červeného/zeleného pruhu podél pravé strany položky.

Obrázek 4.13 Návrh vzhledu Přehled pacientů návrh (Zdroj: vlastní zpracování)

4.5.1.3 Přehled o měřeních

V této aktivitě uživatel potřebuje kompletní přehled o všech svých měřeních. Struktura je podobná jako v případě pacientů, avšak nachází se zde spodní menu, kde může uživatel procházet detaily pacienta. Nachází se zde tedy tlačítko přidání měření, spodní menu s tlačítky informace, měření a akce a klikatelný seznam všech měření, viz obrázek 4.14. Zároveň se zde vyskytuje informace o synchronizaci měření v podobě červeného/zeleného pruhu v pravé části položky.



Obrázek 4.14 Návrh vzhledu Přehled měřeních (Zdroj: vlastní zpracování)

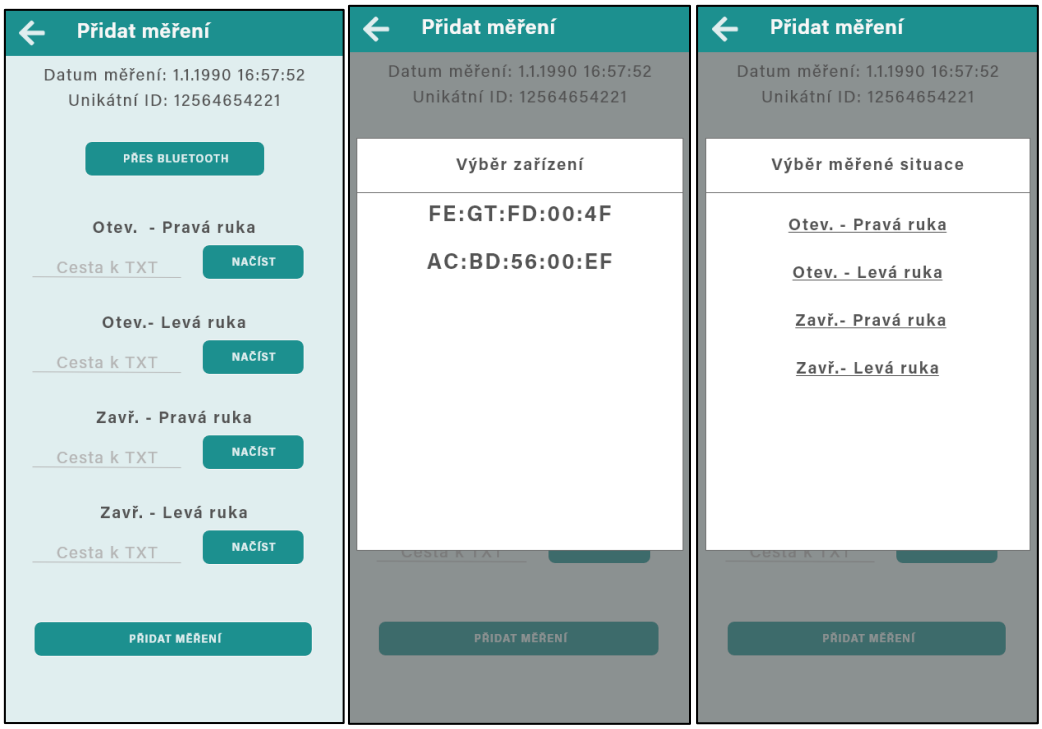
4.5.1.4 Přidat pacienta

V této aktivitě uživatel potřebuje vyplnit veškerou diagnózu pacienta, která vychází z atributů, které jsou u pacienta naplánovány. Dále potřebuje tlačítko potvrdit, kterým odešle formulář. Zároveň je potřeba dle zadání zadávat všechny potřebné parametry pomocí posuvníků, aby bylo vyplňování formuláře pro uživatele co nejvíce pohodlné. Přesto musí formulář obsahovat možnost parametry upravovat manuálně. Také musí být v aktivitě možnost náhodně generovat přezdívku pacienta. Zpracování této obrazovky si lze prohlédnout na obrázku 3.15.

Obrázek 4.15 Návrh vzhledu Přidání pacienta (Zdroj: vlastní zpracování)

4.5.1.5 Přidat měření

Zde potřebuje uživatel načíst čtyři soubory, buď ze souborového systému nebo z bluetooth zařízení MetaWear. Na obrázku 4.16 si je představeno kromě samotné aktivity (vlevo) situace po kliknutí na tlačítko Bluetooth (uprostřed), které znázorňuje výběr zařízení MetaWear a vpravo specifikaci cíle měření. Nejedná se o samotné aktivity, ale pouze o vyskakující oznámení.



Obrázek 4.16 Návrh vzhledu Přidání měření(vlevo), výběr zařízení (uprostřed) a výběru cíle (vpravo)

(Zdroj: vlastní zpracování)

4.5.1.6 Měření třesu

Zde musí mít uživatel možnost spustit automaticky časované měření, ale zároveň musí mít uživatel možnost měření přerušit. K tomuto účelu slouží tlačítko stop. Dále jsou navrženy v oznámení další dvě tlačítka. Pro uložení daného cíle a pro uložení celého měření, viz obrázek 4.17.

Obrázek 4.17 Návrh měření třesu (Zdroj: vlastní zpracování)

4.5.1.7 Přehled a akce pacienta

Na závěr byl ještě zpracován návrh přehledu pacienta a jeho akcí. Přehled pacienta uživateli zobrazuje souhrnná data o pacientovi. Také má zde uživatel možnost zkontrolovat, jestli jsou pacienti a jeho měření synchronizována. Na aktivitě akce pacienta jsou zobrazeny souhrnně všechny akce, které může uživatel s pacienty provádět, viz obrázek 4.18.

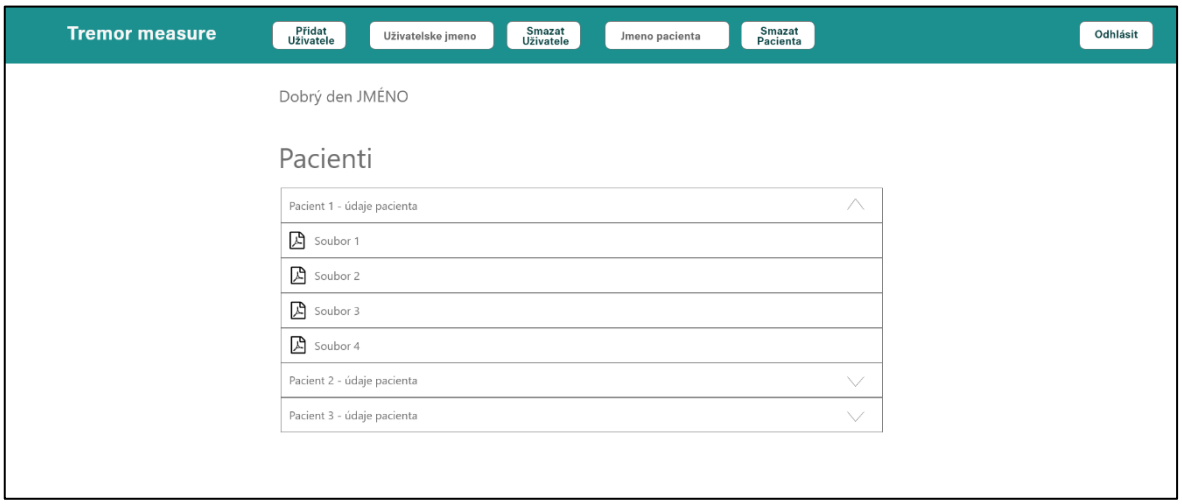
Obrázek 4.18 Přehled pacienta(vlevo) a akcí (vpravo) (Zdroj: vlastní zpracování)

4.5.2 **Návrh webové aplikace**

Původní návrh webové aplikace byl koncipován tak, že bude rozhraní používáno pouze správcem databáze. Při návrhu však došlo ke změně zadání a systém je nakonec koncipován tak, že se i uživatelé mohou připojit do webového rozhraní s uživatelskými právy „User“.
Z tohoto důvodu se nakonec přistoupilo designové úpravě i u webového rozhraní. Statický wireframe byl opět zpracován a konzultován. Byl navržen tak, aby se podobal vzhledu aplikace a odpovídal celkovému konceptu.
Přihlašovací stránka je převzata z grafického návrhu aplikace, a proto zde nebude uvedena. Na obrázku 4.19 je představen grafický návrh mobilní verze pro uživatele s právy „User“. Tito uživatelé mohou pouze nahlížet na své pacienty a stahovat jejich měření.

Obrázek 4.19 Návrh webového rozhraní pro mobilní zařízení - User (Zdroj: vlastní zpracování)

Na obrázku 4.20 je představen návrh rozhraní PC verze pro uživatele s právy „Superuser“. Tento uživatel má oproti uživateli s právy „User“ navíc některé funkcionality. V horním panelu se nacházejí nástroje pro smazání pacienta, smazání uživatele a tlačítko pro přidání uživatele.



Obrázek 4.20 Návrh webového rozhraní pro PC - Superser (Zdroj: vlastní zpracování)

4.6 Implementace mobilní aplikace

V této kapitole je popsána funkce některých tříd, aktivit a fragmentů, které se v aplikaci vyskytují. Vzhledem k rozsáhlému zdrojovému kódu (cca 4 000 řádků) zde budou zobrazeny pouze některé části. Kompletní zdrojový kód je k nahlédnutí v příloze A.

4.6.1 Aktivity a Fragmenty

4.6.1.1 Main Activity a LoadingActivity

Tyto aktivity spouštějí startovací procesy aplikace a zobrazují úvodní animaci. Ta je zajištěna zdrojem "Animace" a má podobu xml souboru, který animuje alpha kanál dané obrazovky. V následující ukázce je ukázán efekt „fadeout“, viz zdrojový kód 1.

```
<alpha
    android:duration="1000"
    android:fromAlpha="1.0"
    android:interpolator="@android:anim/accelerate_interpolator"
    android:toAlpha="0.0" />
```

Zdrojový kód 1 fadeout.xml

4.6.1.2 LoginActivity

Tato aktivita se stará o přihlášení uživatele do aplikace. Jak již vyplývá z příslušného aktivity diagramu v kapitole 4.4.2.1.1, jedná se o celkem složitý proces. Zde si pro ukázku představíme předvyplňování uživatelského jména a kontroly jména na centrálním SQL serveru.

Předvyplňování Username
Ze všeho nejdříve se aktivita připojí do lokální databáze a zkontroluje, zdali existuje nějaký uživatel. Pokud uživatele nalezne, vyplní jeho jméno do políčka username.

```
DatabaseHelper myDB = new DatabaseHelper(LoginActivity.this);

Cursor cursor = myDB.readUserActiveState("1");
if (cursor.getCount() == 0) {
} else {
    while (cursor.moveToNext()) {
        userNameActiveBefore = (cursor.getString(1));
        myDB.updateUserNameState(userNameActiveBefore, "0");
        user_input.setText(userNameActiveBefore);
    }
    userNameActiveBefore = null;
}
```

Zdrojový kód 2 LoginActivity - předvyplňování username

Stahování username ze serveru
Poté, co uživatel zadá přihlašovací údaje, zkontroluje se, zda uživatel není uložen v lokální databázi. Při nenalezení záznamu se aplikace připojí do centrálního SQL serveru, pokusí se uživatele nalézt a stáhnout do lokální databáze.

```
if (serverCommunicator.checkICMPConnection()) {   //Kontrola zapnuté VPM
    ResultSet rs = serverCommunicator.checkServerUsers(userName);
    if (rs == null) {
        Toast.makeText(LoginActivity.this,
getString(R.string.noexistpassuser), Toast.LENGTH_SHORT).show();
    } else {
        String userNameServer = null;
        String passHashServer = null;
        try {
            userNameServer = rs.getString(2);
            passHashServer = rs.getString(3);
        } catch (SQLException throwables) {
            throwables.printStackTrace();
        }
        newUser = new
androidx.appcompat.app.AlertDialog.Builder(LoginActivity.this,
R.style.AlertDialogTheme);
        newUser.setTitle(R.string.dialog_title2);
        newUser.setMessage(R.string.dialog_message2);
        String finalUserNameServer = userNameServer;
        String finalPassHashServer = passHashServer;
        newUser.setPositiveButton(getString(R.string.yes), new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                myDB.addUser(finalUserNameServer, finalPassHashServer, "0");
            }
        })
//… kód pokračuje viz příloha A
```

Zdrojový kód 3 LoginActivity – Stahování username ze serveru

4.6.1.3 PatientActivity

Aktivita PatientActivity se stará o korektní zobrazení všech pacientů. Při jejím vytváření se nastaví výchozí jazyk aplikace a v horním pravém rohu se nastaví menu, ve kterém lze vybrat několik možností, jako jsou změna jazyka, vyhledání pacienta v centrální SQL databázi, smazat celou lokální databázi mobilního zařízení a zobrazit nápovědu. V neposlední řadě se tato aktivita také stará o přidání nového pacienta tlačítkem v pravém dolním rohu.

Zobrazení pacientů
Zobrazení všech pacientů se provádí tak, že se nejprve načtou všechny příslušné položky z databáze, které se následně odesílají do aktivity CustomScreen, která je rozdělí do příslušných karet dle pacientů, viz zdrojový kód 4.

```
myDB = new DatabaseHelper(PatientActivity.this);
id = new ArrayList<>();
uniqueId = new ArrayList<>();
name = new ArrayList<>();
sex = new ArrayList<>();
date = new ArrayList<>();
arm = new ArrayList<>();
weight = new ArrayList<>();
patientsSync = new ArrayList<>();
patientsUser = new ArrayList<>();

storeDataInArrays();

customScreen = new CustomScreen(PatientActivity.this, this, id, uniqueId,
name, sex, date, arm, weight, patientsSync, patientsUser);

recyclerView.setAdapter(customScreen);
recyclerView.setLayoutManager(new LinearLayoutManager(PatientActivity.this));
//… kód pokračuje viz příloha A
```

Zdrojový kód 4 PatientActivity – generování náhodného jména

4.6.1.4 AddPatient

Tato aktivita se stará o formulář přidání nového pacienta. Zde si uživatel může zvolit parametry přidávaného pacienta pomocí tlačítek a posuvníků. Přidávání náhodného jména pacientů, jak bylo požadováno v zadání, probíhá pomocí dvou polí datového typu string, které se mezi sebou náhodně kombinují. Neboť registr pacientů nezávisí na jménu, není nutné hlídat opakující se názvy v rámci centrální databáze a ani v rámci aplikace. Generování náhodných jmen slouží pouze k případnému usnadnění vyplňování (uživatel při vyplňování náhodně generuje jméno a až po odchodu pacienta upraví políčko příslušnou přezdívkou).

Generování náhodného jména

Generování náhodného jména probíhá v metodě changeName pomocí funkce Math.*random*(), která generuje náhodné číslo v rozmezí počtu položek v poli a tím vybírá náhodnou kombinaci slov, viz zdrojový kód 5.

```
String changeName(String sex) {
    if (sex.equals("Male")){
        String[] animal = {getString(R.string.lion ), getString(R.string.tiger ),
getString(R.string.eagle ), getString(R.string.deer ), getString(R.string.bear ),
getString(R.string.elephant )};
        String[] color = {getString(R.string.blackM ), getString(R.string.blueM ),
getString(R.string.greyM ), getString(R.string.brownM ), getString(R.string.greenM )};
        return color[(int) (Math.random() * color.length)] + " " + animal[(int)
(Math.random() * animal.length)];

    }else if (sex.equals("Female")){
        String[] animal = {getString(R.string.lioness ), getString(R.string.giraffe ),
getString(R.string.ladybug ), getString(R.string.otter ), getString(R.string.squirrel
), getString(R.string.fox )};
        String[] color = {getString(R.string.pink ), getString(R.string.green ),
getString(R.string.blue ), getString(R.string.black ), getString(R.string.grey )};
        return color[(int) (Math.random() * color.length)] + " " + animal[(int)
(Math.random() * animal.length)];
//… kód pokračuje viz příloha A
```

Zdrojový kód 5 AddPatient – Generování náhodného jména

4.6.1.5 FragmentMain

FragmentMain je základním fragmentem aplikace a stará se o nastavení spodního menu pacienta a o distribuci argumentů rozbaleného pacienta.

Spodní menu pacienta
Menu má tři položky. Každá položka ve spodním menu načítá svůj vlastní fragment, který je u položek uveden v závorkách.Položky jsou: informace o pacientu (FragmentPatientInfo), seznam všech měření pacienta (FragmentPatientMeasure) a akce pacienta (FragmentPatientAction), které může uživatel s pacientem provádět. K tomuto řešení bylo využito poměrně nové funkcionality NavigationUI. Pokud uživatel vybere položku v nabídce, NavController zavolá funkci onNavDestinationSelected(), která automaticky aktualizuje vybranou položku ve spodním navigačním panelu. V tomto řešení bylo nutné přistoupit ke složitější konstrukci, protože bylo potřeba spolu s přechodem na jiný fragment poslat i informace o pacientu z výchozí aktivity, viz zdrojový kód 6.

```
setContentView(R.layout.activity_bottom_menu);
BottomNavigationView bottomNavigationView  = findViewById(R.id.bottomNavigationView);
NavHostFragment  navhostFragment = (NavHostFragment)
getSupportFragmentManager().findFragmentById(R.id.fragment);
NavController navController = navhostFragment.getNavController();
// Vkládání příslušných argumentů
args.putString("id4", id);
args.putString("uniqueId4", uniqueIdV);
args.putString("name4", name);
args.putString("sex4", sex);
args.putString("date4", date);
args.putString("arm4", arm);
args.putString("weight4", weight);
args.putString("patientsSync4", patientsSync);
args.putString("patientsUser4", patientsUser);
//…
//…
//…
//…
NavigationUI.setupWithNavController(bottomNavigationView, navController);
bottomNavigationView.setOnNavigationItemSelectedListener(new
BottomNavigationView.OnNavigationItemSelectedListener() {
    @Override
    public boolean onNavigationItemSelected(@NonNull MenuItem item) {
        navController.navigate(item.getItemId(), args);
        return true;
    }
});
navController.setGraph(R.navigation.pat_nav, args);
//… kód pokračuje viz příloha A
```

Zdrojový kód 6 FragmentMain - NavigationUI

4.6.1.6 FragmentPatientInfo

Tento Fragment slouží pouze jako přehled rozkliknutého pacienta a je nastaven jako výchozí fragment. Uživatel zde může vidět všechny parametry pacienta na jedné obrazovce a také se pomocí barevných šipek může přesvědčit, zda je pacient synchronizován se serverem. Ve všech fragmentech (rozkliknutý konkrétní pacient) není možno k návratu do přehledu pacientů využít hardwarové tlačítko zpět ve spodní části mobilního zařízení. Dle provedených testů se při nechtěném doteků stávalo, že uživatel odešel z rozhraní pacienta omylem a vzhledem ke složité struktuře aplikace ztrácel přehled, ve které části se nachází. Taktéž pro návrat například z fragmentu FragmentPatientAction bylo nutno se proklikávat přes přehled pacientů, úvodní fragment a až následně k fragmentu. Proto byla zvolena varianta, kdy v přehledu konkrétního pacienta je potřeba pro přechod na přehled pacientů využít zpětnou šipku v horní liště. Při kliknutí na tlačítko „Zpět“ systém uživatele na tuto skutečnost upozorní, viz zdrojový kód 7 .

```
@Override
public void onResume() {
    super.onResume();
    getView().setFocusableInTouchMode(true);
    getView().requestFocus();
    getView().setOnKeyListener(new View.OnKeyListener() {
        @Override
        public boolean onKey(View v, int keyCode, KeyEvent event) {
            if (event.getAction() == KeyEvent.ACTION_UP && keyCode ==
KeyEvent.KEYCODE_BACK){
                if(getFragmentManager().getBackStackEntryCount() > 0) {
                    Toast.makeText(getContext(), R.string.backnav,
Toast.LENGTH_SHORT).show();
                }
                return true;
            }
            return false;
        }
    });
}
```

Zdrojový kód 7 FragmentPatientInfo - Deaktivace tlačítka zpět

4.6.1.7 FragmentPatientAction

Tento fragment obsahuje několik funkcionalit, která uživateli umožňují úpravy účtu a jiné akce. Jedna z předních funkcí je odeslání pacienta na server. Před samotným odesláním musí dojít ke kontrole, zda má uživatel stále platné přihlašovací údaje. Pokud nemá platné heslo, systém uživatele odhlásí. Pokud již na serveru není ani uživatelské jméno, smaže systém uživateli přístupové údaje do aplikace a následně jej odhlásí. Tím mu systém znemožní i off-line přístup do aplikace. Data se v aplikaci nesmažou. Tato opatření chrání server před zneužitím a dávají správci centrální databáze možnost zablokování nechtěného uživatele. Metoda setOnClickListener() tlačítka odesílání pacienta viz zdrojový kód 8.

```
    synch_button2.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            ServerCommunicator serverCommunicator = new ServerCommunicator();
            if (serverCommunicator.checkICMPConnection()) {
                 // Kontrola VPN spojení
                ResultSet rs = serverCommunicator.checkServerUsers(userLocalName);
                  // Kontrola existence username
                if (rs == null) {
                    Toast.makeText(getActivity(), getString(R.string.dialog6),
Toast.LENGTH_SHORT).show();
                   // Pokud uživatel na sereveru nenalezen, maže se z lokální databáze
a uživatel je odhlášen
                    myDB.deleteUserName(userLocalName);
                    Intent intent = new Intent(getContext(), LoadingActivity.class);
                    getActivity().startActivityForResult(intent, 1);
                } else {
                    try {
                        passHashServer = rs.getString(3);
                    } catch (SQLException throwables) {
                        throwables.printStackTrace();
                    }
            // pokud uživatelské jméno odpovídá, kontroluje se posledně zadané heslo
                    if (userLocalHash.equals(passHashServer)) {
                        DatabaseHelper myDB = new DatabaseHelper(getContext());
                        dateUpdate4 = (da + "." + mnt + "." + ye);
                        // Pokud heslo odpovídá, pacient se odesílá na server
                        if (serverCommunicator.sendPatient(uniqueId, name, sex, date,
arm, weight, dateUpdate4, patientsUser)) {
                            myDB.updateSyncData(uniqueId, "1", dateUpdate4);
                        } else {
                            Toast.makeText(getActivity(), getString(R.string.failure)
, Toast.LENGTH_SHORT).show();
                        }
                    } else {
            //Pokud se na serveru heslo změnilo, pouze se uživatel odhlásí z aplikace.
                        Toast.makeText(getActivity(), getString(R.string.dialog7),
Toast.LENGTH_SHORT).show();
                        myDB.updateUserNameHash(passHashServer, userLocalName);
                                                Intent intent = new
Intent(getContext(), LoadingActivity.class);
                        getActivity().startActivityForResult(intent, 1);
                    }
                }
            } else {
                    Toast.makeText(getActivity(), getString(R.string.vpnChck),
Toast.LENGTH_SHORT).show();
            }
        }
    });
```

Zdrojový kód 8 FragmentPatientAction - Odeslat pacienta na server

4.6.1.8 FragmentPatientMeasure

Tento Fragment funguje na podobném principu jako aktivita PatientActivity a CustomScreen, pouze se jako adapter používá aktivita CustomScreenMeasure a parametery se liší dle diagramu tříd.

4.6.1.9 AddMeasure

Jedná se o vůbec nejsložitější aktivitu systému. V této aktivitě se vytváří měření, je zde umožněno vybírat soubory ze souborového systému, zajišťuje se zde připojení přes bluetooth rozhraní k zařízení MetaWear a je zde vytvořen proces odeslání pacienta, měření a odesílání souborů na centrální SQL server a webový server. Popis funkcionalit je popsán níže.

4.6.1.9.1 Komunikace se zařízením MetaWear

Výběr zařízení MetaWear

Existují různá zařízení MetaWear, která je možno použít pro sběr třesu. Proto bylo zvoleno řešení výběru zařízení takovým způsobem, že se vždy proskenuje celé spektrum příslušných zařízeních a nabídnou se MAC adresy zařízeních v dosahu. To je zajištěno aktivitou ScannerActivity.

Snímání dat

Snímání dat je zajištěno přímým streamem dat do proměnné, a tak nevzniká žádná časová prodleva po naměření dat, během které by se musely data přenášet ze zařízení do aplikace. Toto řešení je však možné použít pouze do 100Hz – 120Hz na jeden senzor. To je hodnota, která je pro projekt snímání třesu rukou na spodní hranici požadavků. V případě, že by se při testování prototypu dospělo k závěru, že reálná frekvence snímání dat je nedostačující, muselo by se přistoupit ke snímání dát tzv. logováním a následným stažením dat ze zařízení do aplikace. Toto řešení však vyžaduje několikanásobně delší čas strávený s pacientem, a proto bylo rozhodnuto zpracovat v tomto projektu snímání v streamovacím režimu. Bylo však zvoleno takové řešení, které umožňuje jednoduchý přechod mezi oběma variantami. Ukázka kódu viz zdrojový kód 9.

```
startButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
// nastavení parametrů senzorů
accelModule.setOutputDataRate(ACC_FREQ);
accelModule.setAxisSamplingRange(ACC_RANGE);
gyroModule.configure()
        .setOutputDataRate(OutputDataRate.ODR_100_HZ)
        .setFullScaleRange(FullScaleRange.FSR_500)
        .commit();
// Nastaven streamovací režim
AsyncOperation<RouteManager> routeManagerResultAccel =
accelModule.routeData().fromAxes().stream(STREAM_KEY).commit();
AsyncOperation<RouteManager> routeManagerResultGyro =
gyroModule.routeData().fromAxes().stream(GYRO_STREAM_KEY).commit();

routeManagerResultAccel.onComplete(new CompletionHandler<RouteManager>() {
    @Override
    public void success(RouteManager result) {
        result.subscribe(STREAM_KEY, new RouteManager.MessageHandler() {
            @Override
            public void process(Message msg) {
                final CartesianFloat axes = msg.getData(CartesianFloat.class);
                 // Stream akcelerometru do logu zařízení
                Log.i(TAG, String.format("Accelerometer: " + ", %s", axes.toString()
));
// Ukládání do proměnné jako message s přízviskem accel pro zobrazení v aplikaci
                sensorMsg(String.format(msg.getTimestampAsString()+",
"+axes.toString()), "accel");
//…
//…
//…
routeManagerResultGyro.onComplete(new CompletionHandler<RouteManager>() {
    @Override
    public void success(RouteManager result) {
        result.subscribe(GYRO_STREAM_KEY, new RouteManager.MessageHandler() {
            @Override
            public void process(Message msg) {
                final CartesianFloat spinData = msg.getData(CartesianFloat.class);
                 // Stream gyroskopu do logu
                Log.i(TAG, String.format("Gyroscope: " + msg.getTimestampAsString() +
", %s", spinData.toString()));
// Ukládání do proměnné jako message s přízviskem gyro pro zobrazení v aplikaci
                sensorMsg(String.format(msg.getTimestampAsString()+ ", " +
spinData.toString()), "gyro");
//…
//…
//…
// Začátek měření
accelModule.enableAxisSampling();
accelModule.start();
gyroModule.start();
//… kód pokračuje, viz příloha
```

Zdrojový kód 9 addMeasure - Snímání třesu akcelerometrem a gyroskopem

Diodová odezva zařízení MetaWear

Dle zadání mělo zařízení adekvátně reagovat rozsvícením diody. Tato funkcionalita byla implementována tak, že po výběru mac adresy a následnému spojení zařízení dvakrát modře blikne. Tím uživatele upozorní, že je připraveno k měření. Když uživatel stiskne tlačítko start, zařízení spustí snímání a začne zeleně přerušovaně blikat. Pokud uživatel přeruší snímání, zařízení 2x červeně blikne na znamení, že je snímání dokončeno.

Ukládání do CSV souboru

Pro ukládání nasnímaných dat byl zvolen formát csv a to z toho důvodu, že se data streamují po jednotlivých osách, a je tedy možno je jednoduše ukládat do dílčích buněk, a tím umožnit následnou editaci v tabulkovém programu. Ukázka formátu, ve kterém se data ukládají do souboru, viz zdrojový kód 10.

```
String accel_entry = String.format("Accelerometer;"+msg.getTimestampAsString()+
";"+axes.x()+";"+axes.y()+";"+axes.z());
String csv_accel_entry = accel_entry + ";";
OutputStream out;
try {
    out = new BufferedOutputStream(new FileOutputStream(path, true));
    out.write(csv_accel_entry.getBytes());
    out.write("\n".getBytes());
    out.close();
} catch (Exception e) {
    Log.e(TAG, "CSV error", e);
}
```

Zdrojový kód 10 addMeasure - Ukládání dat do souboru

4.6.1.9.2 Ověření uživatele pří posílání souborů na webový server

V této části kódu se generuje speciální hash kód pro komunikaci s webovým serverem. Obecně není doporučováno, aby webová stránka přijímala uživatelské jméno a hash hesla, a dle něj udělila přístup. Pokud by totiž došlo k úniku uživatelských jmen a jejich hashů, měl by útočník přístup do webového serveru. Správné řešení je, že webová stránka na přijímací straně přijímá heslo v textové podobě, ze kterého vytvoří hash, a až následně ověřuje tento hash se svou databází.

V aplikaci by tedy uživatel musel při odesílání zadávat heslo, nebo by se muselo heslo v aplikaci uchovávat v textové podobě a aplikace by následně heslo odesílala v textové podobě „do internetu“.

Z tohoto důvodu byl v aplikaci vytvořen speciální hash kód, jenž se generuje z uživatelského jména, hesla a tajného kódu, který mají společný jak aplikace, tak webové rozhraní. Po kombinaci těchto třech údajů se vytvoří speciální hash, který slouží k ověření uživatele na webovém serveru. Pokud by došlo k uniknutí databáze uživatelských jmen a hesel, útočník by k prolomení potřeboval nejen tajný kód, ale ještě informaci, jak má data před zahashováním zkombinovat.

4.6.2 DatabaseHelper

Tato třída obsluhuje lokální SQL databázi. Obsahuje veškeré metody, které jsou potřeba s prací s databází. Tato třída ve velké míře odpovídá třídnímu diagramu, a proto zde nebude blíže popisována.

4.6.3 ServerCommunication

Tato třída se stará o synchronizaci dat s centrální SQL databází. Propojení se serverem probíhá pomocí jdbc konektoru, který je pro tyto účely vytvořen a spravován tvůrci databáze. Ukázku zjištění, zdali se dané uživatelské jméno vyskytuje v databázi, viz zdrojový kód 11.

```
// proměnná userName již otestována na SQL injection.
public static ResultSet checkServerUsers(String userName ) {
    try {
        Class.forName("org.mariadb.jdbc.Driver").newInstance();
        Connection conn =
DriverManager.getConnection("jdbc:mariadb://IP:PORT/NAMEDATABASE?user=USER&password=PA
SSWORD");
        String query  = "SELECT * FROM user_registered WHERE user_name = '" + userName
+ "';";
        Statement st = conn.createStatement();
        ResultSet rs = st.executeQuery(query);
        if (rs.next()) {
            conn.close();
            return rs;
        }
        else {
            conn.close();
            return null;
        }
    }  catch (Exception e) {
        e.printStackTrace();
        return null;
    }
}
```

Zdrojový kód 11 ServerCommunicator - Dotaz na existenci uživatele

4.6.4 Odesílání souborů

Třída GetFiles je zodpovědná za odeslání souborů a ostatních dat na webový server. Přídavná data, která se odesílají spolu se soubory (speciální hash a username) jsou vytvořeny pomocí funkce Parcel ve třídě FileUsersInfo. Po jejich konverzi z GSON na JSON jsou přidána k informacím o souborech a spolu s nimi odeslána na webový server. Následně se čeká na odpověď serveru. To zajišťuje třída ResponseServer. Odpověď je následně dále zpracovávána. Ukázka viz zdrojový kód 12.

```
//…
//…
//…
  //Vytváření RequestBody instance souborů
  RequestBody requestFile1 = RequestBody.create(MediaType.parse("multipart/form-
data"), file1);
  MultipartBody.Part body1 = MultipartBody.Part.createFormData("fale",
file1.getName(), requestFile1);

  RequestBody requestFile2 = RequestBody.create(MediaType.parse("multipart/form-
data"), file2);
  MultipartBody.Part body2 = MultipartBody.Part.createFormData("gale",
file2.getName(), requestFile2);

  RequestBody requestFile3 = RequestBody.create(MediaType.parse("multipart/form-
data"), file3);
  MultipartBody.Part body3 = MultipartBody.Part.createFormData("hale",
file3.getName(), requestFile3);

  RequestBody requestFile4 = RequestBody.create(MediaType.parse("multipart/form-
data"), file4);
  MultipartBody.Part body4 = MultipartBody.Part.createFormData("jale",
file4.getName(), requestFile4);

  Gson gson = new Gson();
  // načítání informací o uživateli
  String patientData = gson.toJson(fileUserInfo);

  RequestBody description = RequestBody.create(okhttp3.MultipartBody.FORM,
patientData);
//Odeslání souborů
  Call<ResponseServer> call = interfaceApi.files(description, body1, body2, body3,
body4);
//… kód pokračuje viz, příloha A
```

Zdrojový kód 12 GetFiles 1 – Odesílání souborů

4.7 **Implementace webové aplikace**

Webová aplikace se dělí na dvě spolu nekomunikující části.
První část se stará o přijímání souborů od aplikace. Druhá část je rozhraní, ve kterém má uživatel možnost zobrazit data pacientů a případně provést nějaké další operace dle uživatelských práv.
Mimo souborovou strukturu webové aplikace se nachází složka „pacienti-include“, která obsahuje soubor server.php. V něm jsou zapsány přihlašovací údaje do centrální SQL databáze a kód, pomocí kterého se vytváří spojení s centrální databází. Tento soubor byl přesunut mimo přístupnou složku z internetu z důvodu bezpečnosti.

4.7.1 Přijímání souborů od aplikace

Zde se provádí autentizace připojení. Po ověření se data zkopírují do příslušné složky a na závěr server odešle odpověď zpět aplikaci, zda byl přenos úspěšný.
Parametry, kterými jsou speciální hash a uživatelské jméno, jsou funkcí json_decode dekódovány, dále jsou zapsány do příslušných proměnných, které jsou dále zpracovávány (je vytvořen speciální hash kód) a následně jsou využity k autentizaci připojení, viz zdrojový kód 13.

```
//…
$user_hashcode = $row["user_hashcode"];
    $input = '78fdfg554'.$sender_name.''.$user_hashcode.'';
    $hashedPasswordNew = hash('sha256', $input);
    if($hashedPasswordNew == $sender_pass){
        $message = "Your pass is correct - Your files are uploaded
successfully!";
//… kód pokračuje, viz příloha B
```

Zdrojový kód 13 acceptData.php - Ověření přístupu

4.7.2 Zobrazení pacientů a měření

V této části je zpracován jednoduchý formulář pro přihlášení uživatelů a následně dvě přístupová rozhraní, jedno pro uživatele s právy „user“ a druhé pro uživatele s právy „superuser“.

Přihlašovací formulář
Při implementaci přihlašovacího formuláře byly provedeny základní bezpečnostní prvky, přestože přístup k tomuto webovému rozhraní je pouze přes VPN. Přihlašovací formulář obsahuje ochranu proti SQL injection (Zdrojový kód 14) a při zadání chybných přihlašovacích údajů dochází k zdržení pěti vteřin.

```
if(!(preg_match("/(;)|(=)|(')|(\")|(\\\\)|( )/", $username) === 0)) {
    $errName = '<p class="errText">Zakázané jsou mezera a tyto znaky: \ ; \" \\
= \' </p>';
}
```

Zdrojový kód 14 index.php - Zakázané znaky

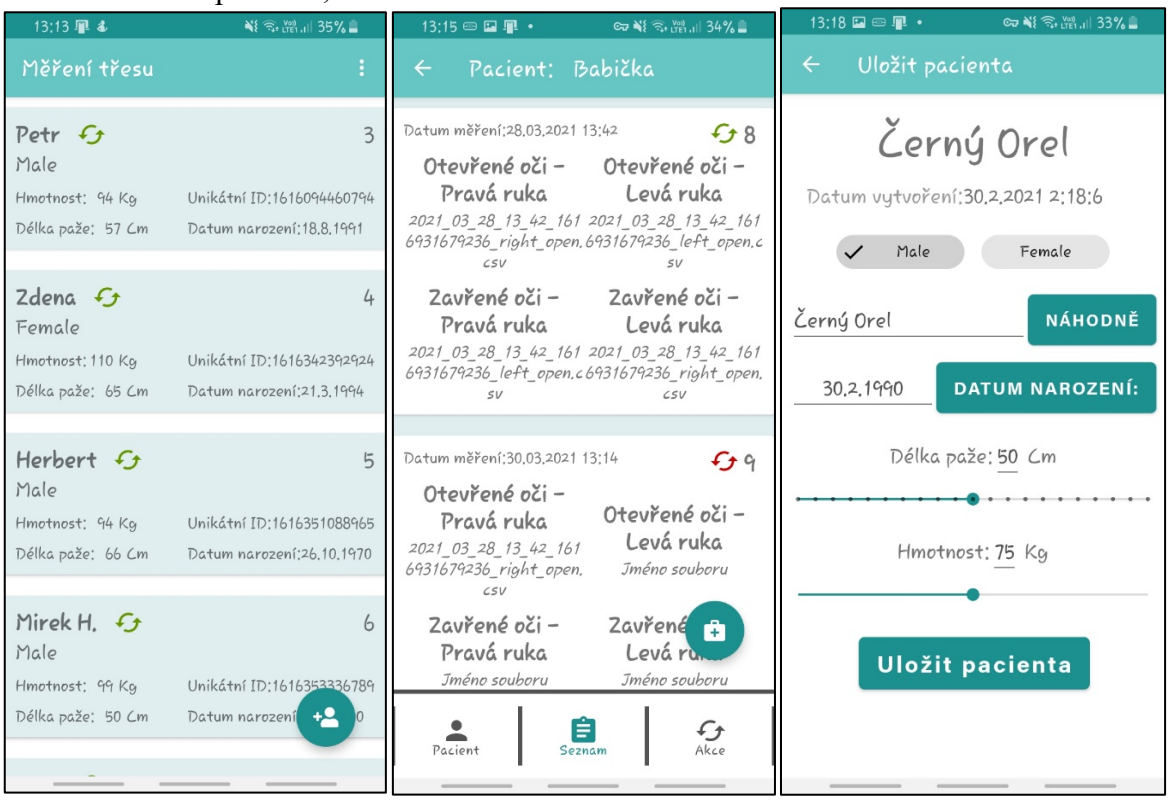
Zobrazení pacientů
Zobrazování pacientů se provádí pomocí dvou souborů admin.php a user.php, na které je uživatel přesměrován dle svých uživatelských práv. Při jakémkoli neoprávněném přístupu (přesměrování na stránku s jinými právy) je uživatel automaticky přesměrován na index.php. Navázání spojení s databází se provádí až po autentizaci uživatele, aby nemohlo dojít k přetížení SQL databáze (přetížení pomocí vyvolání mnohonásobného připojení). Pro implementaci grafického rozhraní byl použit nástroj Bootstrap. U uživatele s právy „superuser“ byly přidány funkcionality přidání uživatele, smazání pacienta a vymazání uživatele, viz zdrojový kód 15 .

```
include "../pacienti-include/server.php";
if(isset($_POST['del_user'])) {
     $unamedel = mysqli_real_escape_string($mysqli, $_POST['userdel']);
    if (empty($_POST['userdel']) ) {
        $errDel = 'Username nemůže být prázdné';
    }else {
        if (!(preg_match("/(;)|(=)|(')|(\")|(\\\\)|( )/", $unamedel) === 0)) {
            $errDel = '<p class="errText">Zakázané jsou mezera a tyto znaky: \
; \" \\ = \' </p>';
        } else {
            $sql10 = "DELETE FROM $useTable WHERE $usernameColl =
'".$unamedel."' ";
            if ($mysqli->query($sql10) === TRUE) {
                $errDel = "Smazan uživatel '".$unamedel."' ";
            } else {
                $errDel = "Smazání uživatele se nezdařilo ";
//… kód pokračuje, viz příloha B
```

Zdrojový kód 15 admin.php - Samzání uživatele

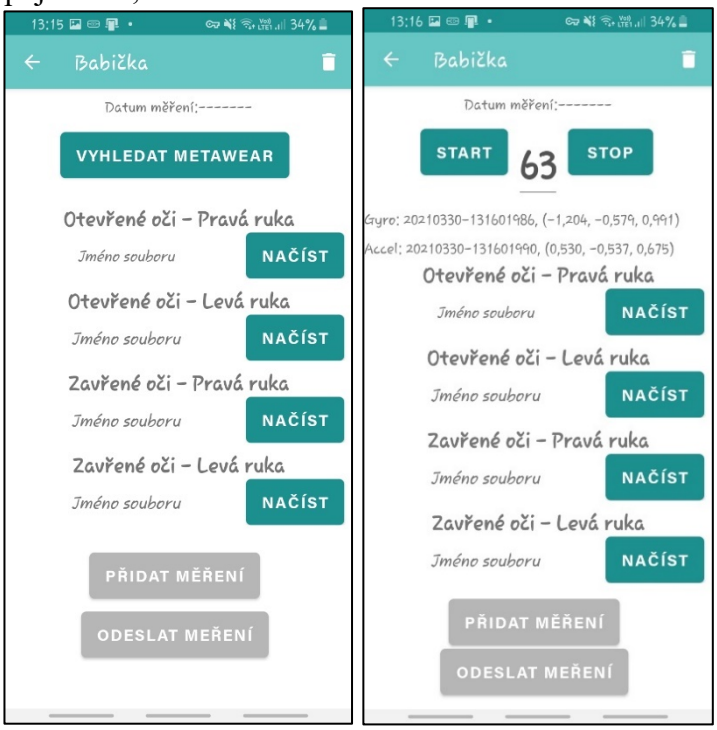
4.8 **Výsledné grafické rozhraní mobilní aplikace**

Grafické rozhraní mobilní aplikace se podařilo vypracovat do podoby, která se od návrhu liší jen mírně. Mírné změny proběhly na obrazovce přehledu pacientů a přehledu měření a tvorbě nového pacienta, viz obrázek 4.21.



Obrázek 4.21 Výsledné grafické zpracování - přehledy

Jediná aktivita, která dostála výrazných změn byla aktivita přidání měření. V původním návrhu bylo předpokládáno, že začátek a konec měření bude vyvoláván ze speciálního okna, viz obrázek 4.16. Nakonec se při implementaci ukázalo, že bude vhodnější tlačítka start a stop umístit přes tlačítko Vyhledat MetaWear, jenž se objeví až ve chvíli, kdy bude zařízení připraveno k připojení viz, obrázek 4.22.



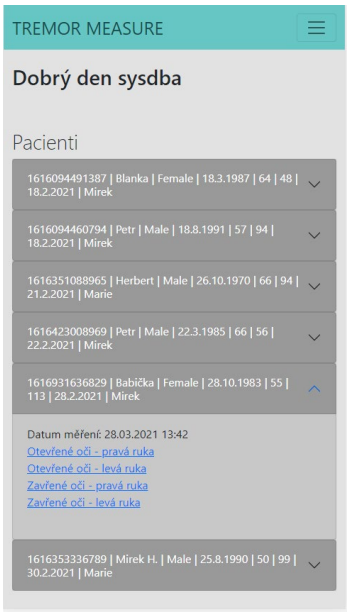
Obrázek 4.22 Výsledné grafické zpracování – Přidat měření (Zdroj: vlastní zpracování)

Zde si lze všimnout, že tlačítka Přidat měření a Odeslat měření jsou deaktivována. Tyto tlačítka se uvolní až ve chvíli, kdy bude přidáno první měření danému cíli, viz obrázek 4.23.

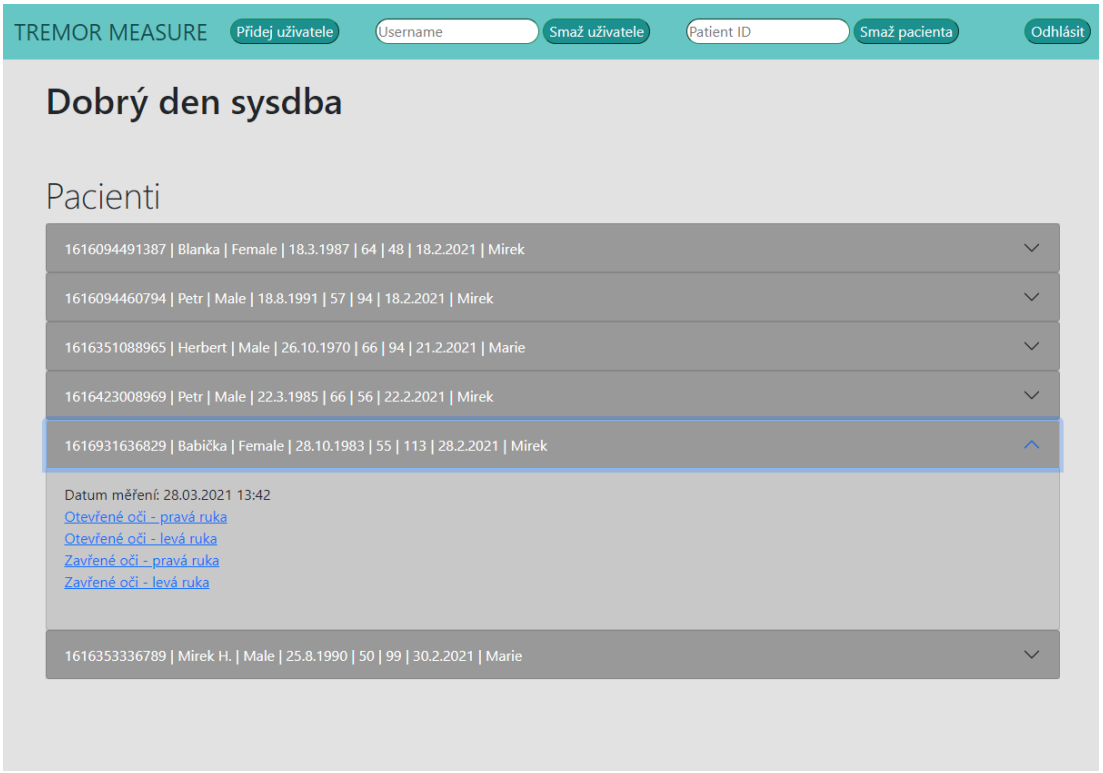
Obrázek 4.23 Přidat měření - Závěrečná obrazovka

4.9 **Výsledné grafické rozhraní webové aplikace**

Cílem výsledného grafického návrhu webové aplikace bylo vytvořit responsivní web, proto jej můžeme rozdělit do dvou částí na mobilní, viz obrázek 4.24 a počítačovou, viz obrázek 4.25.



Obrázek 4.25Grafické zpracování mobilní verze webového rozhraní



Obrázek 4.24 Grafické zpracování počítačové verze webového rozhraní

5 Výsledky a diskuse

5.1 Komplikace při návrhu a implementaci

První komplikace nastala již při samotném výběru technologií a softwaru. Vzhledem k mnoha oblastem, ve kterých bylo potřeba zajistit kompatibilitu mezi různými částmi systému, bylo zapotřebí před samotným návrhem nalézt několik různých řešení, otestovat kompatibilitu mezi nimi, z nich vybrat nejlepší možná řešení, a až následně je navrhnout k implementaci.
Druhá a podobná komplikace nastala při návrhu grafického rozhraní. Před samotným návrhem aplikace bylo potřeba udělat analýzu aplikací na trhu, vyhledat možnosti interakce s uživatelem a až poté nejlepší možná řešení navrhnout k implementaci.

5.2 Testování prostředí

Během vypracovávání diplomové práce nebyly kvůli epidemiologické situaci umožněny osobní konzultace, a i možnosti testování byly velice omezené. Průběžné výsledky byly pravidelně testovány a konzultovány s konzultantem a po implementaci nejdůležitějších komponent bylo zřízeno testovací prostředí, které věrně simulovalo reálný provoz (webový server, MariaDB SQL databáze umístěná na serveru, VPN spojení se serverem). Po dokončení implementace byl proveden videostream, na kterém se prezentovala funkčnost všech částí systému a vedoucího práce s konzultantem se mohli seznámit s dosaženými výsledky.

5.3 Dosažené výsledky

Veškeré požadavky, které byly na software kladeny, se podařilo naplnit. Nad rámec zadání došlo i k implementaci snímání třesu přes rozhraní bluetooth a zakomponování zařízení MetaWear do aplikace. Tímto způsobem došlo k vytvoření webového rozhraní nejen pro správce centrální databáze, ale i pro uživatele, kteří si takto mohou kontrolovat poslaná data na server. Z důvodu změny konceptu byl vypracován design i pro webovou část.

5.3.1 Souhrn implementovaných funkcionalit

Obecné funkcionality

- Při realizaci byl kladen důraz na využívání posuvníků a tlačítek namísto psaní na klávesnici mobilního zařízení.
- Data se ukládají tak, aby nepodléhaly směrnici GDPR. Uživatel je před uložením pacienta na tuto skutečnost vždy upozorněn.

Mobilní aplikace

- Prototyp aplikace byl testován na reálném zařízení.
- Aplikace obsahuje dvě jazykové mutace.
- Systém funguje bez nutnosti přístupu k internetu (kromě synchronizace a prvního přihlášení).

Pacient

- Každý pacient má své unikátní ID, přezdívku, datum narození, pohlaví, délku paže a hmotnost.
- V systému je možnost přezdívku náhodně generovat.
- V systému je možnost pacienta upravit i smazat.
- Pacient může mít přiděleno více na sobě nezávislých měření.

Měření

- Každé měření má unikátní ID, datum měření a čtveřici naměřených souborů.
- Uživatel má možnost soubory přidat ze souborové struktury nebo je naměřit zařízením MetaWear.
- Při komunikaci se zařízením MetaWear je využito diodové signalizace.

Přihlašování uživatelů

- Prvotní přihlašování do aplikace je podmíněno vytvořením uživatele na centrálním serveru.
- Při potřebě zablokovat uživatele stačí upravit data na centrálním serveru, uživatel bude při příštím kontaktu se serverem zablokován.

Webové rozhraní

- Uživatel má možnost zobrazit měření a stáhnout příslušná data.
- Byla vytvořena práva „User“ a „Superuser“.
- „User“ může zobrazit pouze svá data a měření.
- „Superuser“ má možnost zobrazit všechna měření.
- „Superuser“ má možnost přidávat uživatele, mazat uživatele a mazat pacienty i s jejich daty.

5.4 Vzorové použití stávajícího řešení

5.4.1 Vytvoření uživatele

Správce centrální SQL databáze se připojí přes VPN do stejné sítě, jako je server a zadá adresu webového serveru do prohlížeče. Přihlásí se a vytvoří přihlašovací údaje nového uživatele. Následně zašle novému uživateli email s přihlašovacími údaji, aplikací a VPN certifikátem.

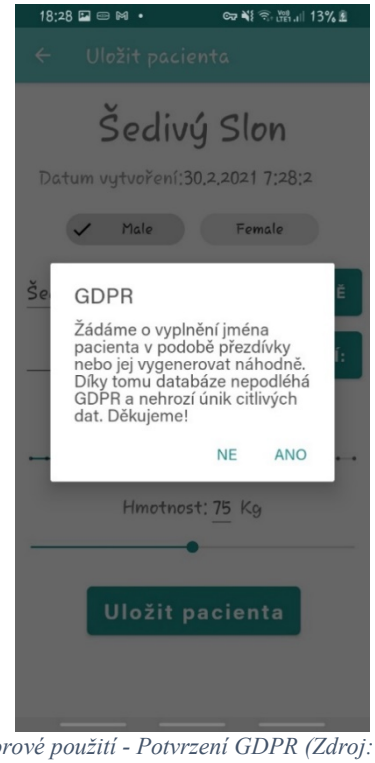
5.4.2 První spuštění

Uživatel dostane emailem VPN certifikát, aplikaci a přihlašovací údaje. Dle pokynů si nastaví VPN a nainstaluje aplikaci. Po spuštění vyplní přihlašovací údaje a potvrdí zkopírování přihlašovacích údajů ze serveru do lokální databáze, viz obrázek 5.1 .

Obrázek 5.1 Vzorové použití - Stažení uživatele ze serveru (Zdroj: vlastní zpracování)

5.4.3 **Přidání pacienta**

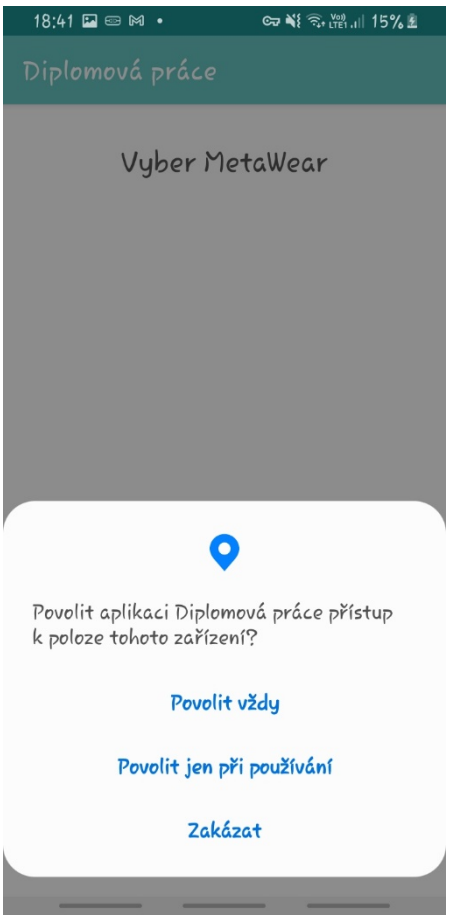
Po přihlášení do systému uživatel klikne na tlačítko přidat v pravém dolním rohu a zobrazí se mu obrazovka přidání nového pacienta. Ještě před samotným výběrem parametrů daného pacienta musí uživatel potvrdit, že jméno pacienta musí být pouze přezdívka, viz obrázek 5.2 .



Obrázek 5.2 Vzorové použití - Potvrzení GDPR (Zdroj: vlastní zpracování)

5.4.4 **Přidání a odeslání měření**

Po přidání pacienta uživatel klikne na přidaného pacienta a zobrazí se mu přehled pacienta. Ve spodním menu překlikne buď na prostřední ikonu měření nebo na pravou ikonu akcí a vybere „Přidat měření“, viz obrázek 4.21. Po otevření obrazovky si Android vyžádá práva do souborové struktury. Ty je potřeba potvrdit, jinak by uživatel neměl přístup k souborům zařízení.
Poté stiskne tlačítko Vyhledat MetaWear a ze seznamu zařízení vybere příslušnou MAC adresu MetaWear zařízení, které vlastní. Před samotným skenováním musí uživatel potvrdit, že bere na vědomí, že aplikace bude využívat rozhraní Bluetooth a potvrdit příslušná práva, viz obrázek 5.3.



Obrázek 5.3 Vzorové použití - Potvrzení práv na rozhraní bluetooth (Zdroj: vlastní zpracování)

Poté klikne na tlačítko připojit a zařízení MetaWear dvakrát blikne modře jako potvrzení spojení. V tu chvíli se zobrazí dialogové okno a vyzve uživatele ke zvolení cíle. Poté se obrazovka změní a místo tlačítka „Připojit“ se objeví tlačítko „Start“, „Stop“ a časovač nastavený na 65 sekund. Uživatel si přenastaví časovač na požadovanou hodnotu a stiskne tlačítko „start“. Po odpočítání časovače se soubor automaticky uloží k danému cíli,

viz obrázek 4.22 a uživatel může postup opakovat do té doby, než budou u všech cílů příslušná měření. Následně potvrdí odeslání souborů a počká na odpověď serveru. Tím měření ukončí a pacient může odejít.

Na závěr si uživatel při zapnutém VPN připojení zadá do prohlížeče adresu serveru, přihlásí se svými přihlašovacími údaji a zkontroluje, že vše zadal správně a stáhne si soubor s měřeními. Ukázka souboru viz tabulka 5.1.

SENSOR	TIMESTAMP	X_AXIS	Y_AXIS	Z_AXIS
Accelerometer	20210328-134224032	0.58935547	0.73828125	0.47705078
Accelerometer	20210328-134224033	0.6479492	0.64453125	0.6352539
Gyroscope	20210328-134224035	-201.37196	159.8933	40.670734
Accelerometer	20210328-134224036	0.73657227	0.5480957	0.75341797
Gyroscope	20210328-134224037	-208.65854	166.32622	42.484756
Accelerometer	20210328-134224039	0.7668457	0.47045898	0.8781738
Gyroscope	20210328-134224039	-243.23172	190.2744	54.039635
Accelerometer	20210328-134224039	0.84399414	0.39111328	0.87719727
Gyroscope	20210328-134224040	-290.27438	223.44513	75.99085
Accelerometer	20210328-134224078	0.9003906	0.29248047	0.5961914
Gyroscope	20210328-134224078	-305.39636	264.74084	103.33842

Tabulka 5.1 Ukázka dat nasnímaným zařízením MetaWear (Zdroj: vlastní zpracování)

5.5 **Možnosti rozšíření**

Po závěrečné prezentaci a následné konzultaci byl nastíněn další možný postup. Jednou z variant by bylo umožnit aplikaci propojení s chytrými náramky nebo hodinkami a umožnit snímání třesu širšímu okruhu pacientů. Tomuto kroku však musí předcházet odborná analýza, jestli jsou takovéto nástroje schopny zajistit spolehlivé snímání třesu pro daný výzkum. Také by bylo vhodné nejprve provést testování metody snímání třesu na větším vzorku pacientů, aby se mohly dosavadní výsledky na větším vzorku potvrdit nebo vyvrátit.

6 Závěr

Cílem diplomové práce bylo navrhnout a implementovat systém pro registr pacientů a jejich měření, který bude uživatelsky přívětivý, nebude vyžadovat technické vzdělání, nebude vyžadovat trvalé připojení k internetu a bude bezpečně odesílat získaná data na centrální SQL server za účelem pořízení třesu pacientů trpících roztroušenou sklerózou. V druhé části měl být vytvořen webový přístup k centrální SQL databázi, pomocí kterého měla být data zpřístupněna v hierarchii pacientů a jejich měření. Jako volitelná součást diplomové práce bylo zpřístupnit přes rozhraní bluetooth snímací zařízení MetaWear, pomocí něhož mělo být možné naměřit požadovanou čtveřici souborů, která se měla spolu s daty odesílat na centrální server.

Cíl popsaný v předešlém odstavci jsem se snažil naplnit ve čtvrté kapitole - Vlastní práce, při využití dovedností získaných při studiu oboru Systémové inženýrství a informatika. Samotnému návrhu a implementaci předcházelo několikaměsíční období studia dané problematiky a byl konzultován nejen s pracovníky univerzity, ale i odborníky z praxe.

Při implementaci navrhovaného řešení nenastal žádný vážnější problém a všechny požadavky kladené na systém se podařilo implementovat, takže bylo přistoupeno k doimplementování volitelných částí, které se také podařilo implementovat bez větších komplikací.

V páté kapitole Výsledky a diskuze byly shrnuty komplikace v rané fázi návrhu. Prezentace dosažených výsledků byla provedena pomocí videostreamu. Následně v kapitole Testování prostředí byl popsán způsob testování, který byl z důvodu pandemické situace poněkud nestandardní, a systém mohl být otestován pouze konzultanty a nejbližší rodinou. Testování probíhalo na reálném mobilním zařízení i serveru a systém byl otestován v prostředí podobném ostrému provozu. Ze závěrů testování bylo sestaveno vzorové použití stávajícího systému, kde jsou popsány postupy při běžném používání. Na závěr kapitoly jsou shrnuty možnosti rozvoje daného řešení.

Výstupem práce je tedy funkční řešení, které obsahuje aplikaci včetně zdrojových kódů pro zařízení se systémem Android, podklady pro vytvoření struktury centrální SQL databáze a zdrojové kódy webového rozhraní pro přístup k databázi a pro příjem souborů z aplikace.

7 Seznam použitých zdrojů

1. ABDULLAH, Hasan, 2017. Image or any File Upload to Server using Retrofit in Android App. *Github.com* [online]. [cit. 2021-03-31]. Dostupné z: https://github.com/hasancse91/Android-File-Upload-To-Server

2. ALUSI, S. H., 2001. A study of tremor in multiple sclerosis. *Brain* [online]. **124**(4), 720-730 [cit. 2021-03-20]. ISSN 14602156. Dostupné z: doi:10.1093/brain/124.4.720

3. BAKKEN, Stig, Alexander AULBACH, Egon SCHMID a Torben LARS, 2000. *PHP Manual*. 2. vydání: iUniverse. ISBN 9780595132287. Dostupné také z: https://www.php.net/manual/en/index.php

4. BOSCH-SENSORTEC, Bosch-sensortec a Bosch-sensortec BOSCH-SENSORTEC, 2021. BMI160: IMU combining accelerometer and gyroscope. *Bosch-sensortec.com* [online]. [cit. 2021-01-12]. Dostupné z: https://www.bosch-sensortec.com/products/motion-sensors/imus/bmi160/

5. CARPINELLA, Ilaria, Davide CATTANEO a Maurizio FERRARIN, 2015. Hilbert–Huang transform based instrumental assessment of intention tremor in multiple sclerosis. *Journal of Neural Engineering* [online]. **12**(4) [cit. 2021-03-31]. ISSN 1741-2560. Dostupné z: doi:10.1088/1741-2560/12/4/046011

6. COMBY, B., G. CHEVALIER a M. BOUCHOUCHA, 2008. A new method for the measurement of tremor at rest. *Archives Internationales de Physiologie, de Biochimie et de Biophysique* [online]. **100**(1), 73-78 [cit. 2021-03-20]. ISSN 0778-3124. Dostupné z: doi:10.3109/13813459209035262

7. DEVELOPER.ANDROID.COM, 2020. Platform Architecture. *Android Developers* [online]. California, U.S.: Google and Open Handset Alliance n.d. [cit. 2021-03-25]. Dostupné z: https://developer.android.com/guide/platform

8. DOBSON, R. a G. GIOVANNONI, 2018. Multiple sclerosis – a review. *European Journal of Neurology* [online]. **26**(1), 27-40 [cit. 2021-03-25]. ISSN 1351-5101. Dostupné z: doi:10.1111/ene.13819

9. FAKHROUTDINOV, Kirill, 2020. Unified Modeling Language (UML) description. *Uml-diagrams.org* [online]. Uml-diagrams.org [cit. 2021-03-23]. Dostupné z: https://www.uml-diagrams.org/

10. HAYES, Alexander, Mayukh DAS, Phillip ODOM a Sriraam NATARAJAN, 2017. User Friendly Automatic Construction of Background Knowledge. In: *Proceedings of the Knowledge Capture Conference* [online]. New York, NY, USA: ACM, s. 1-8 [cit. 2021-03-24]. ISBN 9781450355537. Dostupné z: doi:10.1145/3148011.3148027

11. HEROUT, Pavel, 2007. *Učebnice jazyka Java*. 3., rozš. vyd. České Budějovice: Kopp. ISBN 80-7232-323-7.

12. CHOUDHRY, Asad Ali, 2020. Get File Path From URI in Android JAVA: Utility Class to Get File Path From URI in Android. *Handyopinion.com* [online]. Handy Opinion [cit. 2021-03-31]. Dostupné z: https://handyopinion.com/get-file-path-from-uri-in-android-java/

13. MARIADB.ORG, 2015. About MariaDB Server. *Mariadb.org* [online]. Delaware, USA: MariaDB Foundation [cit. 2021-03-24]. Dostupné z: https://mariadb.org/about/

14. MARTÍNEZ, Francisco a Ambrosio ÁLVAREZ, 2005. A Precise Approach for the Analysis of the UML Models Consistency. AKOKA, Jacky, Stephen W. LIDDLE, Il-Yeol SONG et al. *Perspectives in Conceptual Modeling* [online]. Berlin, Heidelberg: Springer Berlin Heidelberg, s. 74-84 [cit. 2021-03-23]. Lecture Notes in Computer Science. ISBN 978-3-540-29395-8. Dostupné z: doi:10.1007/11568346_9

15. NETCRAFT.COM, 2021. Other vendor and hosting news. *Netcraft.com* [online]. Belmont, UK: Netcraft Ltd [cit. 2021-03-26]. Dostupné z: https://news.netcraft.com/archives/category/web-server-survey/

16. OMG.ORG, 2017. About the unified modeling language specification version 2.5.1. *Omg.org* [online]. Milford, USA: An OMG® Unified Modeling Language® Publication [cit. 2021-03-23]. Dostupné z: https://www.omg.org/spec/UML

17. OTTO, Mark, 2020. Bootstrap 5: The most popular HTML, CSS, and JavaScript framework for developing responsive, mobile first projects on the web. *Github.com* [online]. San Francisco, CA [cit. 2021-03-31]. Dostupné z: https://github.com/twbs/bootstrap

18. PUSCHMANN, Andreas a Zbigniew WSZOLEK, 2011. Diagnosis and Treatment of Common Forms of Tremor. *Seminars in Neurology* [online]. **31**(01), 065-077 [cit. 2021-03-20]. ISSN 0271-8235. Dostupné z: doi:10.1055/s-0031-1271312

19. SCHIMKE, Yuri, 2019. OkHttp. *Square.github.io* [online]. Square, Inc. [cit. 2021-03-31]. Dostupné z: https://square.github.io/okhttp

20. SQLITE.ORG, 2018. About SQLite. *Sqlite.org* [online]. Georgia, USA: Richard Hipp [cit. 2021-03-24]. Dostupné z: www.sqlite.org/about.html

21. SUBRAMANIAM, Venkat, 2019. *Programming Kotlin*. Raleigh, NC: Pragmatic Bookshelf. ISBN 9781680506358.

22. TME.EU, 2020. Jak funguje a k čemu slouží akcelometr. *Tme.eu* [online]. [cit. 2021-02-21]. Dostupné z: https://www.tme.eu/cz/news/library-articles/page/22568/jak-funguje-a-k-cemu-slouzi-akcelerometr/

23. TSAI, Eric, 2016. MetaWear Android API. *Github.com* [online]. Github.com [cit. 2021-03-31]. Dostupné z: https://github.com/mbientlab/MetaWear-SDK-Android

24. VAN DRODGELEN, Mike, 2015. *Android Studio Cookbook*. 2015. Birmingham, England: Packt Publishing. ISBN ISBN 9781785286186.

25. VAROL, Tolga, 2019. *Comparison of Consumer-Grade MEMS IMUs in UBI Context*. KTH Royal Institute of Technology, SE-100 44 Stockholm, Sweden. Diplomová práce. KTH, School of Electrical Engineering and Computer Science (EECS).

26. VISUAL-PARADIGM.COM, 2016. Wireframe. In: *Visual-paradigm.com* [online]. Hong Kong: Visual Paradigm [cit. 2021-03-26]. Dostupné z: https://www.visual-paradigm.com/learning/handbooks/agile-handbook/wireframe.jsp

27. VOLKHART, Marius, 2016. Gson. *Github.com* [online]. Google Inc. [cit. 2021-03-31]. Dostupné z: https://github.com/google/gson

28. ZEMAN, Josef, 2020. *Osobní sdělení*. (Technická fakulta ČZU v Praze, Kamýcká 129, 165 21 Praha-Suchdol) dne 22.5.2020.

29. ZEMAN, Josef, 2021. *Diagnostika RS z klidového tremoru*. (in preparation).

8 Přílohy

Odkazovaný seznam příloh

K práci je přiložené CD se všemi zdrojovými daty:

Příloha A - Přiložený zip soubor se zdrojovým kódem Android aplikace.
Příloha B - Přiložený zip soubor se zdrojovým kódem Webové aplikace.
Příloha C - Přiložený PDF soubor s grafického návrhem.
Příloha D - Přiložený soubor aplikace ve formátu apk.
Příloha E - Přiložené příkazy pro tvorbu databáze.