ČESKÉ VYSOKÉ
UČENÍ TECHNICKÉ
V PRAZE

F3

Fakulta elektrotechnická
Katedra počítačů

Bakalářská práce

Administrační uživatelské rozhraní a API pro sběr dat na projektu TERESA

Bakalářská práce

Martin Funda

Srpen 2022
Vedoucí práce: doc. Ing. Miroslav Bureš, Ph.D.

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Funda Martin** Personal ID number: **474469**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Computer Science**

Study program: **Software Engineering and Technology**

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Administrative user interface and API for Teresa project data collection

Bachelor’s thesis title in Czech:

Administrační uživatelské rozhraní a API pro sběr dat na projektu Teresa

Guidelines:

Masse, Mark. REST API Design Rulebook. O’Reilly, 2011.
Madhusudhan, Konda. Just Hibernate. O’Reilly, 2014.
Turnbull, James. The Docker Book. 2019.
Craig Walls, Spring in Action, Fifth Edition, Manning, 2018.

Bibliography / sources:

Masse, Mark. REST API Design Rulebook. O’Reilly, 2011.
Madhusudhan, Konda. Just Hibernate. O’Reilly, 2014.
Turnbull, James. The Docker Book. 2019.
Craig Walls, Spring in Action, Fifth Edition, Manning, 2018.

Name and workplace of bachelor’s thesis supervisor:

doc. Ing. Miroslav Bureš, Ph.D. System Testing IntelLigent Lab FEE

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **27.01.2022** Deadline for bachelor thesis submission: **15.08.2022**

Assignment valid until: **30.09.2023**

doc. Ing. Miroslav Bureš, Ph.D. Supervisor's signature	Head of department's signature	prof. Mgr. Petr Páta, Ph.D. Dean's signature

III. Assignment receipt

The student acknowledges that the bachelor’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor’s thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student’s signature

Poděkování / Prohlášení

Rád bych poděkoval doc. Ing. Miroslavu Burešovi, Ph.D., za cenné rady, připomínky a vstřícnost při konzultacích během vypracovávání bakalářské práce. Dále chci poděkovat doc. Kateřině Neumannové, Ph.D., za spolupráci na analýze požadavků pro systém TERESA.

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 6. 8. 2022

.....

Abstrakt / Abstract

Tato práce je zaměřena na vytvoření administračního uživatelského prostředí a API pro sběr dat z chytrých wearable zařízení na projektu TERESA (TEleREhabilitation Self-training Assistant), který podporuje studie vlivu telecoachingu na pacienty s plicním onemocněním.

Součástí práce je sběr požadavků od uživatelů, jejich analýza, návrh řešení, implementace a následné testování a provoz celého systému. Z nasbíraných dat mohou lékaři skrze administrační rozhraní generovat exporty pro další využití. Dále mohou spravovat jednotlivé komponenty systému jako zaregistrovaná zařízení, údaje o pacientech či nastavovat dotazníky pro pacienty. Aplikace tak zjednodušuje práci a umožňuje opakované provádění studií.

Klíčová slova: telecoaching, telerehabilitation, wearables, COVID-19, TERESA

This thesis focuses on creating an administration user interface and API for data collection from smart wearable devices on the TERESA (TEleREhabilitation Self-training Assistant) project, which is designed to support studies of the impact of telecoaching on patients with lung disease.

The work includes collecting requirements from users, analyzing them, designing a solution, implementing it and then testing and operating the whole system. From the collected data, physicians can generate exports for further use through an administration interface. They can also manage individual system components such as registered devices, patient data or set up patient surveys. The application thus simplifies the work and allows repeated studies.

Keywords: telecoaching, telerehabilitace, wearables, COVID-19, TERESA

Title translation: Administrative user interface and API for TERESA project data collection (Bachelor thesis)

Obsah /

Tabulky / Obrázky

Kapitola 1
Úvod

Projekt TERESA (TEleREhabilitation Self-training Assistant) [1] vznikl začátkem roku 2021. Jde o společný projekt Českého vysokého učení technického v Praze (ČVUT), Fakultní nemocnice Hradec Králové (FN HK), Univerzity obrany (UNOB) a Univerzity Palackého v Olomouci (UPOL). Cílem projektu TERESA je vyvinout komplexní a snadno spravovatelný systém pro podporu telerehabilitace s možností sběru dat sledovaných pacientů a s uživatelsky přívětivou aplikací jak pro pacienty, tak i lékaře, kteří mají za průběh telerehabilitace odpovědnost. Tento projekt má zároveň umožnit podporu studií, které se zabývají pozorováním vlivu telecoachingu na stav pacientů po těžkém průběhu onemocnění nemocí COVID-19 a dalších těžkých respiračních onemocnění.

1.1 Telerehabilitace

Telerehabilitace [2] neboli distanční terapie, e-rehabilitace, znamená poskytování rehabilitačních služeb prostřednictvím telekomunikačních sítí a internetu. Mezi metodami telerehabilitace se nachází i sběr dat od pacientů skrze nositelná zařízení. Vedle pojmu telerehabilitace se zároveň můžeme setkat i s pojmem telecoachingu. Jedná se o upřesňující název pro vedení pacienta ke zlepšování celkové kondice a zvýšení pohybové aktivity.

Hlavní výhodou telerehabilitace je její dostupnost a plošná využitelnost [3]. Pacienti v odlehlých lokacích nejsou nuceni denně dojíždět do nemocnice, což znamená významnou časovou i finanční úsporu. Terapie se účastní v domácím prostředí a s lékařem komunikují primárně distančně pomocí telefonu či internetu.

Velký význam telerehabilitace ukázala současná pandemie COVID-19, která ještě více zhoršila dostupnost klasické rehabilitace, ať už z pohledu vytížení nemocnic a zdravotnického personálu, tak i z pohledu omezení kontaktu a nutné izolace mnoha pacientů v jejich domácnostech. Telerehabilitace se v této době tedy dočkala širšího využití a podnítila rychlejší vývoj informačních technologií, zejména IoT (Internet of Things), v oblasti zdravotnictví.

Jedním z možných způsobů telerehabilitace je kombinace využití mobilní aplikace a pedometru. Pacient tak může okamžitě získávat zpětnou vazbu o průběhu svých pohybových aktivit během dne a tuto aktivitu na základě konzultací s lékaři postupně upravovat, například formou navyšování cílového denního počtu kroků či délky trvání pohybové aktivity střední intenzity. Tato varianta je obzvláště účinná pro pacienty, kteří trpí chronickou obstrukční plicní nemocí (CHOPN).

1.2 Význam telerehabilitace pro pacienty s plicním onemocněním

Cílová skupina, pro které je systém TERESA vytvořen, byla určena jako skupina pacientů trpící CHOPN. Stav těchto pacientů bývá ve většině případů ovlivněn nedostatečnou pohybovou aktivitou a tolerancí zátěže. Pacienti se často pohybovým aktivitám

vyhýbají z důvodu dušnosti či svalové slabosti, což vede ke zhoršování stavu a zvýšení rizika hospitalizace či smrti [3].

CHOPN nemůže být úplně vyléčena, ale je možné zmírnit závažnost symptomů a omezit rozvoj onemocnění. Vhodnou metodou pro tyto účely je plicní rehabilitace, která redukuje dušnost a zvyšuje toleranci těla na zátěž. Plicní rehabilitaci je vhodné doplnit o telerehabilitaci a to jak v průběhu samotné plicní rehabilitace, tak i jako pokračující intervenci po jejím ukončení. Úspěšné provedení této strategie ukazuje velmi významné zlepšení kondice a pohybové aktivity pacientů, konkrétně v denním počtu kroků a rychlosti aktivní chůze [4].

1.3 Existující řešení

Lze nalézt mnoho příkladů, které ukazují význam senzorových sítí a wearables (tj. chytrých zařízení, které uživatelé nosí na svém těle) v oblasti medicíny [5]. Existuje řada komerčně dostupných řešení, které svému uživateli dovolí v reálném čase sledovat jeho fyzickou aktivitu a zdravotní funkce. Jde primárně o různé druhy pedometrů, chytrých hodinek, náramků či mobilních telefonů. Nicméně samotná tato zařízení nejsou určena pro pomoc chronicky nemocným pacientům, kteří vyžadují péči a konzultace s lékařem [1]. V takovém případě je nezbytná rehabilitace či telecoaching. I pro účely telecoachingu však vzniká řada řešení, která jsou podporovaná částečně automatizovanými aplikacemi [6] a motivují jejich uživatele k pohybové aktivitě.

V České republice je však projekt TERESA v současné době jedinečným řešením, které podporuje telecoaching a zároveň umožňuje provádět studie jeho vlivu na pacienty s chronickým plicním onemocněním.

1.4 Pilotní běh projektu

Díky inkrementálnímu vývoji bylo možné projekt již otestovat a využít i v praxi. První pilotní běh projektu TERESA byl proveden v roce 2021 v období duben-červenec [7]. Během této doby zúčastnění pacienti nosili náramky typu Xiaomi Mi 5. Získaná data pravidelně zasílali pomocí mobilní aplikace ke zpracování. Z těchto dat byly pro lékaře generovány týdenní reporty, které byly použity k vyhodnocování stavu pacientů a komunikaci s nimi. Tento pilotní běh přinesl řadu cenných poznatků, které byly využity k dalšímu vývoji systému, ať už jde o opravy defektů či zjištění potřeby úplně nové funkcionality.

1.5 Cíl této práce

Aby bylo možné zpracovávat a centrálně uchovávat data o aktivitě pacientů účastnících se studie, která jsou sesbírána pomocí chytrých wearable zařízení, je nutné vytvořit webové API (application programming interface), které tato data bude od pacientů přijímat, zpracovávat a ukládat do jednotné databáze.

Zároveň pro usnadnění práce se systémem vznikne webové administrační rozhraní, které dovolí administrátorům a lékařům jednoduše spravovat jednotlivé části systému a přistupovat k sesbíraným datům ve formě exportů, které budou dále využívány pro účely studie.

Cílem této práce je v rámci agilního procesu sesbírat a zanalyzovat požadavky na zmíněné API a administrační rozhraní, provést návrh řešení, aplikaci implementovat, otestovat a následně provést nasazení a údržbu celé aplikace do produkčního prostředí.

Kapitola 2
Stávající systém TERESA

Systém TERESA se skládá z několika hlavních komponent. První komponentou, která provádí sběr dat o aktivitě pacienta, je tzv. smart wearable zařízení. To je propojeno s mobilním telefonem, který obsahuje dvě aplikace, a to aplikace Gadgetbridge a Konektor. Mobilní telefon bude následně komunikovat s backend serverovou aplikací, která vznikne v rámci této práce. Sem bude zároveň i odesílat nasbíraná data od pacientů.

2.1 Chytré náramky

Systém momentálně plně podporuje jediný chytrý náramek, a to Xiaomi Mi Band 5 [8]. Ten poskytuje měření aktivity pomocí PPG optického senzoru srdeční frekvence, akcelerometr a gyroskop. Tento náramek byl pro pilotní studii zvolen primárně z důvodu velice dobrého poměru cena/funkcionalita, který umožňuje nasazení a provedení studie s širším množstvím sledovaných pacientů bez nutnosti vysoké počáteční investice.

Tento náramek umožňuje sledování základních dat jako počet kroků, tepová frekvence, kvalita spánku či monitorování aktivit. Náramek sice nedokáže měřit pokročilá data jako okysličení krve, v první pilotní studii byl nicméně za primární sledovanou metriku zvolen počet kroků, které pacient vykonal, a proto byl tento náramek vyhodnocen jako plně dostačující. Data o počtu kroků, tepové frekvenci a intenzitě aktivity jsou poskytována v minutových intervalech.

Na projektu zároveň probíhá výběr a integrace náramků od dalších výrobců. Dalším zvoleným náramkem je Garmin VivoSport [9], jehož integrace do systému je ve finální fázi. Tento dražší náramek již umožňuje přesnější měření a sběr dalších dat, např. okysličení krve.

2.2 Gadget Bridge

Gadget Bridge [10] je aplikace pro OS Android, která umožňuje použití řady nositelných zařízení jako Mi Band, Amazfit Bip, HPlus a Pebble bez nutnosti použití closed-source aplikace výrobců a získávání dat skrze proprietární cloudové API.

Gadget Bridge v systému TERESA dovoluje propojit náramky Xiaomi Mi Band 5 přímo s poskytnutým mobilním telefonem, který pacienti účastnící se studie dostávají po vyplnění vstupního dotazníku a před začátkem měření. Gadget Bridge data z náramku pravidelně stahuje a ukládá do lokální databáze SQLite na zvolené místo v uložišti mobilního telefonu, odkud je následně načítá dále zmíněná mobilní aplikace Konektor a odesílá ke zpracování na backend.

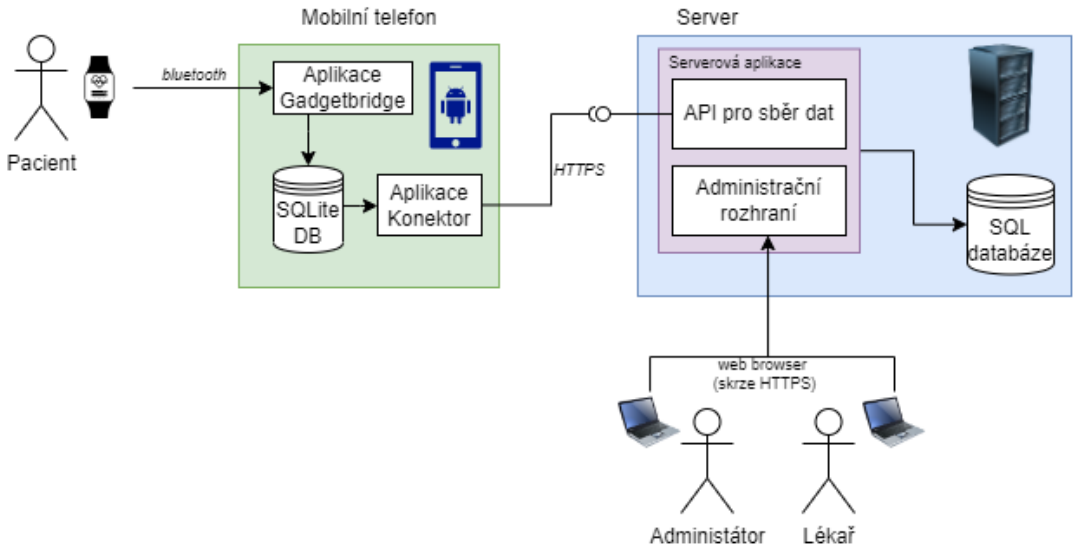
2.3 Mobilní aplikace Konektor

Pacienti účastnící se studie mají na přiděleném mobilním telefonu s operačním systémem Android k dispozici aplikaci s názvem Konektor. Tato aplikace slouží primárně

pro komunikaci s pacientem, například ve formě mobilních notifikací ohledně úspěšného či neúspěšného splnění stanoveného cíle počtu kroků. Dále také umožňuje vyplnit každodenní dotazník o stavu a průběhu cvičení. Tato aplikace zároveň získává data z lokální SQLite databáze, kde jsou uložena data z náramků synchronizovaná skrze Gadgetbridge. Apikace Konektor tato data načítá a během synchronizace následně odesílá na endpoint backend serveru, kde dojde k jejich finálnímu zpracování a uložení do databáze systému pro pozdější generování exportů.

Obrázek 2.1. Ukázka hlavní obrazovky mobilní aplikace Konektor

Schéma celého systému po implementaci administračního rozhraní a API pro sběr dat lze vidět na obrázku 2.2. Modře označená část schématu značí backend, který vznikl v rámci této práce.



Obrázek 2.2. Technické schéma systému TERESA [1]

Kapitola 3
Analýza a návrh

V této kapitole bude vysvětlen proces sběru požadavků v agilním prostředí projektu TERESA. Dále jsou zde sepsány požadavky na jednotlivé celky systému jako administrační rozhraní a API pro sběr dat. Z těchto požadavků následně vycházejí případy užití pro jednotlivé aktéry v systému.

3.1 Proces sběru požadavků

Projekt TERESA je veden v agilním stylu ve velice blízké spolupráci vývojářů se stakeholdery (zainteresovanými stranami). Jedná se o stakeholdery z Českého vysokého učení technického v Praze (ČVUT), Fakultní nemocnice Hradec Králové (FN HK), Univerzity obrany (UNOB) a Univerzity Palackého v Olomouci (UPOL). Veškeré požadavky byly sbírány a konzultovány na pravidelných týdenních schůzkách s uživateli systému. Primárně jde o administrátory a lékaře, jež mají studii a zúčastněné pacienty na starost. Těm byla následně průběžně demonstrována vyvinutá funkcionalita, na které se dále pracovalo po vzoru agilního iterativního vývoje.

3.1.1 Metodika Agile

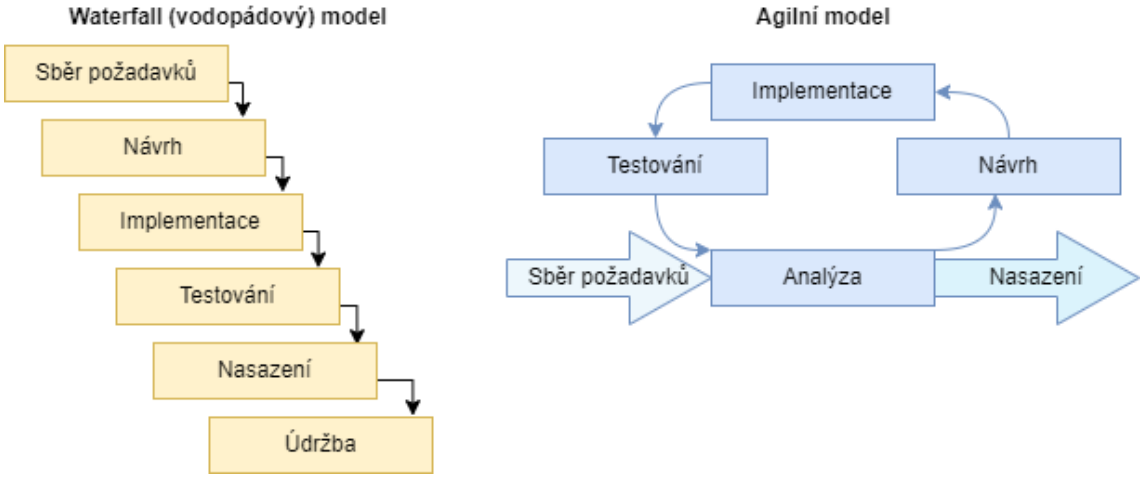
Agile [11] je iterativní přístup pro řízení zejména softwarových projektů. Tento přístup byl popularizován v roce 2001 publikací tzv. Manifesto for Agile Software Development, na kterém se podílelo sedmnáct softwarových vývojářů, mezi které patří např. Martin Fowler či Robert C. Martin. Agile vznikl za účelem odstranění problémů, se kterými se potýkají ostatní tradiční metodiky jako např. waterfall.

Waterfall (vodopádová) metodika používá sekvenční model pro vývojový cyklus, kde je projekt rozdělen na oddělené fáze jako analýzu, návrh, vývoj a testování. Tyto fáze na sebe sekvenčně navazují. Zatímco waterfall metodika je vhodná pro malé projekty a projekty, kde jsou již dopředu dobře známy požadavky, ukazuje se být méně efektivní v prostředích, ve kterých se požadavky často mění. Jakákoliv pozdější změna v návrhu systému, která již nebyla původně zpracována ve fázi analýzy či návrhu, může být velice drahá na implementaci. Rovněž díky oddělené fázi testování na konci projektu může být nákladné i opravování odhalených defektů.

Agilní metodiky tento problém řeší využitím inkrementálního přístupu. Celý vývojový cyklus analýzy, návrhu, implementace a testování probíhá zpravidla ve velice krátkých cyklech (např. 1-2 týdny), které se obvykle nazývají sprinty. V první iteraci zpravidla dojde k doručení takzvaného minimal viable product (MVP). Tento základ může být následně prezentován a konzultován se zákazníkem, aby v následujících iteracích docházelo k rozvíjení funkcionality až do požadovaného finálního stavu. Tento přístup dovoluje včas odhalit problémy v návrhu a je mnohem flexibilnější metodou pro vývoj softwaru.

Agile se stal základem pro množství frameworků pro řízení softwarových projektů, jako je například Scrum nebo Kanban.

Jednoduché porovnání agilní metodiky oproti tradiční waterfall metodice lze vidět na obrázku 3.1.



Obrázek 3.1. Porovnání Agile a waterfall metodik [12]

Vzhledem k často se měnícím požadavkům na systém, což je ve vývoji pro zdravotnickou sféru obvyklé, a možnosti úzce spolupracovat s uživateli aplikace, kteří pro nás představují pomyslného koncového zákazníka, je zde agilní přístup k vývoji považován za vhodnější.

3.2 Data zpracovávaná při správě zařízení a pacientů

Pro možnost snadného nastavení a přehledu nad zařízeními a pacienty účastnících se studie vznikne jako součást systému administrační rozhraní. To bude dovolovat administrátorům a lékařům dle potřeby registrovat nová zařízení do systému, upravovat jejich nastavení či jim přiřazovat pacienty a nastavovat jejich preference či další různé hodnoty.

3.2.1 Data zpracovávaná při správě zařízení

Jednou z hlavních součástí administračního rozhraní systému TERESA je správa připojených zařízení, ze kterých systém přijímá data. V době psaní této práce byl pro systém vybrán jeden typ zařízení, který bude použit v produkci, a to chytrý náramek Xiaomi Mi Band 5. Tento náramek je na straně pacienta spárován s mobilní aplikací, která data z náramku načítá a následně je odesílá k uložení do backendu systému. Administrátor tato zařízení může do systému přidávat a existující zařízení upravovat.

Administrátor má možnost u zařízení upravovat následující parametry:

- **MAC adresa náramku**
  řetězec generovaný aplikací Gadgedbridge na straně klienta, určen pro identifikaci náramku během přenosu dat
- **ID zařízení**
  řetězec generovaný aplikací Gadgedbridge identifikující mobilní zařízení. Je určen pro identifikaci zařízení během přenosu dat v kombinaci s MAC adresou. Kombinace těchto dvou identifikátorů byla zvolena z důvodu, že po vyresetování zařízení do továrního nastavení dojde k vygenerování nové MAC adresy.
- **Nickname zařízení**
  lidsky čitelný řetězec pro snadnější vyhledání náramku v administraci

- **Evidenční číslo náramku**
  číslo přiřazené náramku pro evidenci zakoupeného hardware
- **Allowed checkbox**
  pro určení, zda je toto zařízení aktivně používáno v rámci běžící studie
- **Banned checkbox**
  pro možnost zablokování příjmu dat ze zařízení

3.2.2 Data zpracovávaná při správě pacientů

Ke každému zařízení může být přiřazen pacient, který se účastní studie. V systému TERESA rozlišujeme dva typy pacientů: monitorovací a intervenční.

Monitorovací pacient je součástí kontrolní skupiny ve studiích, které systém TERESA využívají. Tento pacient nemá průběžně nastavované cílové počty kroků ani o nich nedostává notifikace v mobilní aplikaci. Zároveň dostává pouze omezenou sadu otázek jako součást pravidelných dotazníků.

Intervenční pacient je pacient, na kterém je pozorován vliv telecoachingu při rehabilitaci po plicním onemocnění. Tento pacient má již naplánovanou aktivitu ve formě počtu kroků, které každý den plní, a tento počet je průběžně navyšován. V případě, kdy tento počet nedokáže za den splnit, je mu o tomto faktu zobrazena v mobilní aplikaci notifikace a zároveň se součástí každodenního dotazníku stane otázka, z jakého důvodu se cílový počet kroků pacientovi nepodařilo splnit.

Každý pacient účastnící se studie je v prvním týdnu součástí pouze monitorovací skupiny, tzn. nemá nastaveny cílové počty kroků a neobdržuje notifikace. Tento týden slouží pro zhodnocení pohybové aktivity každého individuálního pacienta a získání základních hodnot pro plánování dalších aktivit.

Stejně jako u nastavení zařízení může administrátor či lékař přidávat či upravovat následující parametry:

- **Datum začátku měření**
  datum, od kdy se pacient účastní studie a od kdy jsou očekávána data z měření aktivity
- **Checkbox pro intervenčního pacienta**
  nastavení určující, zda je pacient zařazen do intervenční či monitorovací skupiny pacientů
- **Datum začátku intervence**
  datum, od kdy se pacient řadí do tzv. intervenční skupiny, tzn. od kdy začne dostávat notifikace o počtu kroků a od kdy začíná vyplňovat dotazníky
- **Cílový počet kroků**
  nastavení pro cílový počet kroků, které by měl pacient denně vykonat. Jedná se o seznam hodnot, kde vedle cílového počtu kroků lze nastavit datum, od kdy tento limit platí. Je tak možné tyto hodnoty skrze administrační rozhraní naplánovat dopředu
- **Informace o pacientovi**
  věk, váha, výška, pohlaví. Momentálně pouze pro informační účely, s vyplněnými hodnotami systém nijak nepracuje
- **Checkbox pro povolení notifikací**
  nastavení na straně serveru, zda má pacient získávat notifikace z mobilní aplikace, např. notifikace o počtu kroků a notifikace pro vyplnění dotazníku
- **Časy notifikací**
  nastavení jednotlivých časů, ve kterých se intervenčním pacientům zobrazují notifikace. Je zde možné nastavit tři různé časy: ranní notifikaci (pro informaci o stanove-

ném počtu kroků pro daný den), odpolední notifikaci (o průběhu při plnění denního cíle) a večerní notifikaci (připomenutí vyplnění každodenního dotazníku)

Veškerá tato data jsou poskytována mobilní aplikaci skrze zabezpečené API.

3.3 API pro sběr dat

Veškerá data o aktivitě pacientů jsou sbírána ze zařízení na uživatelské straně skrze mobilní aplikaci Gadgetbridge, která slouží jako alternativa oproti synchronizaci náramků s Xiaomi Cloud a stahování dat odtud. Namísto toho aplikace Gadgetbridge uloží sesbíraná data do lokální databáze v mobilním telefonu. Odtud si je následně načítá mobilní aplikace Konektor, která data z lokální databáze poté synchronizuje na server. Pro tento účel bylo nutné vytvořit na straně backendu API (application programming interface).

Hlavním požadavkem ze strany aplikace Konektor bylo vytvoření endpointu, který bude přijímat HTTPS POST požadavky, které obsahují identifikační informace o zařízení, ze kterého data pochází, a samotné naměřené hodnoty. Tento endpoint musí být chráněn proti neoprávněnému přístupu alespoň jednoduchou formou zabezpečení, kterou bude sdílený klíč. Musí také respektovat nastavení uvedené v požadavcích na správu zařízení, tj. možnost zablokovat požadavky z daného zařízení.

Konkrétní zpracovávaná data budou diskutována v kapitole o implementaci.

3.4 Dotazníky pro pacienty

V pilotní studii bylo nutné s pacienty velice často komunikovat a při jakékoliv odchylce pacienta kontaktovat a zjistit, co způsobilo například nesplnění stanoveného cílového množství kroků za den. Toto vyžadovalo mnoho času ze strany organizátorů studie, a tak vznikl požadavek do systému implementovat funkcionalitu dotazníků, které dovolí velkou část komunikace s pacienty zjednodušit a urychlit.

Aplikace podporuje několik druhů otázek. První je **otázka s otevřenou odpovědí**. Jedná se o jednoduchou otázku, ke které může uživatel napsat jakoukoliv odpověď do textové oblasti v mobilní aplikaci. Jelikož ale tato forma není vždy uživatelsky nejpřívětivější a zpracování dat z ní vyžaduje pohled lékaře, preferují se ve většině případů následující dvě varianty otázek.

Druhou formou otázky je **otázka s výběrem několika možností** z připraveného seznamu. Uživatel v aplikaci u takové otázky pak vybere 0 až N různých odpovědí. Pokud mu žádná z odpovědí nevyhovuje, může zároveň vyplnit i krátký textový komentář. Pro lepší uživatelskou přívětivost lze ke každé možnosti přidat emoji, které uživateli mobilní aplikace usnadní orientaci ve výběru. Příkladem je otázka “Jaké pohybové aktivity jste dnes vykonával(a)?” se seznamem různých aktivit a sportů, doplněných o emoji.

Poslední podporovanou formou je **otázka s výběrem hodnocení v rozsahu 1 až 5**. Toto je nejčastěji používaný typ otázek a je použit na otázky typu “Jak silné jste dnes měl(a) dechové obtíže”, “Jak se vám spalo minulou noc?” a podobně. V administračním rozhraní je zároveň možné nastavit individuální popisky pro minimální i maximální hodnotu

Otázky rozlišujeme na dvě hlavní oblasti, a to **denní** a **výjimečné**. Denní otázky patří do dotazníků, které pacientovi přicházejí každý den. Jedná se hlavně o zhodnocení pacientova průběhu dne a denního cvičení, souhrn prováděných aktivit a podobně. Výjimečné otázky jsou zobrazovány pouze ve výjimečných situacích. V současné době je jedinou takovou příležitostí stav, kdy pacient nedokáže splnit nastavené cílové množství kroků a mobilní aplikace se jej zeptá, z jakého důvodu tento cíl nesplnil.

Každou z vytvořených otázek je možné libovolně aktivovat či deaktivovat. U každé otázky lze rovněž nastavit, zda se má zobrazovat všem pacientům, či pouze intervenčním. Například lze nastavit, aby monitorovací skupině nebyly zobrazovány výjimečné otázky v případě nesplněného cílového množství kroků. To dovoluje lepší rozdělení monitorovací a intervenční skupiny pro studie.

Otázky je možné v případě potřeby libovolně upravovat. V případě upravování otázky, která už však má vyplněné odpovědi od pacientů, dojde k zobrazení varovné hlášky o možnosti vytvoření nekonzistence v datech. Zároveň otázky, na které ještě nebyla vyplněna žádná odpověď, je možné ze systému smazat.

Aplikace dále poskytuje API, díky kterému lze seznam otázek načíst do mobilní aplikace, a skrze které lze přijmout vyplněné odpovědi od pacientů.

3.5 Export dat

Data pacientů v systému jsou hlavně využita pro účely studií a jsou tak dále zpracovávána mimo administrační prostředí TERESA. Z tohoto důvodu byl jako primární možnost zobrazování dat zvolen export do CSV a XLS souborů. Tyto soubory jsou následně sdíleny mezi lékaře, kteří nad nimi provádějí další analýzy.

Administrační rozhraní nabízí dva různé druhy exportů. Prvním je export nezpracovaných dat do CSV souboru. Aplikace pro zvolená zařízení vygeneruje neagregovaný obsah databáze pro zvolená zařízení. Takový export je vhodný například pro účely analýzy přesnosti měření v porovnání s lékařskými pedometry.

Druhým exportem je tabulka obsahující souhrn dat ve formátu XLSX (formát rodiny Extensible Stylesheet Language) určena pro zobrazení v tabulkovém procesoru Microsoft Excel. Tento export již slouží pro běžnou lékařskou práci a slouží jako podklad pro komunikaci s pacienty.

Tabulka dále obsahuje souhrn intenzity aktivit pro každý den. Jde o počty minut, které jsou agregované dle intenzity dané následujícími pravidly:

- **Nízká aktivita** - počet minut, ve kterých pacient vykonal méně než 80 kroků
- **Střední aktivita** - počet minut, ve kterých pacient vykonal mezi 80 až 110 kroky
- **Vysoká aktivita** - počet minut, ve kterých pacient vykonal nad 110 kroků

Dalším požadavkem na obsah tabulky jsou sloupcové grafy zobrazující výše zmíněné hodnoty počtu kroků a intenzity pohybu.

Posledním prvkem jsou odpovědi na otázky z dotazníku. U těch je vždy uvedeno datum, pro které byla otázka vyplňována, a její vyplněná hodnota s případným komentářem.

Pro každý z těchto exportů lze také nastavit, z jakých zařízení a z jakého časového rozsahu bude obsahovat data. V případě exportu do tabulky jsou data z více zařízení oddělena do různých listů XLSX souboru.

3.6 Funkční požadavky

- **FRQ01 - Správa zařízení**

  Aplikace umožní administrátorům přidávat a upravovat zařízení, na kterých probíhá sběr dat od pacientů. Veškerá zařízení budou zároveň zobrazena v seznamu pro snadný přehled. Aplikace bude dovolovat určitá zařízení zablokovat či je vyřadit z aktivních, což zamezí nežádoucímu příjmu dat z těchto zařízení. Ta budou vizuálně oddělena od ostatních.

- **FRQ02 - Správa pacientů**

  Aplikace umožní lékařům zobrazovat a spravovat údaje o pacientech, kteří jsou přiřazeni k zařízením. Mimo jiné dovolí nastavit denní cíle počtu kroků, které má pacient vykonat. Zároveň umožní na serverové straně nastavit preference pacienta, jako jsou např. časy, kdy pacient dostává mobilní notifikace. Ti budou také rozděleni na monitorovací a intervenční skupiny.
- **FRQ03 - Nastavení dotazníků a otázek**

  Aplikace umožní lékařům spravovat dotazníky, které následně budou pacienti vyplňovat skrze mobilní aplikaci. Aplikace bude podporovat několik typů otázek, jako je otevřená otázka, otázka s výběrem z možností či výběr na škále 1-5. Dotazníky budou rozděleny na dva druhy, a to každodenní a vyjímečné. Otázky bude možné libovolně aktivovat či deaktivovat.
- **FRQ04 - Export sesbíraných dat**

  Aplikace umožní lékařům exportovat data ve formě kompletních exportů dat ve formátu CSV a XSLS tabulek s agregovanými daty a grafy. Současně dovolí omezit, z jakých zařízení a za jaké časové období má být export vygenerován.
- **FRQ05 - API pro sběr dat a komunikaci s mobilní aplikací**

  Aplikace bude podporovat propojení s mobilní aplikací, kterou využívají pacienti, skrze webové rozhraní. Rozhraní bude podporovat příjem dat o aktivitě pacientů, poskytovat jejich údaje a přijímat odpovědi na dotazníky.

3.7 Nefunkční požadavky

- **NFRQ01 - Zabezpečení**

  Vzhledem k přenosu potencionálně citlivých dat bude aplikace komunikovat v zašifrované podobě pomocí protokolu HTTPS. Uživatelé administračního rozhraní (lékaři, administrátoři) budou autentikováni pomocí kombinace e-mailu a hesla. Příchozí komunikace z mobilních aplikací pacientů bude autentikována sdíleným tajemstvím. Mobilní aplikace nebude moci získat údaje jiného pacienta.
- **NFRQ02 - Rozšiřitelnost**

  Aplikace bude navržena tak, aby bylo možné rozšířit existující funkcionalitu či případně do systému TERESA přidávat další nové moduly.
- **NFRQ03 - Zálohování**

  Veškerá data uložená do databáze v produkčním prostředí budou zálohována nejméně na dvě různá offsite uložiště a to minimálně dvakrát za den v časech 12:00 a 00:00. Celý virtuální stroj, ve kterém je nasazeno produkční prostředí, bude rovněž každý týden kompletně zálohován.
- **NFRQ04 - Doba odezvy**

  Aplikace bude schopná zpracovat 95 % požadavků do 4 vteřin. Nejdelší možná doba odezvy je 20 vteřin. Tyto hodnoty bude aplikace schopna zajistit pro 50 simultálních uživatelů. Pozn.: Očekává se, že největší zátěž bude probíhat na konci posledního dne každého monitorovacího týdne, tedy zpravidla v neděli mezi 18:00 až 23:00, kdy pacienti manuálně synchronizují svá data.
- **NFRQ05 - Integrita dat**

  Přijatá data budou ukládána do databáze způsobem, aby nevznikla datová nekonzistence či nedošlo ke ztrátě dat. Uživatelské akce, které mohou vést k vytvoření nekonzistence skrze úpravy v administračním rozhraní, budou omezené nebo budou obsahovat varování na tuto skutečnost.

3.8 Případy užití

Na základě konzultací byly identifikovány tři typy aktérů, kteří budou interagovat se systémem TERESA, primárně s jeho backendem. Prvním aktérem je pacient zaregistrovaný do systému, respektive jemu přiřazené zařízení a mobilní aplikace, která s backendem systému komunikuje a odesílá sesbíraná data. Dalším aktérem je lékař, který pracuje s administračním rozhraním za účelem generování exportů dat, nastavování údajů o pacientech, vytváření dotazníků apod. Posledním aktérem je administrátor, který přejímá oprávnění od lékařů a rozšiřuje je o technické záležitosti, jako je například zaregistrování zařízení do systému.

3.8.1 Případy užití pro pacienta (mobilní aplikace, koncové zařízení)

- **UC01 - Získat seznam otázek**
  1. Pacientova mobilní aplikace odešle požadavek na API pro získání seznamu otázek pro dotazník
  2. Systém odešle odpověď se seznamem otázek pro daného pacienta
- **UC02 - Odeslat odpovědi na dotazník**
  1. Pacient na mobilním zařízení vyplní odpovědi na dotazník
  2. Pacientova mobilní aplikace v rámci synchronizace odešle vyplněné otázky
  3. Systém ověří správnost nastavení zdrojového zařízení
  4. Systém uloží odpovědi do databáze a přiřadí je k zdrojovému pacientovi
  5. Systém potvrdí úspěšný příjem odpovědí
- **UC03 - Odeslat sesbíraná data**
  1. Pacient spustí synchronizaci v mobilní aplikaci
  2. Pacientova mobilní aplikace odešle backend serveru data nasbíraná z chytrého náramku
  3. Systém autentikuje zařízení a ověří, zda z něj není zakázán příjem dat
  4. Systém uloží přijatá data do databáze a aktualizuje čas poslední synchronizace pro dané zařízení
  5. Systém potvrdí úspěšný příjem dat
- **UC04 - Získat nastavení a údaje o zařízení/pacientovi**
  1. Pacientova mobilní aplikace odešle požadavek na API pro získání svých údajů a nastavení
  2. Systém odešle odpověď s údaji a nastavením o pacientovi přiřazenému zdrojovému zařízení

3.8.2 Případy užití pro lékaře

- **UC05 - Editovat údaje o pacientovi**
  1. Lékař přihlášený do systému zvolí pacienta pro editaci
  2. Systém zobrazí formulář s předvyplněným současným nastavením pacienta
  3. Lékař upraví požadované hodnoty a potvrdí je
  4. Systém zvaliduje a upraví požadované hodnoty v databázi
  5. Systém potvrdí úspěšnost úprav

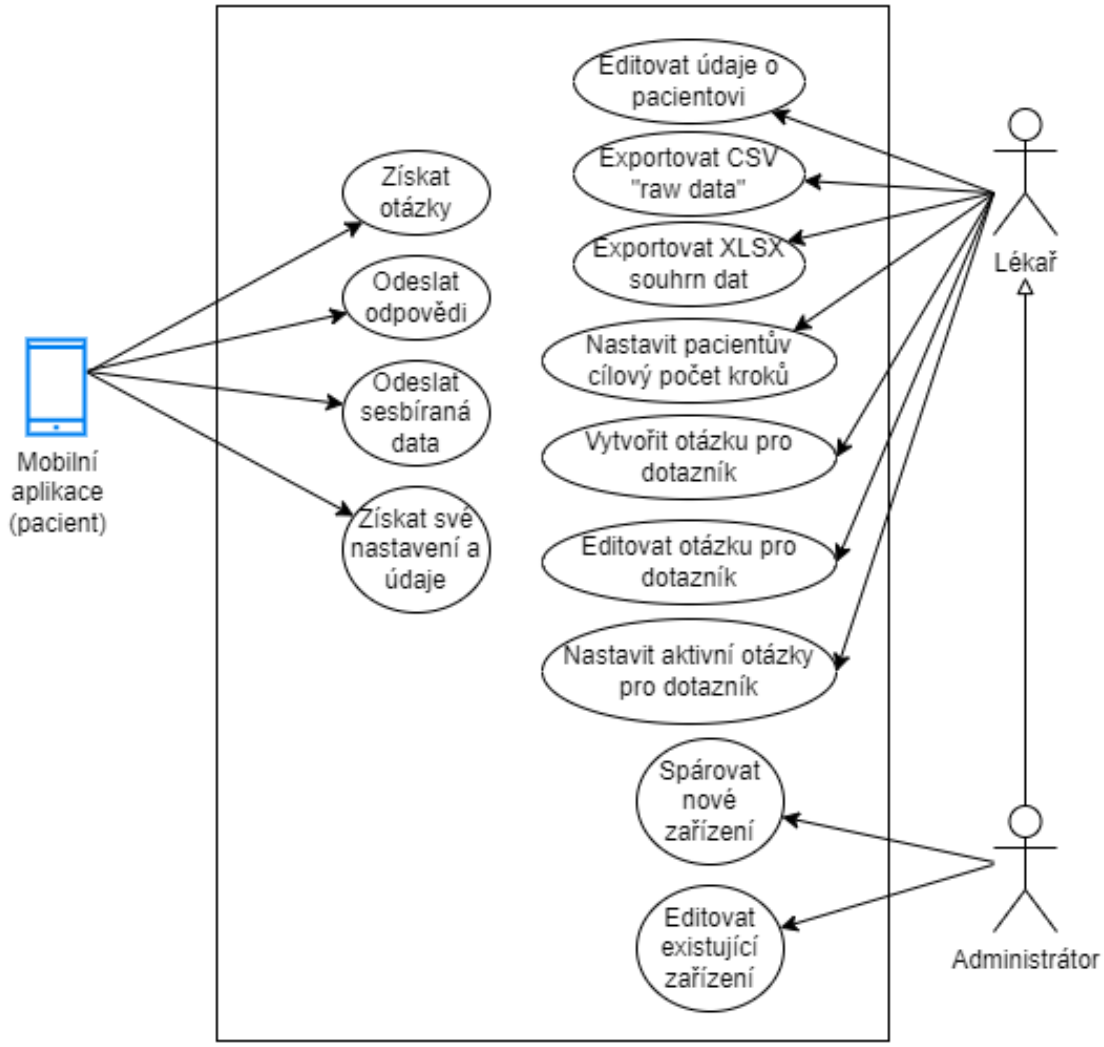
- **UC06 - Vytvořit CSV s kompletním datovým exportem**
  1. Lékař přihlášený do systému zvolí seznam zařízení, pro která chce vytvořit export
  2. Lékař zvolí časový rozsah pro data, která mají být obsažena v exportu
  3. Systém pro vybraný časový rozsah a seznam zařízení vygeneruje kompletní export dat ve formátu CSV
  4. Systém odešle výsledný CSV soubor lékaři
  5. Lékař uloží CSV soubor na svůj lokální disk
- **UC07 - Exportovat XLSX tabulku se souhrnem dat**
  1. Lékař přihlášený do systému zvolí seznam zařízení, pro která chce vytvořit souhrn
  2. Lékař zvolí časový rozsah pro data, která mají být obsažena v souhrnu
  3. Systém pro vybraný časový rozsah a seznam zařízení vygeneruje tabulku formátu XLSX spolu s agregovanými daty, grafy a odpověďmi na dotazníky
  4. Systém odešle výslednou XLSX tabulku lékaři
  5. Lékař uloží XLSX tabulku na svůj lokální disk
- **UC08 - Nastavit pacientův cílový počet kroků**
  1. Lékař přihlášený do systému zvolí pacienta pro úpravu
  2. Systém zobrazí formulář s předvyplněným současným nastavením pacienta
  3. Lékař přidá záznam do seznamu cílových počtů kroků
  4. Lékař vyplní cílový počet kroků a datum, od kterého tento počet platí
  5. Systém uloží nový seznam záznamů o cílovém počtu kroků
- **UC09 - Vytvořit otázku pro dotazník**
  1. Lékař přihlášený do systému chce vytvořit novou otázku
  2. Systém zobrazí formulář se základním nastavením otázky - titulek, možnost přidat poznámku, typ otázky, cílový dotazník
  3. Lékař vyplní požadované nastavení
  4. Systém zobrazí rozšířený formulář s dalším nastavením na základě typu otázky
  5. Lékař vyplní dodatečné nastavení a potvrdí vytvoření otázky
  6. Systém uloží otázku do databáze
- **UC10 - Editovat otázku pro dotazník**
  1. Lékař přihlášený do systému zvolí otázku pro editaci
  2. Systém zobrazí formulář s předvyplněným nastavením otázky
  3. Lékař upraví požadované údaje
  4. Systém uloží upravenou otázku do databáze
- **UC11 - Nastavit aktivní otázky pro dotazník**
  1. Lékař přihlášený do systému zobrazí seznam vytvořených otázek
  2. Lékař pro každou otázku zvolí, zda má být aktivní a zobrazovat se jako součást dotazníku
  3. Systém uloží nastavení do databáze

3.8.3 Případy užití pro administrátora systému

- **UC12 - Spárovat nové zařízení**
  1. Administrátor přihlášený do systému chce zaregistrovat nové zařízení
  2. Systém zobrazí formulář pro přidání zařízení
  3. Administrátor vyplní údaje o zařízení, které si přeje spárovat
  4. Systém zvaliduje vyplněné údaje a nové zařízení uloží do databáze

- **UC13 - Editovat existující zařízení**
  1. Administrátor přihlášený do systému vybere zařízení pro editaci
  2. Systém zobrazí formulář s předvyplněným nastavením zařízení
  3. Administrátor upraví požadované údaje o zařízení
  4. Systém zvaliduje vyplněné údaje a upravené zařízení uloží do databáze

Uvedené příklady užití jsou zobrazeny na UML use-case diagramu 3.2.



Obrázek 3.2. UML diagram případů užití

Kapitola 4
Implementace

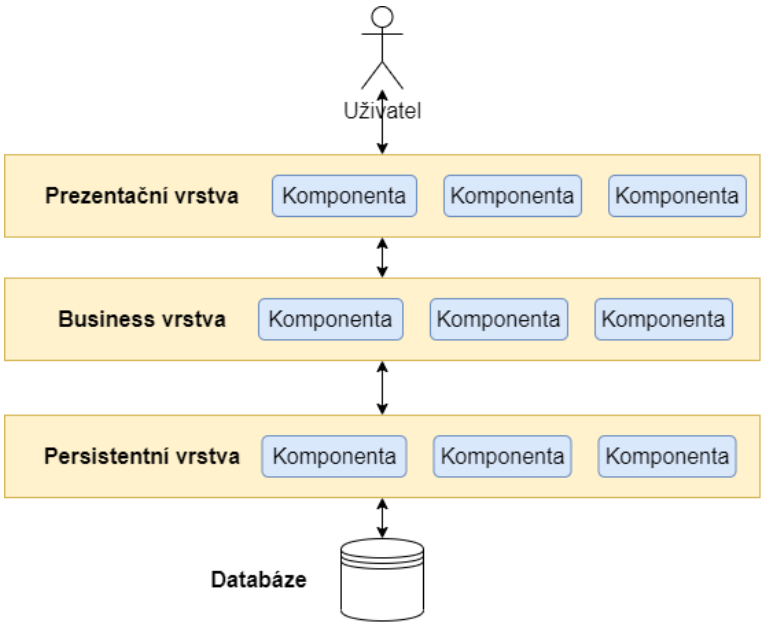
V této kapitole je popsán způsob, jakým byly jednotlivé části serverové aplikace implementovány, včetně použitých technologií a knihoven. Zároveň se zde nachází ukázky administračního rozhraní.

4.1 Architektura aplikace

Serverová aplikace používá *vícevrstvou architekturu* [13]. Jedná se o jeden z nejběžnějších architektonických stylů obzvláště pro malé a střední aplikace. Vícevrstvá architektura znamená rozdělení jednotlivých komponent systému do horizontálních vrstev, kde komponenty ve stejné vrstvě vykonávají podobnou funkcionalitu. Každá vrstva pak komunikuje zpravidla jen s vrstvami, se kterými sousedí.

Přestože vícevrstvá architektura explicitně neomezuje počet vrstev, do kterých má být projekt rozdělen, běžně jsou použity následující čtyři vrstvy: prezentační, business, persistentní vrstva a databáze. Prezentační vrstva se stará primárně o uživatelské rozhraní a jeho logiku. Komunikuje následně s business vrstvou, která obsahuje veškeré výpočty a business logiku. K tomu využívá persistentní vrstvu, jenž je zodpovědná za komunikaci s databází a obsahuje k tomu potřebné komponenty jako např. data access objekty (DAO). Výhodou takovéhoto rozdělení vrstev je tzv. *separation of concerns*, neboli oddělení odpovědnosti. Tato vlastnost umožňuje snazší vývoj, testování a údržbu aplikace díky omezené odpovědnosti jednotlivých komponent a dobře definovanému rozhraní mezi jednotlivými vrstvami.

Diagram zobrazující obecnou strukturu vícevrstvé aplikace lze vidět na obrázku 4.1.



Obrázek 4.1. Obecný diagram vícevrstvé architektury

Výhodou vícevrstvé architektury je relativně jednoduchá implementace a následné rozšiřování. Díky oddělení odpovědností v jednotlivých vrstvách je snadné jednotlivé vrstvy rozšiřovat a případně i práci snadněji rozdělit mezi více vývojářů či celých týmů. Omezením možnosti vrstev komunikovat pouze se sousedícími vrstvami také vzniká posílení bezpečnosti aplikace.

Nevýhodou této architektury je nízká škálovatelnost. Jejím výsledkem je monolitická aplikace, která nedovoluje horizontální škálování (tj. nasazení aplikace a jejích částí na více různých strojů) nebo jej dovoluje pouze v omezené míře. Škálování je tedy nutné provádět vertikálně, tedy přidáváním systémových prostředků, jako je CPU a RAM, do jednoho systému, což výrazně zvyšuje náklady na provoz systému. V případě, kdy se očekává vystavení aplikace vysoké zátěži a velkému množství simultálních uživatelů, vícevrstvá architektura nemusí být vhodná a může být vhodnější využít tzv. *architekturu microservices*, neboli mikroslužeb.

4.2 Datový model

Pro persistenci byla zvolena relační struktura dat, tudíž došlo k využití tradiční SQL databáze. Na začátku projektu po úvodní analýze tak vznikl základ pro relační datový model, který byl v průběhu dále rozšiřován. Pro snadnější orientaci je tato část textu rozdělena na dvě podčásti, a to na datový model pro zařízení, pacienty a sbíraná data, a dále na datový model týkající se dotazníků pro pacienty.

4.2.1 Datový model zařízení, pacientů a sbíraných údajů o aktivitě

Tato část pojednává o hlavních entitách pro správu zařízení, pacientů a dat, která jsou skrze zařízení sbírána. Jde tedy primárně o entitu `Device`, jemu odpovídající entitu `Patient`, dále pak entity `ActivitySummary`, `ActivityEntity` a `HeartRate`.

Entita ***Device*** představuje v systému koncové zařízení, neboli chytrý náramek propojený s mobilní aplikací, který pacient denně nosí a který provádí sběr dat o aktivitě. Obsahuje následující atributy:

- **hwId**
  MAC adresa náramku. Ve spojení s `deviceId` slouží jako unikátní identifikátor zařízení.
- **deviceId**
  Identifikační řetězec mobilního telefonu, se kterým je náramek spárován. Ve spojení s `hwId` slouží jako unikátní identifikátor koncového zařízení.
- **nick**
  Lidsky čitelný název pro konkrétní zařízení či přiřazeného pacienta pro snadnou orientaci v seznamu všech zařízení.
- **evidenceId**
  Číslo přiřazené náramku pro evidenci zakoupeného hardware.
- **lastSyncTime**
  Čas poslední synchronizace. Slouží administrátorům a lékařům pro informaci, kdy pacient naposledy provedl synchronizaci svých dat z mobilní aplikace. V případě dlouhé prodlevy od poslední synchronizace může být poté pacient kontaktován.

Jakmile je zařízení předáno pacientovi, vzniká v systému 1:1 vazba `Device` s `Patient` entitou. Vazba 1:1 je použita z důvodu budoucí možnosti provést úplně oddělení dat zařízení a pacientů a možnosti přiřazovat více náramků k jednomu pacientovi v různých časových obdobích. Díky nedostatkům použitých náramků a aplikace Gadgetbridge,

kdy se při resetu náramku do továrního nastavení změní MAC adresa zařízení a tudíž je nutné zařízení do administračního rozhraní znovu registrovat, však tato změna zatím nebyla žádoucí a její priorita byla snížena. Entita ***Patient*** obsahuje následující atributy:

- **interventional**
  Boolovská hodnota označující, zda je pacient přiřazen do intervenční (`true`) či monitorovací (`false`) skupiny.
- **age**, **weight**, **height**, **gender**
  Údaje o pacientovi jako věk, výška, váha a pohlaví. V systému mají současně pouze informační hodnotu a nejsou využity pro další výpočty.
- **startDate**
  Datum, kdy pacient převzal zařízení, byl zařazen do studie a začal se sběrem dat o aktivitě.
- **interventionStartDate**
  Datum, od kterého pacientovi končí první monitorovací týden a případně začíná přesun do intervenční skupiny (v závislosti na volbě `interventional` pole).
- **notificationsEnabled**
  Boolovská hodnota označující, zda má pacient dostávat notifikace na mobilním zařízení ohledně denních cílů či upozornění na vyplnění dotazníku
- **notificationMorningTime**, **notificationAfternoonTime**, **surveyTime**
  Časy, kdy jsou pacientovi v mobilní aplikaci zobrazeny notifikace informující o počtu kroků stanovených na daný den, o průběhu při splňování cíle pro daný den a připomenutí vyplnění každodenního dotazníku. Tyto atributy jsou volitelné a mohou být ponechány prázdné. V takovém případě je pacientovi dovoleno toto nastavení provést v mobilní aplikaci.
- **lastUpdateTime**
  Čas poslední úpravy pacientových údajů skrze administrační rozhraní.

Díky takto navrženému datovému modelu systém nepracuje s citlivými údaji, pomocí kterých by bylo možné přímo identifikovat identitu pacienta. Systém TERESA tedy pracuje jen s anonymizovanými daty, což snižuje nebezpečí úniku osobních dat. Přímé propojení identity pacienta s daty v systému provádí pouze lékaři, kteří s konkrétními pacienty komunikují.

Pro entitu `Patient` existuje 1:N vazba s entitami ***StepsTarget*** s následujícími atributy:

- **stepsTarget**
  Číslo určující cílový počet kroků
- **dateFrom**
  Datum, od kterého je tento cíl platný

Vazba 1:N oproti variantě, kdy by cílový počet kroků byl pouze atributem na entitě `Patient`, zde dovoluje lékaři dopředu naplánovat cílové počty kroků pacienta na několik týdnů či měsíců dopředu s libovolnou granularitou. Zároveň tato funkcionalita dovoluje uchovávat historii hodnot pro sledování, jakým způsobem se pacientova aktivita navyšovala.

Entitou, která obsahuje hlavní informace o aktivitě a počtu kroků, je entita ***Activity***. Do té jsou ukládána data naměřená na chytrých zařízení. Tato data jsou rozdělena s minutovou granularitou a obsahují následující atributy:

- **datetime**
  Datum a čas měření.

- **steps**

  Počet kroků vykonaných v dané minutě.

- **kind**

  Hodnota z proprietárního číselníku aplikace Gadgetbridge určující typ vykonávané aktivity.

- **intensity**

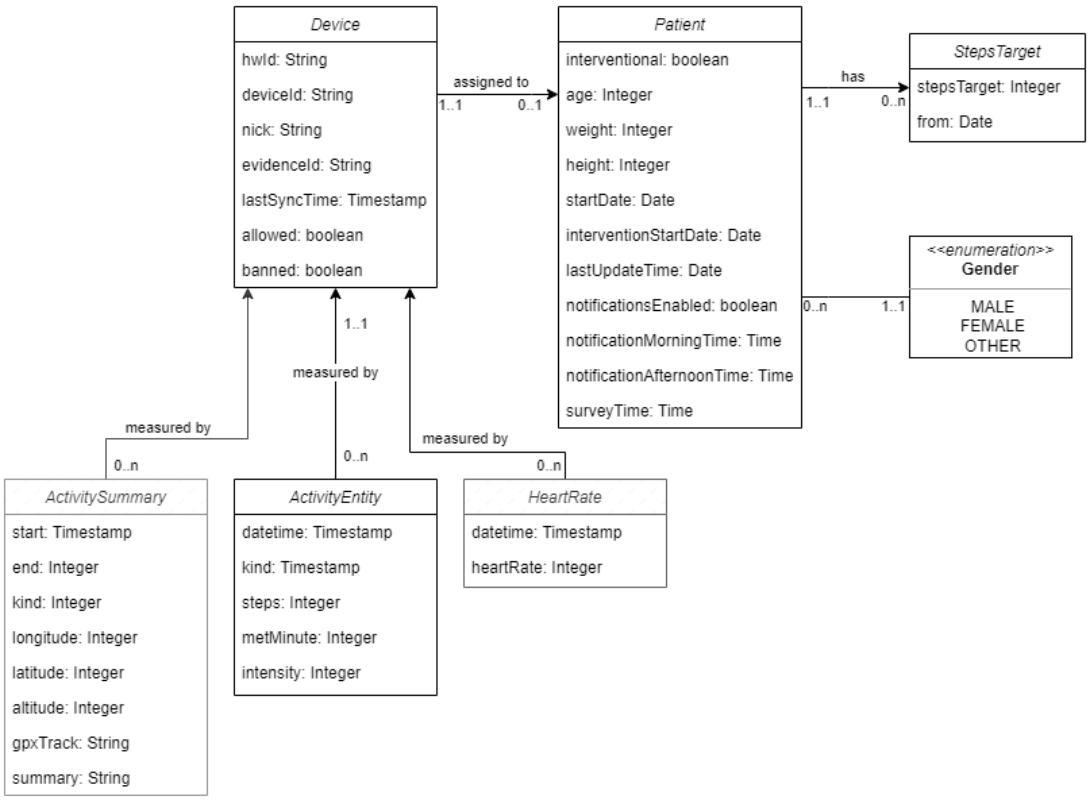
  Hodnota určující intenzitu pohybové aktivity na škále 0-255.

- **metMinute**

  Hodnota aktivity přepočtená na jednotky MET [14] (Metabolic Equivalent of Task).

Další entity obalující data o souhrnu aktivit `ActivitySummary` a srdeční frekvenci `HeartRate` nemají v současném stavu v systému využití pro generování exportů či pro jiné výpočty, proto nebudou detailně popsány. Jsou však přijímány z koncových zařízení a uchovávány v databázi pro případ využití v dalších možných funkcionalitách systému TERESA.

Popsaný datový model zařízení, pacientů a sbíraných údajů o aktivitě je popsán na diagramu 4.2.



Obrázek 4.2. UML class diagram datového modelu zařízení, pacientů a sbíraných dat

4.2.2 Datový model dotazníku pro pacienty

Hlavním uzlem datového modelu je entita ***Question***, která obsahuje veškeré informace o vytvořené otázce. Její součástí jsou dvě enumerace - `QuestionType` a `SurveyType`. `QuestionType` značí, o který z uvedených typů otázky se jedná a jak má být otázka tedy reprezentována. `SurveyType` naopak říká, zda má být otázka součástí běžných každodenních dotazníků, nebo zda se zobrazuje pouze při výjimečných situacích. Dále obsahuje tyto atributy:

- **label**

  Popisek otázky, zpravidla samotný text dotazu
- **allowNote**

  Booleovská hodnota, která značí, zda má být vedle obvyklé odpovědi možné přidat ještě vlastní komentář
- **active**

  Booleovská hodnota značící, zda je otázka aktivní. Neaktivní otázky nejsou při vyplňování dotazníku v mobilní aplikaci uživateli zobrazeny.

Entita `Question` obsahuje dále seznam možností přiřazených otázce, tedy entit ***Option***. V závislosti na typu otázky může být tento seznam prázdný (v případě otevřené otázky), obsahovat dvě hodnoty pro minimum a maximum (v případě otázky s výběrem hodnocení) či obsahovat libovolné množství možností pro otázky typu multiple choice. Mezi entitami `Question` a `Option` tak vzniká vazba 1:N. Entita `Option` obsahuje následující atributy:

- **label**

  Popisek dané možnosti.
- **order**

  Číslo značící pořadí této možnosti v jejich seznamu.
- **emoji**

  UTF-8 emoji, které je zobrazeno v případě otázek s výběrem z několika možností. Slouží ke snadnější orientaci v seznamu možností.

Odpovědi jsou ukládány v entitě ***Answer*** s následujícími atributy:

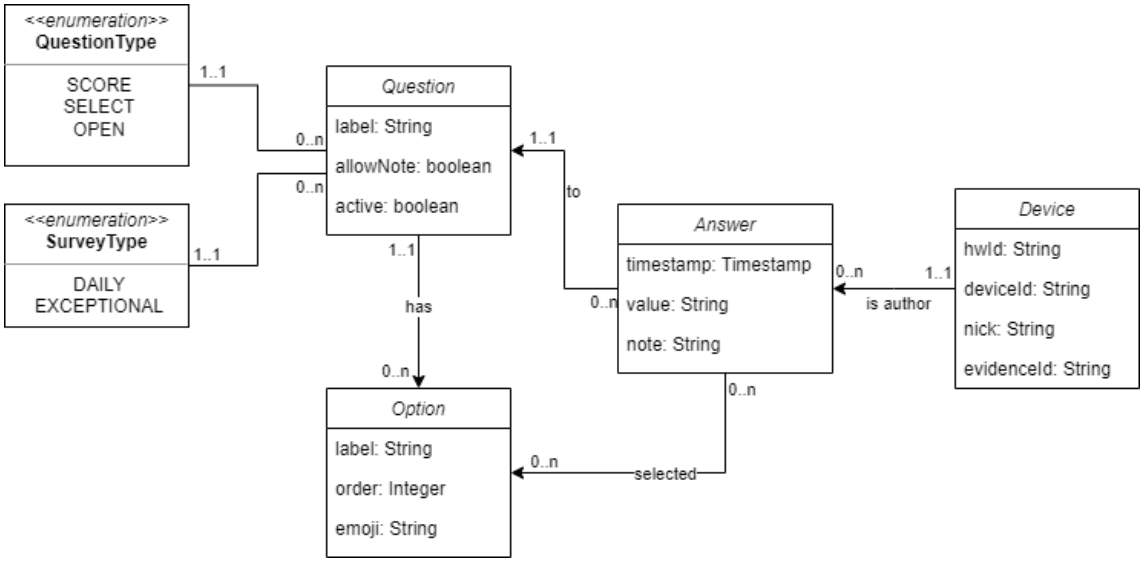
- **timestamp**

  Datum dotazníku, pro který byla tato odpověď vyplněna.
- **value**

  Hodnota odpovědi, která je zpracovávána dynamicky v závislosti na typu otázky. Může tedy jít o číslo, text či seznam vybraných hodnot.
- **note**

  Volitelná vlastní poznámka k odpovědi, bylo-li to u otázky povoleno.

Datový model pro dotazníky je popsán na diagramu 4.3.



Obrázek 4.3. UML class diagram datového modelu dotazníků

4.3 Technologie pro implementaci backendové části

Pro implementaci API a administračního rozhraní byl zvolen jazyk Java s použitím frameworku SpringBoot. Tato volba byla založena primárně na důvodu zkušenosti vývojářů s tímto jazykem a dostupnosti širokého množství nástrojů a knihoven, které dále výrazně usnadňovaly vývoj.

Spring Framework je aplikačním frameworkem a kontejnerem pro IoC (inversion of control) [15]. Jeho základní a nejdůležitější součástí je podpora dependency injection, která dovoluje snadnější správu a konfiguraci Java objektů a jejich vzájemných vztahů. Spring Framework přebírá od vývojáře zodpovědnost za spravování životního cyklu objektů, včetně vytváření nových objektů a poskytování jejich závislostí. Objekty, které Spring Framework takto spravuje, se nazývají *beans* a mohou být konfigurovány skrze XML soubory či přímo v kódu pomocí anotací. Díky této vlastnosti je možné pracovat s přehlednějším a snadněji rozšiřitelným kódem.

Vedle dependency injection kontejneru nabízí Spring řadu dalších funkcionalit v rámci tzv. projektů, jako například Spring Security pro zabezpečení aplikace či Spring Data pro perzistenci dat do různých typů databází. V backendové aplikaci tyto projekty použity.

Nevýhodou frameworku Spring je, že v základu vyžaduje velké množství nastavení i pro běžné scénáře použití. Tuto nevýhodu se snaží vyřešit SpringBoot [16], který aktivuje automatickou konfiguraci. Spring poté dokáže inteligentně rozpoznat, jaký typ aplikace se snažíme vytvořit, a podle toho zvolí vhodnou základní konfiguraci pro daný scénář použití. V případě běžných scénářů je tedy vývojář zbaven povinnosti tuto konfiguraci vytvářet sám. SpringBoot zároveň nabízí tzv. starter balíčky, které agregují množství populárních knihoven. To umožňuje rychlé vytvoření projektu bez nutnosti vybírat jednotlivé závislosti, které jsou k vývoji potřeba. Příkladem je např. Springboot Web Starter, který obsahuje knihovny pro tvorbu webových aplikací.

4.4 Databáze

Jelikož v rámci návrhu aplikace bylo zvoleno využití relační databáze, jako systém pro správu databáze byl vybrán PostgreSQL. Ten je distribuován pod vlastní open-source licencí, která není nijak zpoplatněna. Primárním faktorem při volbě PostgreSQL namísto jiných volně dostupných systémů (např. MySQL, MariaDB, ...) byla předchozí zkušenost vývojářů s tímto systémem z ostatních projektů.

Samotné nastavení vytvoření databází, uživatelů a přiřazení oprávnění v systému dochází při prvotním spuštění a je definováno ve skriptu `init-db.sh`. Samotné vytvoření schématu jednotlivých databází je již odpovědností jednotlivých aplikací, které databáze vlastní. V případě administračního rozhraní dochází k vytvoření schématu pomocí objektově-relačního mapovacího frameworku Hibernate.

Pro snadnou správu a přístup k datům v databázi byla zároveň na server nasazena administrační aplikace pgAdmin 4. Jedná se o nejpopulárnější nástroj pro použití se databázovým systémem PostgreSQL. Během vývoje i produkčních studií výrazně usnadnil ověřování zpracovaných dat přímo v databázi.

4.5 API pro mobilní aplikaci

Administrační rozhraní poskytuje systému rozhraní ve formě HTTP endpointů. Zejména se jedná o endpointy pro příjem nasbíraných dat z koncových zařízení, endpoint pro poskytnutí informací o pacientovi a endpointy pro poskytnutí dotazníků a přijímání odpovědí na ně.

Požadavky na toto API ze zařízení jsou autentizovány tajemstvím sdíleným mezi zařízeními a serverem, která jsou posílána v hlavičce požadavků. O autentizaci se stará třída `DeviceAuthFilter`, která je součástí tzv. Security Filter Chainu [17] frameworku Spring. Tato třída kontroluje přítomnost a správnost sdíleného tajemství. Pokud není v hlavičce uvedeno nebo je nesprávné, požadavek je odmítnut.

Pro práci s daty byl využit návrhový vzor DTO (Data Transfer Object) [18]. Tento vzor je použit primárně při práci se vzdálenými rozhraními, jako jsou webové služby, a dovoluje zapouzdřit více dat do jednoho serializovaného objektu a ten přenášet pouze jedním požadavkem. Vyhneme se tak přenášení dílčích částí dat odděleně, což zásadně sníží náročnost celé operace.

4.5.1 API pro sběr dat

Příchozí data odeslaná z mobilní aplikace přijímá endpoint `/api/gadget-bridge`. Součástí příchozích dat jsou i verifikační data ze zařízení, obsahující jeho MAC adresu a ID, na základě čehož je zařízení verifikováno. Pokud je příjem dat zakázán, požadavek je zahozen a mobilní aplikace dostane odpověď s chybou `HTTP 403 FORBIDDEN`. Aby nedocházelo k zahlcení systému, velikost příchozích dat je omezena na 50 záznamů.

V případě, že zařízení je zaregistrováno, není zakázáno a velikost dat v požadavku nepřekračuje omezení, všechna přijatá data (dle modelu 4.2) jsou úspěšně zpracována a uložena do databáze. Zároveň dojde k aktualizaci času poslední úspěšné synchronizace, která je viditelná v uživatelském rozhraní.

4.5.2 API pro informace o pacientovi

Serverová aplikace vystavuje endpoint `/api/patient/{HW_ID}`, který poskytuje informace o pacientovi přiřazeného danému zařízení.

```
HTTP GET /api/patient/AA:BB:CC:12:34:56
```

```
{
    "id": 23,
    "start_date": "2021-11-26",
    "intervention_start_date": "2022-06-01",
    "steps_target": 4500,
    "notifications_enabled": true,
    "morning_notifications_time": "10:00:00",
    "afternoon_notifications_time": "14:15:00",
    "survey_time": "20:00:00",
    "steps_target_history": [
        {
            "from_timestamp": "2022-06-04T22:00:00.000+00:00",
            "to_timestamp": null,
            "steps_target": 4500
        }
    ],
    "interventional": true
}
```

4.5.3 API pro dotazníky

Mobilní aplikace nejprve skrze endpoint `/api/survey/questions` získá veškeré dostupné otázky, které se nacházejí v databázi. Tyto otázky si následně mobilní aplikace uloží do lokální databáze, aby se snížila závislost funkce na mobilních datech a aplikace lépe fungovala dočasně offline. Seznam otázek je zaslán ve tvaru jako v následujícím případu:

```
HTTP GET /api/survey/questions
```

```
{
    "questions": [
        {
            "id": 1,
            "type": "SCORE",
            "question": "Měl jste dnes dechové obtíže v klidu?",
            "options": [
                {
                    "id": 1,
                    "label": "vůbec ne",
                    "order": 1,
                    "emoji": null
                },
                {
                    "id": 2,
                    "label": "maximální tíže",
```

```
                    "order": 2,
                    "emoji": null
                }
            ],
            "survey_type": "DAILY",
            "with_note": false,
            "for_monitoring": true
        },

        ...
}
```

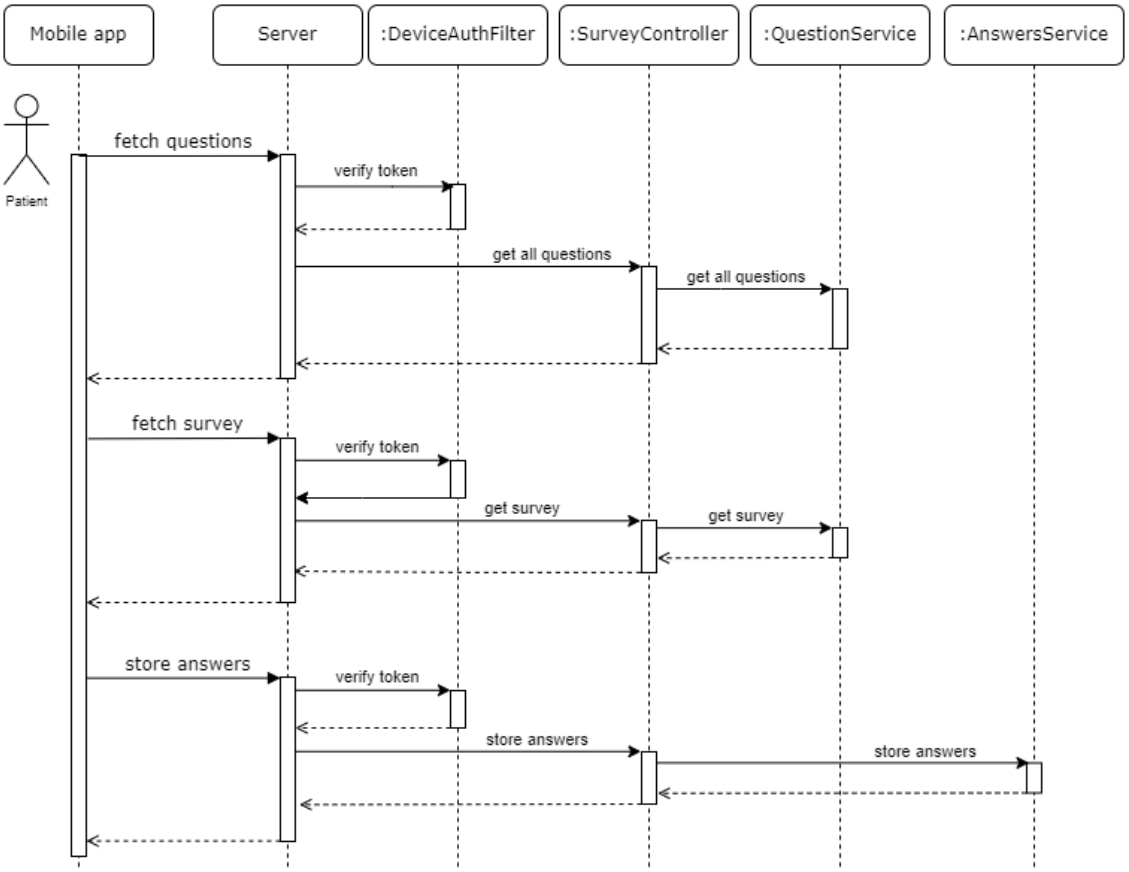
Aplikace následně zavolá endpoint `/api/survey/questions/user/{userId}`, kterým získá seznam ID otázek pro konkrétního uživatele a jejich pořadí v dotazníku.

Jakmile pacient vyplní dotazník a synchronizuje data s backendovým rozhraním, mobilní aplikace pošle data na endpoint `/api/survey/answers`. Zde dojde k identifikaci konkrétního zařízení podle jeho MAC adresy. Následně dojde k zpracování a uložení veškerých odpovědí do databáze. V průběhu dochází k validaci správného formátu odpovědí a nevalidní odpovědi jsou zahozeny, o čemž je mobilní aplikace informována odpovídajícím návratovým HTTP statusem. Příklad formátu požadavku s odpověďmi lze vidět zde:

```
HTTP POST /api/survey/answers
```

```
{
  "verification_data":
  {
      "wristband_mac": "AB:CD:EF:11:22:33",
      "device_id":"12bcf114abcdef"
  },
  "answers":
  [
    {
      "id": 5,
      "value": "4",
      "note": "Poznámka k odpovědi",
      "timestamp": 1626472800
    },
    ...
  ]
}
```

Celý tento proces popisuje následující sekvenční diagram.



Obrázek 4.4. UML Sekvenční diagram získávání otázek a ukládání odpovědí

4.6 Technologie pro implementaci frontendové části

Frontendová část aplikace je napsána za pomocí templatovacího enginu Thymeleaf a využívá integraci pro modul Spring MVC. Jedná se o server-side templatovací engine, který používá HTML předlohy (templates). V těchto předlohách je předepsán základ HTML souboru, do kterého jsou však pomocí speciální syntaxe vloženy proměnné odpovídající datům v Java třídách. Thymeleaf při příchozím požadavku tuto šablonu načte, zpracuje, doplní o konkrétní data a výsledný HTML soubor odešle uživateli, jehož webový prohlížeč jej následně zobrazí.

Alternativou pro využití server-side templatovacího enginu by byla implementace klientské single-page aplikace pomocí Javascriptu, například za použití frameworku jako React či Vue. V takovém případě by veškeré vykreslování obsahu webové stránky probíhalo na klienské straně, zatímco ze serveru by byla stahována pouze relevantní data. Neduhem tohoto řešení by byla nutnost na serverové straně zároveň vytvořit komplexní REST API, skrze které by klientská aplikace mohla komunikovat. Problémem zároveň může být i latence, kdy v případě single-page klientských aplikací může první načtení trvat podstatně déle oproti server-side renderování. Výhodou single-page aplikace by naopak byl snadnější vývoj dynamicky se chovajícího uživatelského rozhraní, jelikož při využití template enginu je nutné velice často překreslovat celou stránku namísto jejích částí.

Pro zjednodušení vytváření frontendu byl použit open-source CSS framework Bulma. Ten poskytuje řadu předpřipravených CSS tříd pro různé UI komponenty, což výrazně urychluje vývoj uživatelského prostředí v HTML.

4.7 Administrační rozhraní

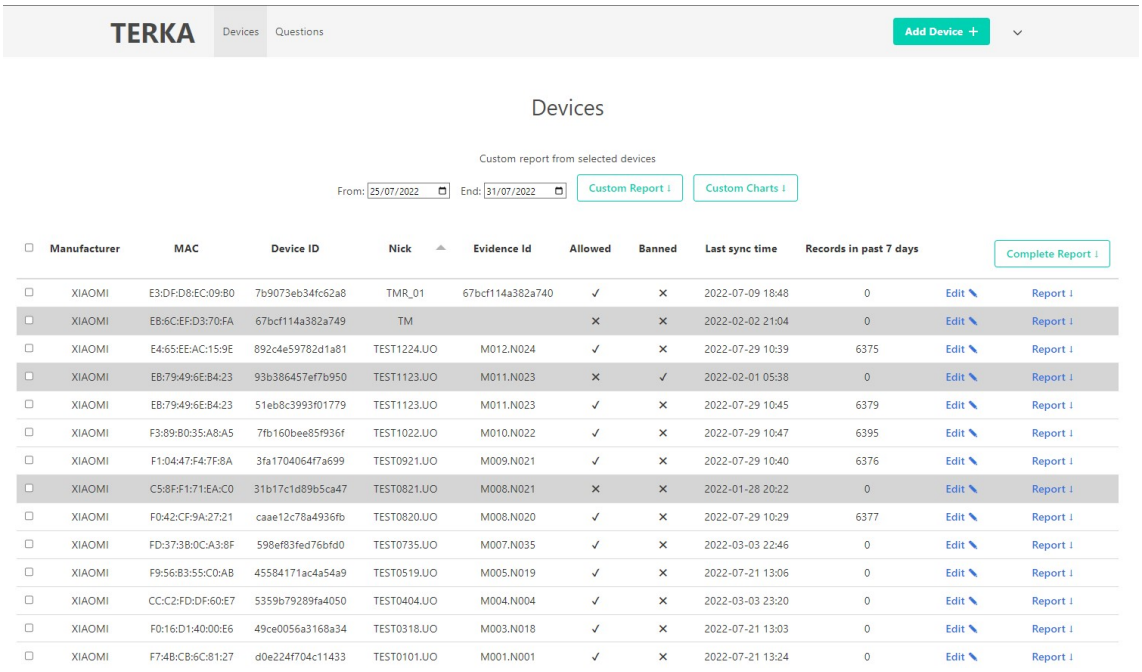
Aby bylo lékařům a administrátorům umožněno snadno spravovat celý systém, došlo ke vzniku webového administračního rozhraní.

Přístup do administračního zabezpečen kombinací emailové adresy a hesla. Pro přístup je nejprve nutné se do systému zaregistrovat a při registraci zadat tzv. *invite kód*, který je uživateli předem poskytnut. Po registraci je možné se přihlásit a přistupovat ke všem částem systému.

4.7.1 Administrační rozhraní pro správu zařízení a pacientů

Úvodní stránkou systému je seznam všech zařízení, které jsou do systému zaregistrované. Kliknutím na hlavičku sloupce lze tuto tabulku seřadit. V případě, že je zařízení vyřazené nastavením `Allowed` či `Banned` také dojde k vizuálnímu oddělení od ostatních zařízení.

Tato obrazovka zároveň obsahuje i tlačítka a nastavení pro vytváření exportů dat. V horní části obrazovky lze pomocí dvou selektorů dat zvolit časový rozsah pro export. Položky seznamu zařízení je také možné jednotlivě označit, což následně umožňuje vytvořit export dat z vícero zařízení.

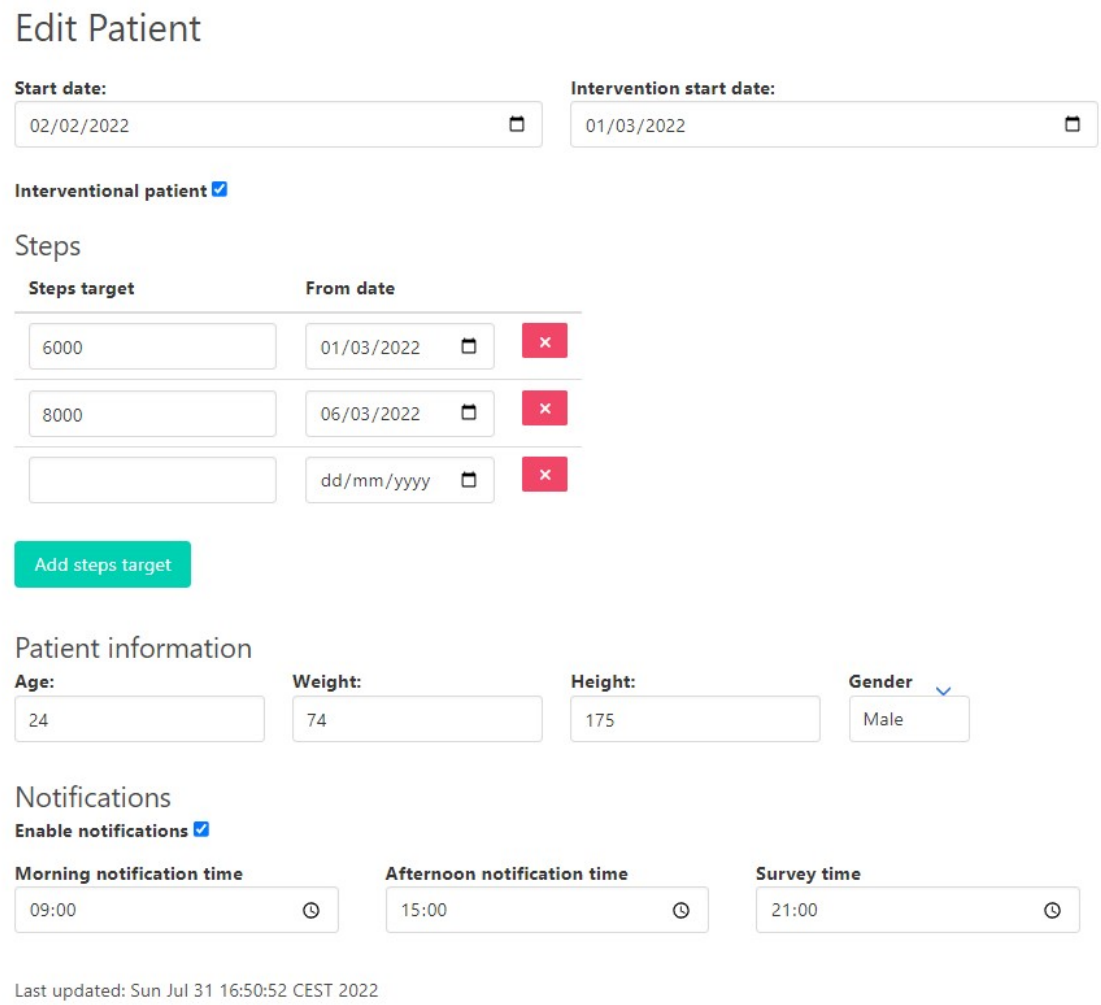


Obrázek 4.5. Seznam zaregistrovaných zařízení

Nové zařízení lze přidat pomocí tlačítka Add Device skrze formulář 4.6. Zde je možné nastavit jeho základní údaje získané z aplikace Gadgetbridge.

Obrázek 4.6. Formulář pro vytvoření či editaci zařízení

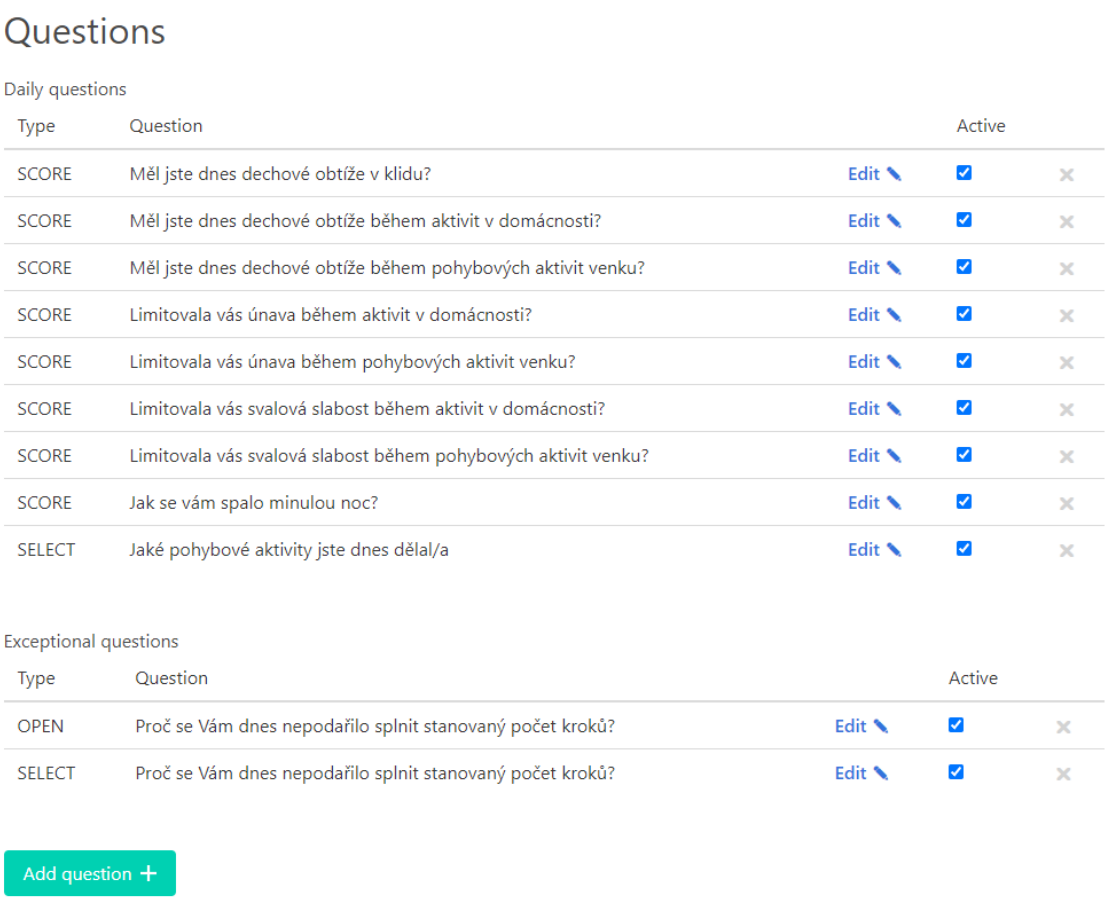
Stejný formulář je rozšířen o sekci pro editaci údajů o pacientovi.



Obrázek 4.7. Formulář pro editaci dat o pacientovi

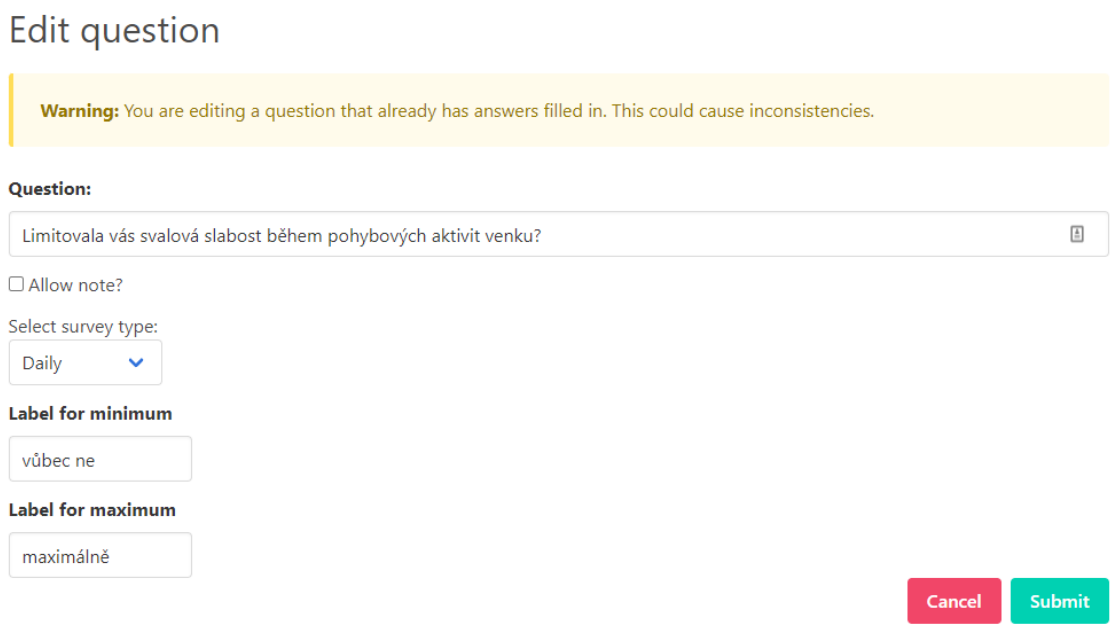
4.7.2 Administrační rozhraní pro tvorbu otázek

Aby bylo možné otázky jednoduše vytvářet a editovat, došlo k rozšíření administračního uživatelského rozhraní o sekci `Questions`, která tyto možnosti nově nabízí. Na úvodní obrazovce této sekce lze vidět všechny otázky, které jsou v systému vytvořené. Zde je možné vytvořit novou otázku, upravit již existující, nastavit aktivní status otázky nebo ji smazat.



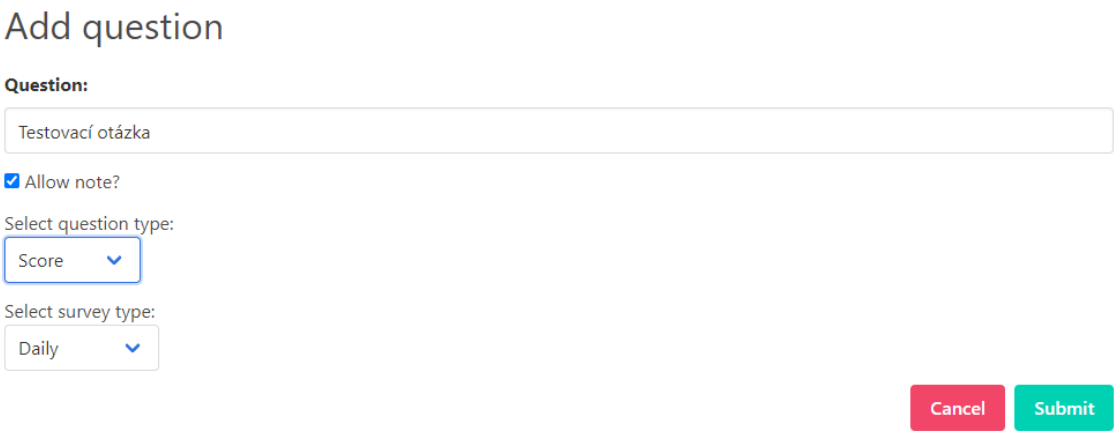
Obrázek 4.8. Seznam vytvořených otázek v administračním rozhraní

Aby nedocházelo v systému k nekonzistencím, je dovoleno mazat pouze otázky, na které ještě nebyla zaznamenána žádná odpověď. Otázky s odpověďmi je stále možné editovat, uživateli je ale zobrazena varovná hláška o možném dopadu na konzistenci systému.



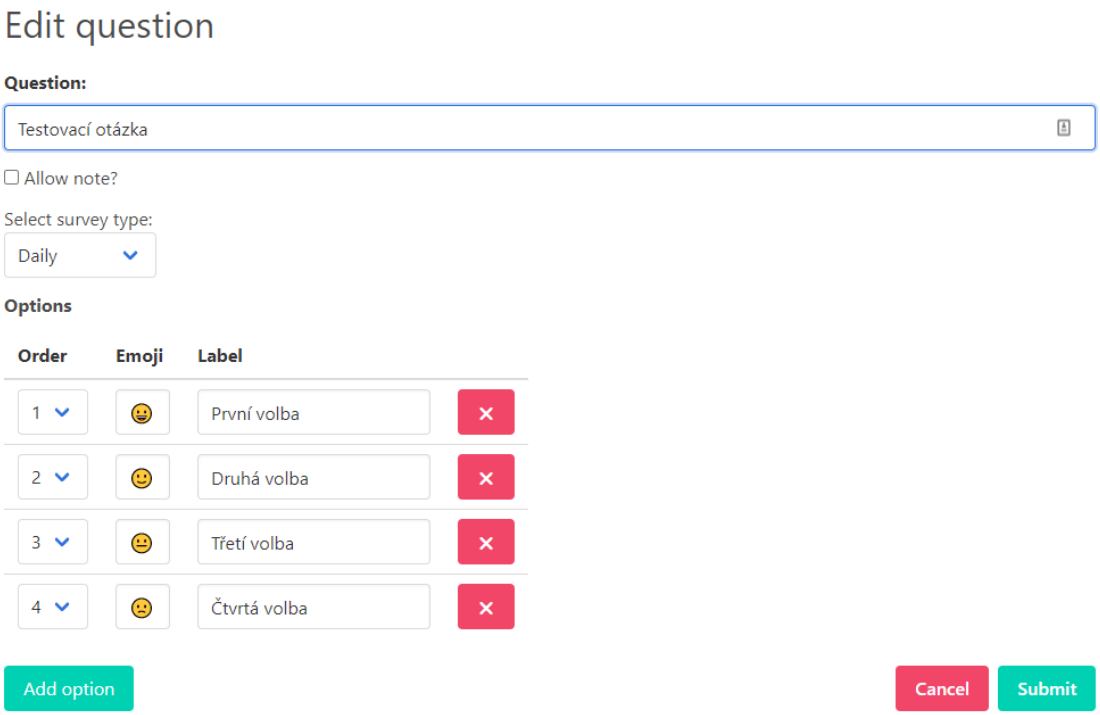
Obrázek 4.9. Editace otázky s varováním o existenci odpovědí

Pro přidání otázky je nejprve nutné vyplnit základní informace jako titulek otázky, možnost povolení komentáře a také nastavení typu otázky a dotazníku dle toho, jakou otázku se uživatel snaží založit. Pokud zvolený typ otázky vyžaduje další nastavení, po potvrzení těchto prvotních změn je uživatel přesměrován na nastavení specifické pro konkrétní zvolený typ otázky.



Obrázek 4.10. První krok přidání nové otázky

Otevřená otázka, neboli otázka typu OPEN, žádná další nastavení nevyžaduje. V případě otázky typu SCORE, tedy otázky z odpovědí v rozsahu 0 až 5, je nutné ještě vyplnit popisek pro minimální a maximální hodnotu. Otázka typu SELECT, neboli výběr ze seznamu, vyžaduje vytvoření jednotlivých možností. Těmto možnostem je dovoleno určovat jejich pořadí a, pokud je to tak žádoucí, přidat k nim i UTF-8 emotikon pomocí přehledného emoji selectoru.



Obrázek 4.11. Nastavení otázky typu výběru z více možností

4.8 Export dat

Na základě požadavků byly implementovány dva různé druhy exportů dat - kompletní datový export, neboli nezpracovaná data z koncových zařízení, a tabulka formátu XLSX obsahující agregovaná data a přehledné grafy.

4.8.1 Kompletní datový export

Pro kompletní datový export vznikl endpoint `/api/export/complete`, který přijímá tři volitelné parametry:

- **deviceIds**
  Seznam ID zařízení, jejichž data mají být obsažena v reportu.
  V případě nevyplnění zahrnuje report veškerá zařízení.
- **timestampFrom**
  Datum ve formátu `YYYY-MM-DD` určující, od jakého dne chceme vygenerovat report. Starší data nebudou součástí reportu.
  Výchozí hodnota je nastavena na 30 dnů od současného datumu.
- **timestampTo**
  Datum ve formátu `YYYY-MM-DD` určující, do jakého dne chceme vygenerovat report. Novější data nebudou součástí reportu.
  Výchozí hodnota je nastavena na současné datum.

Po přijetí požadavku na tento endpoint server vygeneruje a uživateli odešle CSV soubor s daty pro vybraná zařízení a časové rozmezí. Pro vygenerování CSV souboru byla použita třída `CSVPrinter` pomocné knihovny Apache Commons ve verzi 1.8.

Příklad obsahu výsledného exportu lze vidět na obrázku 4.12.

```
Device MAC,Nickname,Evidence ID,Datetime,Steps,Met Minute,Kind,Heart rate,Intensity
CB:C7:E2:81:F1:95,T2.FN01N45,FN01N45,2022-07-14 00:00:00.0,0,0.00,121,78,21
CB:C7:E2:81:F1:95,T2.FN01N45,FN01N45,2022-07-14 00:01:00.0,0,0.00,240,77,0
CB:C7:E2:81:F1:95,T2.FN01N45,FN01N45,2022-07-14 00:02:00.0,0,0.00,240,78,0
CB:C7:E2:81:F1:95,T2.FN01N45,FN01N45,2022-07-14 00:03:00.0,0,0.00,240,83,18
CB:C7:E2:81:F1:95,T2.FN01N45,FN01N45,2022-07-14 00:04:00.0,0,0.00,250,75,0
CB:C7:E2:81:F1:95,T2.FN01N45,FN01N45,2022-07-14 00:05:00.0,0,0.00,240,74,0
CB:C7:E2:81:F1:95,T2.FN01N45,FN01N45,2022-07-14 00:06:00.0,0,0.00,240,72,0
CB:C7:E2:81:F1:95,T2.FN01N45,FN01N45,2022-07-14 00:07:00.0,0,0.00,240,70,0
CB:C7:E2:81:F1:95,T2.FN01N45,FN01N45,2022-07-14 00:08:00.0,0,0.00,240,71,0
CB:C7:E2:81:F1:95,T2.FN01N45,FN01N45,2022-07-14 00:09:00.0,0,0.00,240,70,0
CB:C7:E2:81:F1:95,T2.FN01N45,FN01N45,2022-07-14 00:10:00.0,0,0.00,240,72,18
```

Obrázek 4.12. Ukázka kompletního datového exportuve formátu CSV

4.8.2 Export ve formě XLSX tabulky

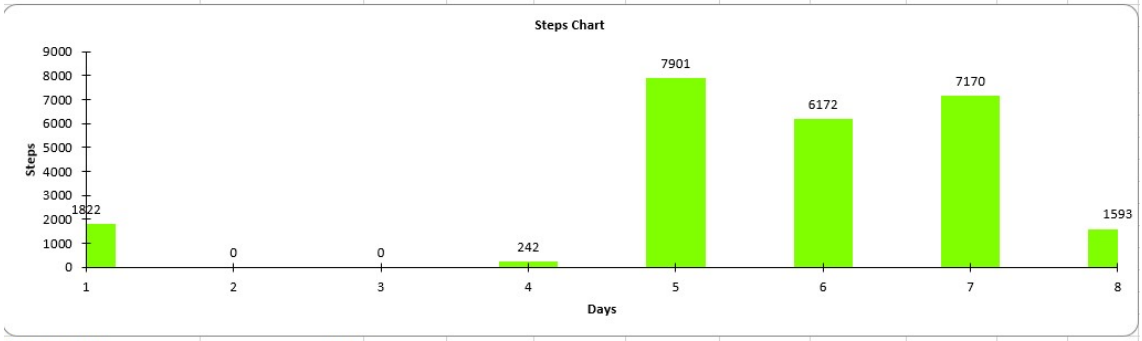
Pro generování reportu ve formátu XLSX tabulky vznikl na straně backendové části endpoint `/api/report/daily-charts`. Přijímá stejné parametry jako endpoint pro generování kompletních datových exportů, tedy parametry `deviceIds`, `timestampFrom` a `timestampTo`.

Export dle požadavků obsahuje agregovaná data o intezitě pohybu za každý den ze zvoleného rozsahu. Tato data jsou následně zobrazena ve sloupcových grafech a doplněna o vypočtené průměry z hodnot. Pokud je k zařízení přidělen pacient, který v daném období vyplňoval odpovědi na každodenní dotazníky, tyto odpovědi jsou rovněž vypsány v rámci této tabulky.

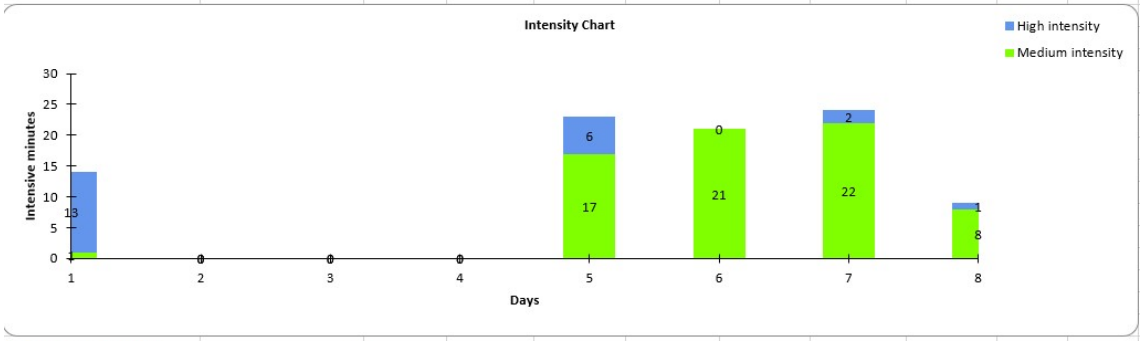
Pro generování XLSX tabulky byla použita knihovna Apache POI ve verzi 4.1.0, která poskytuje Java rozhraní pro práci s Microsoft dokumenty.

Příklad reportu vygenerovaného pro jedno zařízení s časovým rozsahem 14. 7. 2022 až 21. 7. 2022 lze vidět na obrázku 4.13.

Dates (14/07/22-21/07/22)	Thu (14/07)	Fri (15/07)	Sat (16/07)	Sun (17/07)	Mon (18/07)	Tue (19/07)	Wed (20/07)	Thu (21/07)
Number of steps	1822	0	0	242	7901	6172	7170	1593
Low activity minutes	1427	1441	1441	1441	1418	1420	1417	749
Medium activity minutes	1	0	0	0	17	21	22	8
High activity minutes	13	0	0	0	6	0	2	1



Average steps:	3112



High intensity average:	2
Medium intensity average:	8
Total intensity average:	11

Obrázek 4.13. Ukázka exportu ve formátu XLSX tabulky

Kapitola 5
Testování

5.1 Unit testy

Jednotkové neboli unit testy jsou dílčí testy, jejichž úkolem je ověřovat funkčnost nejmenších celků projektu, v tomto případě testování jednotlivých tříd a metod Java kódu. Tyto testy by na sobě měly být nezávislé a testovat pouze jedinou věc. Aby toho mohlo být dosaženo, často je nutné nahradit závislosti testované třídy komponentami, které jsou plně pod kontrolou vývojáře. Jde např. o mock objekty, stuby a podobně.

Běžnou metodou použití unit testů je takzvaný test-driven develompent (TDD) [19]. Tato metoda je často používána v agilních prostředích, které používají např. Scrum framework projektového řízení. Při TDD jsou neprve implementovány testy před samotnou funkcionalitou. Dochází tak jednak k velmi brzkému odhalení případných chyb, které může nový zásah do kód způsobit, druhak i k širokému pokrytí kódu testy.

V prostředí Java existuje pro unit testy široké množství frameworků, nejpopulárnějším z nich je *JUnit*. Ten poskytuje velice rozsáhlé API pro spouštění testů a ověřování očekávaného chování testovaných tříd a metod. Rovněž pro mockování objektů existuje řada knihoven jako *Mockito* či *PowerMock*.

V prostředí serverové aplikace byl použit balíček `spring-boot-starter-test` ve verzi 2.4.3, který zahrnuje zmiňované knihovny JUnit i Mockito. Proběhla identifikace nejkritičtějších částí kódu, které byly následně pokryty jednotkovými testy.

V jednotkových testech bylo zároveň použito fluent API knihovny *AssertJ*. Ta dovoluje psát výrazy pro ověření stavu testované metody ve snadno čitelném formátu, což zlepšuje udržitelnost a čistelnost testů.

Pokrytí tříd [%]	*Pokrytí metod [%]*	*Pokrytí řádek kódu [%]*
47	15	19

Tabulka 5.1. Přehled pokrytí kódu testy

5.2 End-to-end testování

Aby bylo možné systém TERESA otestovat jako celek, v několika fázích projektu proběhly tzv. end-to-end testy. Jejich účelem je ověřit, jak systém funguje od začátku do konce celého procesu použití. Verifikuje, zda spolu veškeré komponenty komunikují správně a chovají se dle očekávání. Je tedy důležité tyto testy provádět co možná nejblíže reálným scénářům užití.

Všichni vývojáři na projektu TERESA měli lokálně zprovozněno kompletní prostředí. To obsahuje náramek Xiaomi Mi Band 5, mobilní aplikace Gadgetbridge a Konektor a korektní nastavení zařízení a odpovídajícího pacienta na straně serveru. Průběžně tedy měli možnost zkoušet, zda některá ze změn nezpůsobila defekty nebo kolaps systému. V případě odhalení defektu je tak možné příčinu snadněji identifikovat a debugovat.

V několika fázích projektu, zejména před očekávaným začátkem produkční studie, došlo k end-to-end testům se širší skupinou testovacích uživatelů. Uživatelům bylo zprovozněno celé prostředí a předány informace, jak s ním pracovat. V následujících cca 14 dnech proběhlo testování, které se co nejvíce blížilo reálnému případu použití u pacientů. Uživatelé byli instruováni vykonávat pohybovou aktivitu, pracovat s mobilní aplikací, kontrolovat její notifikace a vyplňovat každodenní i vyjímečné dotazníky. Někteří uživatelé byli taktéž označeni jako součást intervenční skupiny, tudíž jim byly nastavovány a navyšovány denní cíle počtu kroků, které se snažili vykonat. Vývojáři byli v tento čas k dispozici administrátorům a lékařům pro průběžné generování reportů a kontrolu, zda je na straně backendu vše v pořádku. Došlo tak k ověření, že je systém funkční a připravený na produkční studii.

5.3 Uživatelské akceptační testy

Uživatelské akceptační testy (UAT) jsou druh testů, jejichž primárním účelem není odhalení defektů či splnění specifikací, nýbrž ověření, zda zákazníkovi dodaná funkcionalita vyhovuje [20]. Ta je tak testována primárně klíčovými uživateli systému, kteří jsou obvykle schopni podat nejužitečnější zpětnou vazbu.

Nově vyvinutá funkcionalita byla průběžně nasazována do produkčního prostředí a v rámci pravidelných týmových schůzek prezentována stakeholderům společně s případným předáním potřebných návodů, artefaktů (v případě mobilní aplikace) a dokumentace. V tento moment nastala fáze uživatelského akceptačního testování, kdy lékaři a administrátoři zainteresovaní na projektu funkcionalitu vyzkoušeli a ověřili v praxi. V případě, kdy dodaná funkcionalita neodpovídala požadavkům či bylo nalezeno možné vylepšení, byli vývojáři kontaktováni s požadavkem na změnu, která byla zpravidla diskutována na pravidelných schůzkách a dodána v následujících dnech či týdnech. Tento proces dovoloval uživatelům doručovat co nejvíce užitečné funkcionality pro zrychlení a zlepšení pracovních procesů v projektu.

Kapitola 6
Nasazení

V této kapitole se blíže podíváme na způsob, jakým je aplikace nasazena a provozována.

Celý backend systému TERESA je nasazen na virtuálních serverech VPS [21], které projektu zajišťuje ČVUT. Tento způsob narozdíl od pronajímání virtuálního serveru od běžných poskytovatelů hostingových služeb přináší výhodu v zabezpečení dat. Jelikož zpracováváme a uchováváme potencionálně citlivá data pacientů, v privátní školní infrastuktuře máme tato data plně pod kontrolou a nejsou přístupná třetím stranám jako poskytovatelům hostingu.

Protože vývoj aplikace probíhá souběžně s aktivními studiemi, bylo nutné vytvořit dvě aplikační prostředí, a to produkční a vývojové. Tomu odpovídal i vývojový proces, kdy nově vyvinutá funkcionalita byla nejprve nasazena a otestována na vývojovém prostředí a následně v případě potřeby (u hotfixů či naléhavé funkcionality) integrovány do produkčního kódu, který byl nasazen na produkční server.

6.1 Operační systém, systémové prostředky

Oba virtuální servery, produkční i vývojový, používají operační systém (dále OS) Linux, konkrétně distribuci Debian ve verzi 10, označován kódovým názvem Buster.

Použití OS Linux přináší pro účely projektu TERESA oproti využití OS Windows Server řadu výhod [22]. Linux je open-source operační systém distribuovaný pod licencí GNU General Public License (GPL) verze 2, což přináší možnost provozovat tento OS bez nutnosti platit licenční poplatky, jako je tomu u Windows Server. Dále je Linux pro účely serverového prostředí velice oblíbený [23] a existuje kolem něj široká otevřená komunita. Nevýhodou může být subjektivně nižší uživatelská přívětivost, nicméně to díky zkušenosti vývojářů pracujících na tomto projektu s tímto operačním systémem nevytvářelo velký problém.

Využití Windows Server by však mohlo přinést výhody v případě, kdyby byl projekt závislý na některé z Microsoft služeb či je integroval, například službu Active Directory (AD). Toto ovšem není v současném ani budoucím vývoji plánováno, tudíž byl po zvážení uvedených faktorů pro účely projektu zvolen OS Linux.

Každé z prostředí má přiděleno dvě procesorová vlákna. Produkční prostředí pracuje s 10 GB operační paměti RAM, vývojové prostředí pouze s 6 GB RAM. Aplikace byla na takto relativně omezených prostředcích otestována a žádné problémy s výkonem nebyly nikdy v průběhu odhaleny. Tato skutečnost je současně dána zejména nízkým množstvím uživatelů, kteří k aplikaci přistupují souběžně. Ze zkušeností získané během pilotní studie a vývoje je však nutné podotknout, že pro použití systému ve větším rozsahu, konkrétně s větším množstvím administrátorů, pacientů a včetně nasazení další funkcionality vyvíjené mimo rozsah této práce (služby pro načítání dat z platforem Garmin a Fitbit), by bylo nutné tyto prostředky rozšířit.

Další výhodou využití využití OS Linux je jeho nativní podpora pro Docker, na kterém je infrastruktura aplikace závislá, viz dále.

Veškerá konfigurace a údržba prostředí probíhá skrze SSH terminál. Pro nejlepší zabezpečení vzdáleného přístupu je na straně serveru deaktivováno přihlášení heslem. Vzdálený přístup je tak možný pouze za použití kombinace veřejného a soukromého RSA klíče.

6.2 Reverzní proxy NGINX

Jelikož se backend sysému skládá z více služeb mimo administrační rozhraní a tyto služby jsou rozdělené do různých služeb, kdy je každá služba spuštěna v jiném Docker containeru a využívá jiný port, bylo nutno příchozí požadavky správně směřovat na základě cílové URL. Pro tento účel byl zvolen webový server NGINX a konkrétně jeho funkcionalita pro reverzní proxy.

Reverzní proxy [24] je jednotným vstupním bodem do systému a rozděluje přijatý provoz mezi vícero serverů. Jedná se o důležitý prvek distribuovaných systémů a dokáže například vyvažovat zátěž mezi více serverů, šifrovat provoz, cachovat či komprimovat statický obsah. V tomto případě byla využita pouze funkcionalita pro rozdělení příchozích požadavků odpovídajícím službám, a to na základě cílové URL požadavku. Pokud tato URL obsahuje cestu, která odpovídá některé ze služeb (např. `/api/gadget-bridge`), je požadavek předán ke zpracování příslušné službě. Po zpracování požadavku službou reverzní proxy zpracovanou odpověď zasílá zpět uživateli, pro kterého byla tato proxy jediným rozhraním pro interakci se systémem.

6.3 SSL certifikáty

Pro zajištění bezpečné komunikace mezi uživateli a serverem a šifrování pomocí protokolu HTTPS, aplikace využívá SSL certifikáty. Na začátku komunikace proběhne takzvaný SSL handshake [25]. V jeho první fázi pošle klient serveru `client hello` zprávu obsahující informace o verzi SSL nebo TLS, které klient podporuje. Server na tuto zprávu odpoví zprávou `server hello` a zároveň pošle digitální certifikát. Klient jej ověří a zkontroluje, zda patří mezi důvěryhodné certifikáty. Pokud ano, v další komunikaci dojde k výměně sdíleného tajemství. Tento handshake je mezi serverem a klientem zakončen vzájemnou výměnou zprávy `finished`, která je již zašifrována klíčem, který byl v přechozím kroku předán. Následující komunikace již probíhá pouze v šifrované podobě, a to symetrickou šifrou.

Pro účely projektu byly vystaveny dva certifikáty, a to zvlášť pro doménu pc2e.felk.cvut.cz, tedy produkční prostředí, a dev.pc2e.felk.cvut.cz neboli vývojové prostředí. Tyto certifikáty jsou uloženy na serverech ve formátu PKCS#12 (P12). Jedná se o formát archivu určený k uložení více druhů kryptografických objektů ve formě jediného souboru, a je běžně použit pro sjednocení privátního klíče, certifikátu a mezilehlých certifikátů od certifikační autority. Stejným způsobem je tento soubor využit i v tomto projektu.

6.4 Docker

Docker [26] je open-source platforma pro virtualizaci containerů, neboli velice odlehčených virtuálních strojů. Běžné virtuální stroje (virtual machines, VM) obsahují kompletní kopii celého operačního systému, na kterém je aplikace následně spuštěna. Tyto aplikace jsou následně ovládány tzv. hypervizorem. Avšak tím, že každý spuštěný virtuální stroj má svůj vlastní operační systém, vziká velká režie na systémových prostředcích

vyžadovaných pro běh systému. V komplexních systémech může vznikat potřeba pro běh desítek i stovek mikroslužeb (např. v případě společnosti Netflix až tisíc služeb [27])

Tento problém se Docker snaží vyřešit. Jádro operačního systému je v prostředí Dockeru sdílené napříč všemi containery s hostitelským systémem, který provozuje Docker Engine. Díky tomu nemusí všechny služby nasazené pomocí Dockeru mít zabalený svůj vlastní operační systém. Tato funkcionalita je možná díky využití vlastností operačního systému Linux, jako jsou jmenné prostory či `cgroups` (control groups) pro izolaci procesů. Díky nim je možné provozovat služby bezpečně a izolovaně obdobně, jako kdyby byly nasazené v obyčejných virtuálních strojích.

Docker rozlišuje několik druhů objektů. Prvním z nich je image. Image je předloha, která obsahuje potřebné instrukce k vytvoření konkrétního Docker containeru, viz dále. Image jsou definovány pomocí tzv. `Dockerfile` souboru. Ten obsahuje instrukce např. z jakého jiný image má být použit jako základ, a dále konkrétní příkazy např. pro kompilaci a nasazení vlastní aplikace. Níže lze vidět příklad `Dockerfile` pro administrační rozhraní projektu TERESA:

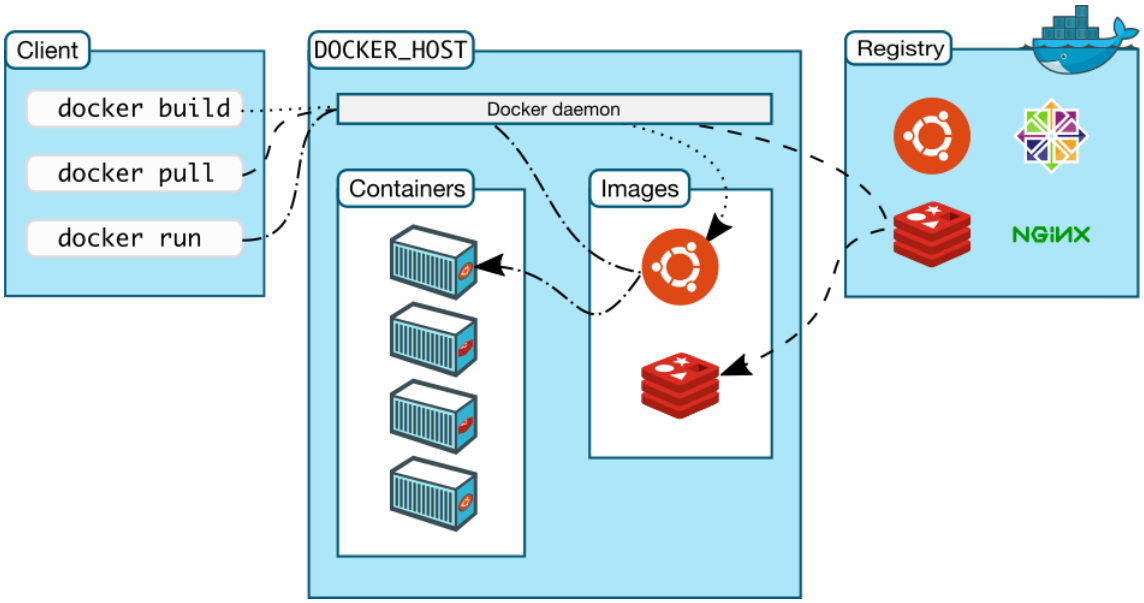
```
 # base image for build container
FROM maven:3-openjdk-11 as builder
ENV APP_HOME=/root/dev/app/
RUN mkdir -p $APP_HOME/src
WORKDIR $APP_HOME

 # cache Maven dependencies for future builds
COPY pom.xml $APP_HOME/pom.xml
RUN mvn dependency:go-offline

 # copy application into build container and build it using Maven
COPY ./src $APP_HOME/src
RUN mvn package -DskipTests

 # base image for runtime container
FROM openjdk:11-jre-slim-buster
ARG APPNAME
RUN apt-get update; apt-get install -y fontconfig libfreetype6
WORKDIR /usr/app/
 # copy application from build container to runtime container
COPY --from=builder /root/dev/app/target/*.jar .
EXPOSE 8080
 # run application
CMD ["java", "-jar", "/usr/app/server-0.0.1-SNAPSHOT.jar"]
```

Zajímavostí je využití tzv. multi-stage build procesu. Pokud bychom se rozhodli naši aplikaci zkompilovat, sestavit a následně i nasadit v rámci jednoho image, výsledný container podstatně nabyde na velikosti a i po nasazení bude obsahovat nástroje (kompilátor, build tools, Maven, ...), které jsou pro běh aplikace nepotřebné. Je tedy efektivnější rozdělit image do dvou částí, a to build a runtime image. Build image obsahuje veškeré potřebné nástroje k tomu, aby aplikaci dokázal ze zdrojového kódu zkompilovat a sestavit do spustitelného artefaktu. Runtime image je již podstatně odlehčený a obsahuje pouze věci nezbytné k samotnému spuštění aplikace. V `Dockerfile` je pak nadefinováno zkopírování sestaveného artefaktu z build image do runtime image. Výsledný

runtime image je objemově podstatně menší a mnohem vhodnější pro další distribuci, např. prostřednictvím Docker repository, internetového uložiště images.

Dalším typem Docker objektu je container. Container je konkrétní spustitelnou instancí image. Pomocí příkazů, které poskytuje Docker API či CLI, jej můžeme dle libosti vytvářet, spouštět, zastavovat či mazat. Containery jsou v základu pomocí dříve zmíněných vlastností OS Linux izolované od hostitelského systému i ostatních containerů. Pro jejich vzájemnou komunikaci mezi nimi můžeme však nadefinovat síť, skrze kterou mohou komunikovat. Aby byly služby zabalené v containeru dostupné i zvenčí Docker prostředí, je možné dále definovat port-forwarding a vytvořit tak mapování TCP či UDP portů mezi hostitelským systémem a Docker containerem. Veškerá data, se kterými container v základu pracuje, nejsou perzistentní a po smazání containeru jsou ztracena. Proto existuje možnost mapování hostitelského uložiště do containeru, ať už formou perzistentních Docker volumes, či přímým namapováním diskového prostoru hostitele.

Výše zmíněné prvky Dockeru jsou zobrazeny na obrázku 6.1.



Obrázek 6.1. Docker architektura [28]

Při nasazování většího množství Docker containerů by bylo nutné každý spouštět a konfigurovat skrze Docker API nebo CLI jednotlivě a následně manuálně nastavit síť pro jejich vzájemnou komunikaci. V takovémto případě je vhodné namísto toho využít tzv. Docker Compose. Docker Compose je nástroj pro definici a provoz systémů skládajících se z několika Docker conteinerů. Základem tohoto nástroje je soubor formátu YAML, běžně nazvaný `docker-compose.yml`. Ten umožňuje v přehledném formátu na jednom místě nakonfigurovat jednotlivé služby, nastavit jejich proměnné prostředí, namapovat perzistentní souborové oddíly, nakonfigurovat port forwarding a další. Následně je již možné celý systém snadno ovládat pomocí `docker-compose` příkazů jako `up` (vytvoření a spuštění služeb), `restart`, `down` apod. Níže je jako příklad uveden ukázkový `docker-compose.yml` soubor části systému TERESA.

```
version: "3"
services:
  admin_ui:
    image: p2ceapp
```

```
    build:
      context: ./admin-ui
      dockerfile: Dockerfile
      args:
        APPNAME: "server-0.0.1-SNAPSHOT.jar"
    environment:
      - SPRING_DATASOURCE_URL=jdbc:postgresql://pg:5432/p2ce
      - SPRING_DATASOURCE_USERNAME=p2ce
      - TZ=Europe/Prague
    volumes:
      - ./logs:/usr/app/logs
      - ${PWD}/.env:/usr/app/.env
      - ./certs:/usr/app/certs
    ports:
      - "8443:8443"
    networks:
      - p2ce

...

networks:
  p2ce:
    external:
      name: p2ce_network
```

Všechny prvky infrastruktury, jako je databázový systém, reverzní proxy, ale i jednotlivé služby systému TERESA, jsou zabalené v Docker images a vzájemně propojené skrze Docker Compose, což umožňuje snadné a rychlé nasazení na nová prostředí, výrazně zrychluje vývoj a umožňuje snadné zprovoznění potřebné infrasturkury na lokálních vývojových prostředích.

Kapitola 7
Závěr

Cílem této práce bylo zanalyzovat požadavky pro administrační rozhraní a API pro sběr dat na projektu TERESA, a na jejich základě navrhnout a vyvinout požadovanou funkcionalitu.

Sběr požadavků probíhal v agilním procesu na pravidelných schůzkách se stakeholdery z Českého vysokého učení technického v Praze (ČVUT), Fakultní nemocnice Hradec Králové (FN HK), Univerzity obrany (UNOB) a Univerzity Palackého v Olomouci (UPOL). Veškerá požadovaná funkcionalita zde byla konzultována a následně demonstrována reálným uživatelům systému jako lékařům a administrátorům.

V rámci práce vznikla webová služba pro příjem, zpracování a uložení dat z pacienských náramků, která přicházela z oddělené mobilní aplikace, se kterou pracují pacienti účastnící se telecoachingu a studií. Toto rozhraní je zabezpečené na základě sdíleného tajemství mezi backendem a mobilními aplikacemi a zároveň umožňuje data z nežádoucích zařízení zablokovat. Po obdržení příchozího požadavku jsou tato data zpracována a uložena do databáze pro pozdější lékařskou analýzu.

Pro umožnění snadné správy jednotlivých zařízení, pacientů a možnosti zobrazit a vyexportovat data vzniklo rovněž webové administrační rozhraní určené pro lékaře a administrátory studií. Toto rozhraní umožňuje přidávat a spárovat jednotlivé náramky s rozhraním pro příjem dat, upravit informace o pacientech, kterým je zařízení přiděleno, aktivovat a nastavit časy notifikací pro mobilní aplikaci, nastavovat pacientům cílové počty kroků apod. Další důležitou součástí je správa dotazníků, které pacient pravidelně vyplňuje. Administrační rozhraní dovoluje definovat jednotlivé otázky, upravovat jejich typ, přidávat možné odpovědi a následně tyto otázky sestavit do finálního dotazníku.

Aplikace je postavena na vrstevnaté architektuře a implementována v jazyce Java, konkrétně ve frameworku SpringBoot. Jako nejvhodnější způsob ukládání zpracovaných dat byla zvolena relační databáze PostgreSQL.

Součástí práce byla zároveň příprava a údržba aplikačních prostředí. Tato údržba zahrnovala nasazení aplikace na produkční a vývojové prostředí, správu okolní infrastruktury jako databázového systému PostgreSQL nebo reverzní proxy Nginx. Aplikace i okolní infrastruktura je provozována v containerech platformy Docker. Zároveň bylo v rámci práce nutné připravit, nasadit a obnovovat SSL certifikáty pro zajištění provozu HTTPS protokolu, kterým administrační rozhraní i endpoint pro pacientská data komunikuje.

Celý systém TERESA byl pak v praxi testován vývojáři a dalšími vybranými uživateli, se kterými probíhala aktivní komunikace a díky ní docházelo k odhalování a opravě nedostatků, ať už technických chyb v implementaci, či problémů v designu a uživatelské přívětivosti.

Řešení již bylo využito v produkčních studiích a je připravené na další rozvoj v rámci dalších projektů. Další z těchto studií je v době psaní této práce ve fázi přípravy a její spuštění se očekává na přelomu léta a podzimu roku 2022 společně se zapojením nově vyvinutých funkcionalit.

Literatura

[1] Bures, Miroslav, Katerina Neumannova, Pavel Blazek, Matej Klima, Hynek Schvach, Jiri Nema, Michal Kopecky, Jan Dygryn a Vladimir Koblizek. A Sensor Network Utilizing Consumer Wearables for Telerehabilitation of Post-acute COVID-19 Patients. *IEEE Internet of Things Journal*. IEEE, 2022.

[2] Loeckx, Matthias, Roberto A Rabinovich, Heleen Demeyer, Zafeiris Louvaris, Rebecca Tanner, Noah Rubio, Anja Frei, Corina De Jong, Elena Gimeno-Santos, Fernanda M Rodrigues a et al. Smartphone-based physical activity telecoaching in chronic obstructive pulmonary disease: Mixed-methods study on patient experiences and lessons for implementation. *JMIR mHealth and uHealth*. 2018, ročník 6, č. 12. Dostupné na DOI 10.2196/mhealth.9774.

[3] Michalčíková, T., K. Neumannová a M. Salčáková. Přínos telerehabilitace pro nemocné s chronickou obstrukční plicní nemocí. *Studia Pneumologica Et Phthiseologica*. Zakouřilova 142, 149 00 Praha 4, Česká republika: TRIOS, s. r. o., duben, 2020, ročník 80, č. 2, s. 47–52. ISSN 1213-810X.

[4] Demeyer, H, Z Louvaris, A Frei, R A Rabinovich, C de Jong, E Gimeno-Santos, M Loeckx, S C Buttery, N Rubio, T Van der Molen a et al. Physical activity is increased by a 12-week semiautomated telecoaching programme in patients with COPD: A multicentre randomised controlled trial. *Thorax*. 2017, ročník 72, č. 5, s. 415–423. Dostupné na DOI 10.1136/thoraxjnl-2016-209026.

[5] Yetisen, Ali K, Juan Leonardo Martinez-Hurtado, Bar ş Unal, Ali Khademhosseini a Haider Butt. Wearables in medicine. *Advanced Materials*. Wiley Online Library, 2018, ročník 30, č. 33.

[6] Demeyer, H, Z Louvaris, A Frei, R A Rabinovich, C de Jong, E Gimeno-Santos, M Loeckx, S C Buttery, N Rubio, T Van der Molen a et al. Physical activity is increased by a 12-week semiautomated telecoaching programme in patients with COPD: A multicentre randomised controlled trial. *Thorax*. 2017, ročník 72, č. 5, s. 415–423. Dostupné na DOI 10.1136/thoraxjnl-2016-209026.

[7] ČVUT. *ČVUT FEL Aktuality Projekt TERESA umožní rehabilitaci pacientů po COVID-19 v domácím prostředí*. [vid. 2021-12-10]. Dostupné na `https://fel.cvut.cz/cz/aktuality/2021/teresa`.

[8] Xiaomi. *Mi Smart Band 4*. Dostupné na `https://www.mi.com/global/mi-smart-band-4/`.

[9] Garmin. *vívosport*. Dostupné na `https://www.garmin.com/en-US/p/574602/`.

[10] *Gadgetbridge*. Dostupné na `https://gadgetbridge.org/`.

[11] Fowler, Martin, Jim Highsmith a others. The agile manifesto. *Software development*. [San Francisco, CA: Miller Freeman, Inc., 1993-, 2001, ročník 9, č. 8, s. 28–35.

[12] Reddy, Niveditha. Dostupné na `https://www.digite.com/es/blog/waterfall-vs-agile-project-management/`.

[13] Richards, Mark. *Software architecture patterns*. O'Reilly Media, Incorporated 1005 Gravenstein Highway North, Sebastopol, CA ..., 2015.

[14] Farrell, Steve. *Using met-minutes to track volume of Physical Activity - Cooper Institute*. Dostupné na `https://www.cooperinstitute.org/2017/12/07/using-met-minutes-to-track-volume-of-physical-activity`.

[15] Walls, Craig. *Spring in action*. Simon and Schuster, 2022.

[16] Walls, Craig. *Spring Boot in action*. Simon and Schuster, 2015.

[17] *The security filter chain*. Dostupné na `https://docs.spring.io/spring-security/site/docs/3.0.x/reference/security-filter-chain.html`.

[18] Fowler, Martin. *Patterns of enterprise application architecture*. Addison-Wesley, 2015.

[19] Martin, Robert C. Professionalism and test-driven development. *Ieee Software*. IEEE, 2007, ročník 24, č. 3, s. 32–36.

[20] Myers, Glenford J, Corey Sandler a Tom Badgett. *The art of software testing*. John Wiley & Sons, 2011.

[21] Westfall, Jon. What is a virtual private server?. *Set Up and Manage Your Virtual Private Server*. 2021, s. 1–16. Dostupné na DOI 10.1007/978-1-4842-6966-4_1.

[22] Cabrera, Joe. Windows vs. Linux: A comparative study. *Proposal, Technical Writing, Blinn College, Bryan, TX. Accessed January*. 2009, ročník 21, s. 2019.

[23] Hussain, Shahid, Faisal Bahadur, Faqir Gul, Arif Iqbal, Ghazala Ashraf a Sumat Nazeer. Survey of Windows and Linux as server operating system. *International Journal of Computer (IJC)*. 2015, ročník 18, č. 1, s. 1–6.

[24] Sommerlad, Peter. Reverse Proxy Patterns.. In: *EuroPLoP*. 2003. s. 431–458.

[25] Hickman, Kipp a Taher Elgamal. The SSL protocol. Internet Draft RFC, 1995.

[26] Rad, Babak Bashari, Harrison John Bhatti a Mohammad Ahmadi. An introduction to docker and analysis of its performance. *International Journal of Computer Science and Network Security (IJCSNS)*. International Journal of Computer Science and Network Security, 2017, ročník 17, č. 3, s. 228.

[27] Tonse, Sudhir. *Scalable microservices at Netflix. challenges and tools of the trade*.

[28] Docker. *Docker overview*. Dostupné na `https://docs.docker.com/get-started/overview/`.