Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

Webová aplikace pro správu neurorehabilitačních programů

Plzeň 2019 Bc. Pavel Skala

Místo této strany bude zadání práce.

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 12. května 2019

Bc. Pavel Skala

Poděkování

Rád bych poděkoval Ing. Romanu Moučkovi, Ph.D. za vstřícnost, trpělivost, odborné rady a cenné připomínky, které mi pomohly tuto diplomovou práci vypracovat.

Abstract

The goal of this Master's thesis is to explore the current tools dealing with neurorehabilitation and subsequently to create an application framework, which will able to provide patients with neurological disorders the possibility of home neurorehabilitation through a series of simple web browser games. The main advantage of this solution is an individual approach to each patient. It allows the patients therapist to tailor the exercises to his specific needs. The first part of this thesis will describe current approaches and solutions to neurorehabilitation. The subsequent part describes the design and implementation of a new solution, which is the goal of this thesis. Last part contains results of testing and evaluation of the created web application.

Abstrakt

Cílem této diplomové práce je prozkoumat současné nástroje zabývající se neurorehabilitací a následně realizovat nový aplikační framework, který umožní osobám s neurologickými problémy možnost domácí neurorehabilitace prostřednictvím sérií jednoduchých her. Hlavní výhodou tohoto řešení bude individuální přístup ke každému pacientovi, kdy každý pacient bude mít k dispozici cvičení vytvořená na míru svým terapeutem. První část této práce bude popisovat současná řešení pro neurorehabilitaci. V další části dojde k navržení a implementaci nového řešení, které bude výstupem této diplomové práce. V poslední části bude zahrnuto testování a vyhodnocení vytvořené webové aplikace.

Obsah

1 Úvod

Neurorehabilitace je multidisciplinární rehabilitační přístup k pacientům s neurologickými problémy. Tito pacienti trpí většinou nejen poškozením motorického systému, ale i poruchy kognitivních funkcí[12].

Neurorehabilitace je složitý a většinou dlouhodobější proces, který je zabezpečován rehabilitačním týmem. Členy týmu by měl být psycholog, neuropsycholog, fyzioterapeut, ergoterapeut, logoped, sociální pracovník a zdravotní sestry.

Pacienti s těžkou disabilitou[1] potřebují pro svoji soběstačnost velmi složité a často finančně náročné kompenzační pomůcky. Většinu těchto pacientů tvoří pacienti s cévními mozkovými příhodami (Dále CMP). Po stabilizaci stavu těchto pacientů by mělo dojít k velmi intenzivní neurorehabilitaci, která ovšem v mnoha případech není možná, protože takových terapeutů je velmi málo. Z porovnání studií [16] zkoumajících různé druhy počítačových her v oblasti neurorehabilitace vyplývá, že konkrétní výsledky velmi závisí jak na použité technologii (pc, tablet, joystick, kinect, Nintendo Wii, atd.), tak na konkrétním úkolu či hře, které pacient provádí. Součástí studie [16] jsou také kontroverzní výsledky, kdy jedna část vykazuje velmi dobré zlepšení pacientova stavu, zatímco druhá žádné zlepšení neprokazuje.

Počítačová neurorehabilitace je ovšem považována za užitečným nástroj pro rehabilitaci, protože u pacientů často dochází ke zvýšení motivace, která je u těchto pacientů minimální, a zapojení se do rehabilitace. Motivaci lze také podpořit zapojením rodinný příslušníků a blízkých přátel, kteří musí pacienta podporovat ke cvičení [15].

Proto je důležité na základě individuálně vytvořeného krátkodobého a dlouhodobého rehabilitačního plánu umožnit pacientům rehabilitovat, a tak jim pomoci o co nejlepší začlenění zpět do běžného života [12]. Existuje studie [22], která pojednává o tom, že tří měsíční trénink vybraných skupin lidí měl na konci tohoto období pozitivní výsledky, které říkají, že pomocí počítačové neurorehabilitace může dojít ke zlepšení neurologických funkcí. Dalším důležitým aspektem je vytvářet cvičení individuálně pro každého pacienta v závislosti na jeho zkušenostech a aktuálním zdravotním stavu. Cíl

[1]Nezpůsobilost některých fyzických, psychických nebo sociálních funkcí a činností vyplývající např. z choroby, závady, poruchy nebo stáří.

konkrétní neurorehabilitace by měl být na jednu stranu dostatečně náročný, ale na druhou stranu nesmí být příliš obtížný, abychom neztratili zájem pacienta o další cvičení. U náročných případů je nejdůležitější ten fakt, že pacient vykonává alespoň nějakou aktivitu.

Právě včasná a důsledná rehabilitace zvyšuje pravděpodobnost zmírnění následků CMP u postižených pacientů[3].

Na základě těchto důvodů jsem se rozhodl vytvořit software, který by umožnil pacientům účinně rehabilitovat a zároveň ušetřil čas terapeutovi. Cílem této diplomové práce je tedy navrhnout a implementovat webovou aplikaci, která dokáže zprostředkovat jejím uživatelům komplexní neurorehabilitační trénink pro snížení dopadu poruchy kognitivních funkcí. Aplikace bude dostupná pomocí internetu, což umožní pacientům provádět cvičení kdykoli, tedy hlavně v rané fázi postižení, kdy je cvičení velmi důležité. V pozdní fázi může cvičení posloužit například jako prevence před oslabení mozkové aktivity.

Abych mohl potřebný software naprogramovat, je nejprve nutné provést analýzu současného stavu trhu s těmito aplikacemi a následně se rozhodnout, které poznatky v práci využít a které nikoli. Tato analýza bude součástí druhé kapitoly.

Další kapitola bude zaměřena na specifikaci požadavků projektu, ve které bude uveden obecný popis aplikace včetně požadovaných funkcí systému.

Před zahájením samotného vývoje bude nutno zvolit potřebné technologie pro vývoj aplikace. Proto součástí čtvrté kapitoly bude analýza dostupných webových technologií, ze které následně zvolím tu nejvhodnější pro zadaný problém.

V páté kapitole provedu návrh aplikace, kde budou uvedeny implementační detaily, které vycházejí ze specifikací požadavků. Následně se budu zabývat samotnou implementací a ověření veškerých funkcionalit.

2 Existující řešení zabývající se neurorehabilitací

V rámci této kapitoly budu popisovat existující aplikační frameworky určené pro dlouhodobou možnost neurorehabilitace. Součástí kapitoly bude také krátký přehled o současné úrovni počítačové neurorehabilitace.

2.1 HAPPYneuron

Jedním z řešení zabývající se stimulací kognitivních funkcí mozku je framework HAPPYneuron. Jedná se o soubor jednoduchých cvičení, které se zaměřují na trénink paměti, koncentrace, logického myšlení, vizuálně-prostorové orientace a řečových funkcí. V těchto pěti oblastech existuje v současné době celkem 49 cvičení, kdy nejpočetněji jsou zastoupeny úlohy z oblasti trénování paměti [1]. Jedná se o placený framework, kde nejlevnější licence stojí 45$ na rok. Na bezplatnou dočasnou verzi mají nárok pouze nově registrovaní uživatelé, a to pouze na dobu jednoho týdne.

Jedná se o částečně personalizovanou formu neuroterapie, kdy se úroveň složitosti jednotlivých cvičení odráží v závislosti na předchozích zkušenostech s úlohami. Výsledky jsou tedy dlouhodobě monitorovány a po provedení určitého počtu cvičení je tak možno analyzovat silné a slabé stránky konkrétního pacienta. Pomocí těchto analýz dochází k sestavení pacientova zdravotního profilu a personalizaci cvičení. Velmi důležitou součástí aplikace je dovednost motivovat pacienty, protože pouze procvičování v dlouhodobém časovém horizontu může přinést pacientovi pozitivní výsledky.

Aplikace je dostupná v několika verzích, kdy základní členění je verze Wellness[1] a Performance[2]. Součástí aplikace je sada několika virtuálních terapeutů, kdy každý z nich má jiný přístup k tréninku, a tudíž záleží na každém z pacientů, kterou cestu si zvolí.

Aplikace není určena pouze pro pacienty, kteří prodělali nějaké onemocnění či léčbu, ale i pro širokou veřejnost, tj. pro sportovce, zaměstnance, starší osoby, studenty a mnoho dalších [2].

[1] Forma wellness programu je určena pro lidi, kteří terapii nutně nepotřebují, ale chtějí svůj mozek uchovat v dobré kondici

[2] Verze Performance je určena pro lidi, kteří chtějí stimulovat svůj mozek mnohem intenzivněji.

2.2 Lumosity

Společnost Lumosity patří mezi nejznámější společnosti zabývající se tréninkem mozku a v současné době mají desítky milionů uživatelů napříč všemi významnými platformami. Na vývoji jejich aplikací se podílí více než stovka osob zabývající se kognitivním výzkumem, vývojářů a vědců z celého světa. Tréninkový program sice umožňuje bezplatný trénink, nicméně nabídne jen velmi omezené množství her oproti placené verzi premium. Nejpopulárnější placená verze je nabízena za 50$ na rok. Právě placená verze přináší uživateli následující možnosti.

- Vyvážený trénink kognitivních funkcí.
- Pomocí inteligentních algoritmů poskytují personalisované cvičení na základě předchozí zkušenosti se cvičeními a osobními preferencemi.
- Poskytuje až 60 her na podporu kognice uživatele.
- Umožňuje uživateli sledovat jeho stav a pokrok.

Tento aplikační framework poskytuje dvě základní kategorie cvičení. První kategorie se skládá ze cvičení poskytující kognitivní trénink, které jsou zaměřeny na rychlost, paměť, pozornost, flexibilitu a řešení problémů. Druhá kategorie se zaměřuje na učení se novým věcem pomocí jazykových, matematických a relaxačních cvičení.

Při samotné registraci jsou po uživateli vyžadovány základní informace, na jejichž základě bude následně docházet k přidělování nových úloh ke cvičení. V tomto dotazníku se sbírají informace typu kdy uživateli vyhovuje provádět cvičení, jak často by chtěl cvičit, kolik hodin denně spí, jeho věku a profese. Společnost tyto informace využije pro účely individuální tréninkového plánu, který se na základě další zkušenosti uživatele se cvičeními vyvíjí.

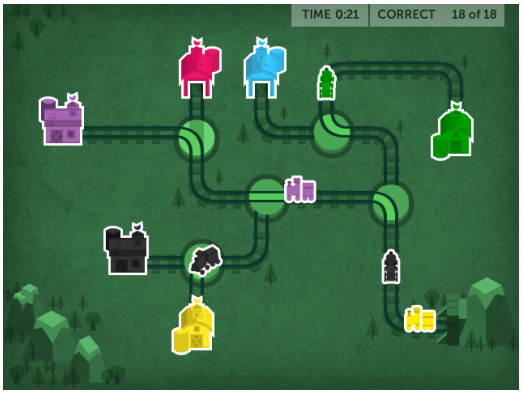
Lumosity podporuje několik světových jazyků, mezi které patří angličtina, němčina, španělština, portugalština, francouzština, japonština a korejština.

2.2.1 Vzorové úlohy

V rámci bezplatné verze jsem mě možnost si vyzkoušet celkem tři úlohy. Dvě z nich v krátkosti shrnu.

Train of Though

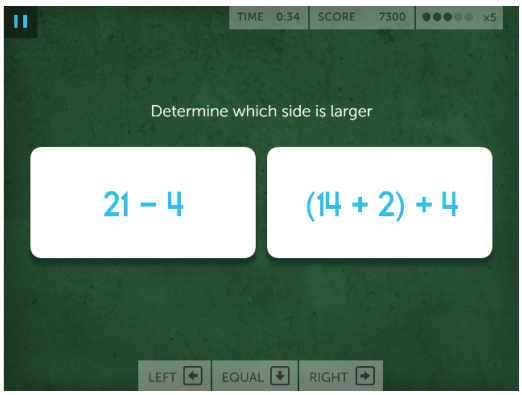
Jedná se o cvičení zaměřené na rozdělenou pozornost viz obr. 2.1. Podstatou úlohy je pomocí klikání na výhybky dopravit vlaky do cílových stanic označených stejnou barvou. Na základě úrovni pacienta dochází k volbě počtu cílových stanic, jejichž maximální množství je 14. Celková doba trvání cvičení je omezena na dvě minuty a počítá se úspěšnost dojetí vlaků do správných stanic.



Obrázek 2.1: Úloha zaměřená na rozdělenou pozornost

Chalkboard challenge

Jedná se o matematickou úlohu, ve které se pacientovi zobrazují dva jednoduché matematické výrazy a uživatel musí rozhodnout, který z nich je větší, nebo zda jsou výsledky shodné (viz obr. 2.2). Pacient má na úlohu opět dvě minuty, ovšem při správném rozhodování může získat časový bonus. V opačném případě o čas přichází.



Obrázek 2.2: Úloha zaměřená na matematiku

2.3 NP3

Dalším řešením pro domácí i hospitalizovanou neurorehabilitaci je soubor programů označovaných jako systém NP3. Tento systém byl vyvinutý a otestovaný týmem neuropsychologů, klinicky pracujících v oboru neuropsychologické rehabilitace. Programy a jejich jednotlivá cvičení reflektují několikaroční klinické zkušenosti s diagnostikou a terapií neurologicky postihnutých pacientů. Pro lepší orientaci v uváděných pojmech doporučuji shlédnout tabulku **??**.

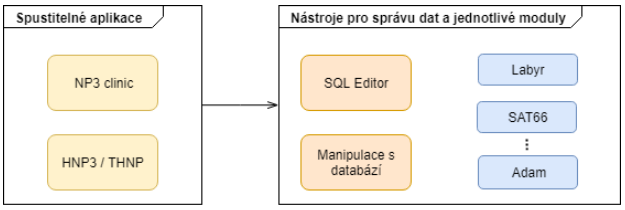
Pojem	**Význam**
Modul	Jedná se o program, který je používám prostřednictvím hlavní aplikace, a umožňuje spouštět jednotlivé úlohy. Součástí modulu bývá i funkcionalita pro analýzu výsledků úloh pomocí grafů. Pod tímto pojmem může také vystupovat SQL editor, který slouží pro analýzu dat uložených v databázi.
Úloha	Textově zadaný předpis, který se následně předá modulu, čímž dojde ke spuštění úlohy pacientovi.
Cvičení	Textově zadaný předpis, který obsahuje seznam úloh pro procvičení. Bez tohoto předpisu není pacientovi umožněno provádět neurorehabilitaci. Místo pojmu cvičení se také může použít termín „testovací skript“.

Tabulka 2.1: Vymezení základních pojmů

Celé řešení se skládá z několika samostatně spustitelných desktopových programů určených pro neurorehabilitaci. Tyto programy jsou dále rozšiřovány programy jednotlivých cvičení a administrativními programy pro analýzu dat, jejichž strukturu můžete vidět na obrázku 2.3. Jednotlivá cvičení jsou tedy realizována samostatnými programy, které jsou spouštěny prostřednictvím hlavních programů, a obsahují veškerou logiku potřebnou pro cvičení. Mezi tuto funkcionalitu patří spouštění cvičení a následná analýza výsledků. Mezi analytické programy patří především SQL Editor pro získávání a analýzu dat z databáze Firebird[3], která musí být na počítači nutně nainstalována. Bez databáze nelze programy spouštět [15].

Současná verze desktopové aplikace má název NP3 a navazuje na své původní dvě verze, tj. `Neurop-1` a `Neurop-2`. Aplikace má tedy svoji historii a při jejím vývoji docházelo ke stále větší snaze zacílení na co nejširší pole

[3]Firebird je multiplatformní relační databáze, která je dostupná zdarma.



Obrázek 2.3: Struktura programů v rámci systému NP3

uživatelů - pacientů. Aktuální programové řešení bylo vydáno v roce 2015 a obsahuje 55 programových modulů[4]. NP3 podporuje rychlé a efektivní vytváření libovolně obsáhlých skupin úloh a cvičení. Úlohy není možné zadávat samostatně, ale pouze jako jednotlivá cvičení, tj. jedno cvičení se skládá z několika úloh. Takto vytvořeným cvičením se také říká skripty. Tyto skripty řídí automatické zobrazení veškerých úloh a zápis výsledků. Skripty lze v budoucnosti opětovně používat. Veškeré výsledky ze cvičení jsou ukládány do databáze `Firebird`, díky čemuž je terapeutovi umožněno provádět nad získanými výsledky pacientů libovolné dotazy pomocí jazyka SQL[5].

2.3.1 Vzorové úlohy

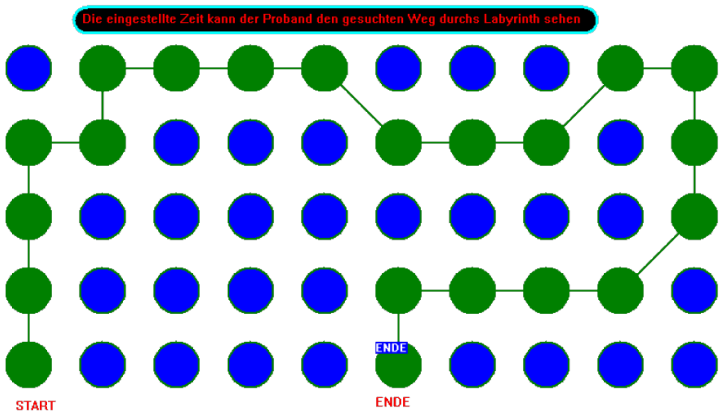
V poslední podkapitole si představím pár úloh, které jsou součástí řešení NP3, abychom si udělali představu o jejich náročnosti i grafické prezentaci. Úlohy jako takové nesmí obsahovat žádné náročné grafické prvky, které by odváděly pacienta od pozornosti. Takové úlohy by pro pacienta byly složité a jejich výsledky velmi zkreslené.

Labyr

Jedná se o jednoduchou úlohu, kdy má pacient za úkol projít bludištěm z jednoho místa do druhého. Úloha má několik vstupních parametrů jako jsou například velikost hrací plochy, doba prezentace správného řešení před započetím hry, viditelnost prvního a posledního bodu, času pro splnění úlohy a mnoho dalších. Grafickou podobu úlohy Labyr můžete vidět na obrázku 2.4.

[4]Pod pojmem programový modul si lze představit jeden konkrétní typ úlohy, tj. jeden konkrétní program, který lze spustit pomocí hlavního spustitelného programu.

[5]SQL je zkratka pro strukturovaných dotazovací jazyk, který je používám pro práci s daty v relačních databázích.



Obrázek 2.4: Ukázka úlohy z programu Labyr

Adam

Dalším typem úloh jsou doplňovací úlohy. V tomto případě se jedná o doplňování slov do vět, kdy na obrázku 2.5 má pacient za úkol doplňovat do vět hlavní města.

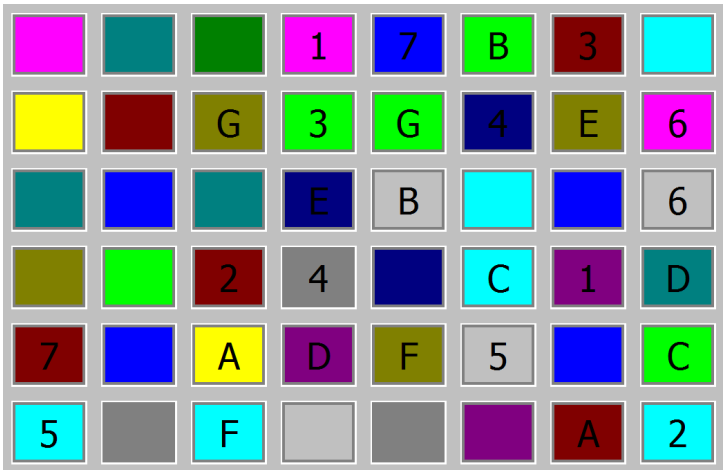
Obrázek 2.5: Ukázka z úlohy Adam se zaměřením na hlavní města

Refind

Poslední typ úloh, které si zde představíme, je Refind. Jedná se o úlohu, kde má pacient za úkol řadit vzestupně či sestupně čísla a zároveň písmena ve stylu 1-A, 2-B, 3-C atd. Možné modifikace úlohy jsou takové, že jedna posloupnost může být vzestupně, jiná sestupně a začínat se nutně nemusí od jedné, nebo od písmena A. Grafické znázornění úlohy můžete vidět na obrázku 2.6.

2.4 NP3 - webová aplikace

Před třemi lety, tedy v roce 2015, vznikal projekt, který měl za úkol převést stávající desktopové řešení nástroje NP3 do podoby webové aplikace. Na



Obrázek 2.6: Ukázka z úlohy Refind

vývoji se podílel tým německých studentů na tamější univerzitě, kteří ovšem aplikaci nedokončili a její vývoj tím skončil. Jelikož se tato diplomová práce zabývá také vývojem webové aplikace a mám k dispozici původní zdrojové kódy, tak jsem se rozhodl mezi existující nástroje uvést i toho řešení.

Analýza se bude opírat převážně o zdrojové kódy, které nejsou okomentované, inicializační skript databáze spolu s jejím ER-modelem[6] a textovým dokumentem, ve které je popsáno jakým způsobem se do aplikace přihlásit. Další informace včetně dokumentace k tomuto řešení bohužel neexistují.

Na základě této analýzy se pokusím získat co nejvíce užitečných informací, nad kterými postavím prototyp modelu mého řešení webové aplikace, tj. výstup této diplomové práce.

2.5 Analýza a zhodnocení existujících řešení

První dvě řešení jsou mezi uživateli velmi populární. Jak HAPPYneuron tak Lumosity nabízejí velké množství programů, které jsou komplexně zaměřeny na rehabilitaci kognitivních funkcí pacientů. Nevýhodou těchto aplikací je to, že se pouze snaží o co nejdůvěrnější napodobení práce skutečného terapeuta množinou algoritmů, které zpracovávají výsledky pacientů a navrhují další cvičení.

Takové řešení je vhodné zejména pro takové typy pacientů, kteří si touho neuroterapií chtějí pouze procvičovat mozek, ale svým způsobem netrpí nijak závažným onemocněním. Pro pacienty, kteří prodělali například cévní

[6]Entity-Relationship model – entitně vztahový model se v softwarovém inženýrství používá pro popis objeků a vztahů mezi nimi.

mozkovou mrtvici by bylo plnění takových cvičení velmi obtížné či dokonce nemožné. Takoví pacienti potřebují absolutní individuální péči a jednotlivé úlohy jim musí být individuálně připraveny.

Proto jsme se rozhodl podrobně zanalyzovat jak desktopové, tak webové řešení systému NP3, které přinášejí zejména výhodu naprosto individuální přístupu k pacientům, který je v případě vážných neurologických problémů klíčový. Následně bych navrhl novou webovou aplikaci, kterou by mohli jak pacienti, tak terapeuti používat kdekoli. Terapeuti tak budou mít možnost bez fyzického kontaktu provádět individuální rehabilitaci, což jim ušetří mnoho času. Právě čas je v současné důsledné neurorehabilitaci problémem - pacientů je mnoho a kvalitních neuroterapeutů nedostatek. Pacienti tak budou moci rehabilitovat kdykoli bude potřeba a hlavně s minimální námahou.

2.5.1 Analýza desktopové verze NP3

Jednou z částí zadání je analýza právě tohoto řešení, na jejíž základě dojde k návrhu nové webové aplikace, která tak umožní využívat lety ověřenou neurorehabilitaci mnohem většímu množství lidí z pohodlí domova bez nutnosti instalace programu na jejich zařízení. Další výhodu, kterou webová verze přinese, je zlepšení odezvy mezi terapeutem a pacientem, který tak bude moci pružněji reagovat na pacientovu aktivitu.

Struktura modulů

Většina modulů umožňuje tvorbu vlastních úloh, které mají většinou textovou nebo obrazovou podobu, čímž je můžeme přizpůsobit zájmům a schopnostem pacienta[7]. Než začnu popisovat detaily, chtěl bych opět připomenout několik termínů, které se v textu budou vyskytovat viz tab. **??**.

Všechny moduly umožňují automatické grafické zobrazení výsledků a několik typů analýz, mezi které patří především *časové*, *chybové* a *kvantitativní*. V programu NP3 je implementován *prohlížeč*, který umožňuje dokumentaci relevantních údajů. Pokud si přejeme provádět sofistikovanější analýzy uživatelských výsledků, pak můžeme použít integrovaný *SQL-editor*.

[7]Personalizace úloh schopnostem a zájmům pacienta je v oblasti neurorehabilitace velmi důležitá, neboť tito pacienti ve většině případů postrádají vnitřní motivaci ke cvičení. Proto je naprosto klíčová podpora ze strany rodinných příslušníků.

Používání programu NP3

Celkové programové řešení se skládá z několika samostatných programů, které se dělí do dvou základních kategorií (viz obr. 2.3). První kategorie tvoří spustitelné programy, které jsou určeny pro pacienty a terapeuty. Do druhé skupiny patří programy, resp. moduly, které se spouští prostřednictvím programů z první skupiny.

Pokud chce terapeut umožnit pacientovi procvičovat úlohu konkrétního typu, musí být splněny dvě základní podmínky. První podmínkou je ta, že pacient musí mít na svém počítači přítomný příslušný modul (např. Labyr viz obr. 2.3 vpravo nahoře). Druhou podmínkou je, že úloha musí být součástí cvičení, které pacient od terapeuta obdrží. Bez cvičení s úlohami není pacientovi umožněno spustit žádný program.

V druhé části této kapitoly zjednodušeně popíši výše uvedené programy, aby měl čtenář povědomí o tom, jak celý systém funguje.

NP3 clinic

Jedná se o program, který je určen pro pacienty, kteří jsou přítomni na klinice. Klient získá přístup k programu tím způsobem, že obdrží svoje heslo, kterým se do programu přihlásí[8].

Po úspěšném přihlášení do programu se pacientovi zobrazí jemu přiřazené balíky úloh. Každý balík se skládá z několika úloh, které si může následně spustit. Z každého cvičení se zaznamenávají klíčové informace, které se následně ukládají do databáze pro následnou analýzu. Tato data mohou mít podobu úspěšnosti v dané úloze, času potřebného pro splnění úlohy, počtu chyb v úloze apod.

Počet úloh v jednotlivých balících není omezen a jsou tvořeny individuálně pro potřeby pacienta. Supervizor tvořící cvičení může jednoduše tvořit nové kombinace úloh v balících v závislosti na vyvíjejícím se stavu pacienta. Součástí programu je také jednoduchá zpětná vazba mezi terapeutem a pacientem v textové podobě.

THNP/HNP3

Druhým řešením je rehabilitace pomocí nástrojů `THNP/HNP3`, které jsou určeny pro používání mimo kliniku. Program byl vyvinut především pro pacienty, kteří nemají možnost se účastnit léčby ve specializovaném léčebném

[8]Přihlášení do aplikace je řešeno pouhým zadáním hesla, tj. pacient nepotřebuje přihlašovací jméno.

zařízení, ale i přes to chtějí používat tento program. Z názvu je patrné, že se jedná o dva programové moduly, kdy každý cílí na jinou skupinu lidí.

1. **HNP3**
   Tento modul je tvořen programy, které používá pacient. Programy dovolí uživateli pouze vybrat cvičení, které bude vykonávat, a ukončit program. Součástí programu je ještě odeslání údajů o jednotlivých cvičeních terapeutovi. Dále umožňuje textovou komunikaci s terapeutem.

2. **THNP**
   Jedná se o sbírku programů určené pro terapeuta, který přes tyto programy vytváří individuálně pro pacienty jejich sady úloh, neboli cvičení[9]. Údaje těchto cvičení jsou kódované a předané pacientovi prostřednictvím nějakého média. Bez tohoto média není možno cvičené programy používat.

   Po každé periodě, zpravidla mezi 6 až 8 týdny, musí pacient předat toto médium zpět terapeutovi, který získaná data vyhodnotí a na jejich základě určí další průběh terapie. Prakticky jde o to, že mezi těmito periodami se data ukládají do databáze. Po skočení těchto období se data z databáze překopírují na médium, které se předá terapeutovi k analýze.

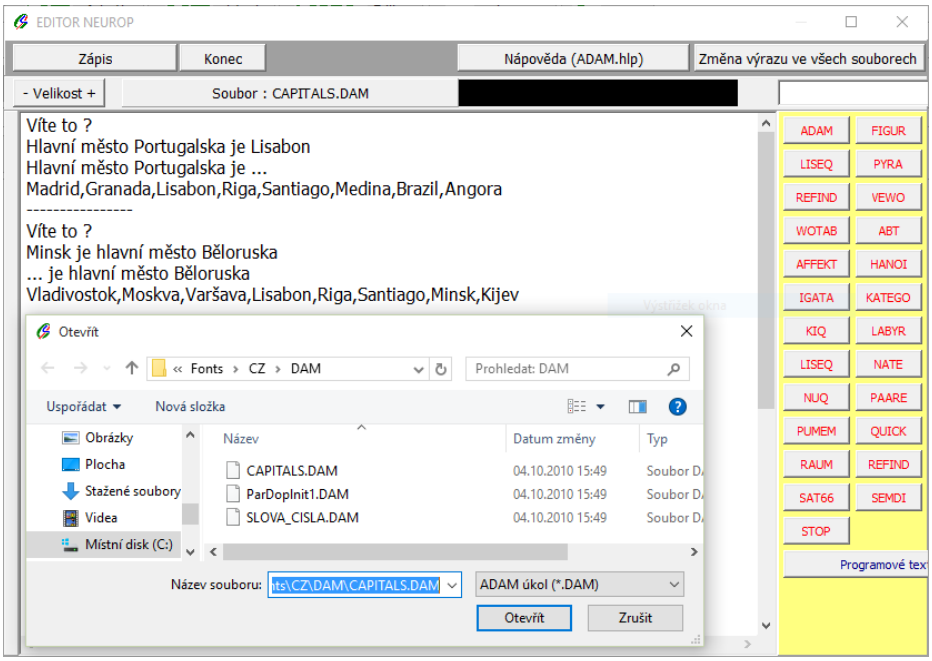
Nástroj `THNP/HNP3` poskytuje několik funkcí, kterým bude věnován zbytek této kapitoly. Nejprve si představíme množinu funkcí u nástroje `THNP` určené pro terapeuty.

1. **Tvorba nových úloh**
   Předtím, než pacientům umožníme spustit libovolné cvičení, je nutné vytvořit úlohy, ze kterých cvičení následně sestavíme. Vytvoření úlohy probíhá v editoru, který se skládá ze dvou částí. První část tvoří textové pole, kde dochází k vyplňování předpisu daného programového modulu. Ve druhé části se nalézá seznam všech dostupných programových modulů, kdy po kliknutí na libovolný z nich máme možnost vybrat jednu z již vytvořených úloh, následně ji editovat a uložit jako novou úlohu[10]. Pro lepší představu o podobě takového editoru se můžete podívat na obrázek 2.7.

   Z takto vytvořených úloh následně vytvoříme cvičení, které může být spouštěno pacienty.

[9] Cvičení je ekvivalentní výraz pro skript

[10] Principiálně máme tedy dvě možnosti vytvoření úloh. První, jednodušší, možnost je editovat již existující. Druhá o poznání složitější je editovat celé cvičení ručním vkládáním textu, což předpokládá perfektní znalost předpisu dané úlohy.



Obrázek 2.7: Ukázka editoru pro vytváření nových úloh.

2. **Tvorba nových cvičení**
   Nové cvičení, nebo-li skript, určuje obsah, obtížnost, pořadí a počet úloh. Cvičení obsahuje několik úloh, které jsou pacientovi předány na cvičném médiu. První řádek skriptu určuje dostupnost cvičení pro pacienty, respektive uvádí, kterým pacientům je toto cvičení nedostupné. Je to realizováno klíčovým slovem „lock“, za kterým následuje libovolný text[11]. Pokud tento text má pacient nastaven ve svém uživatelském účtu (nastavuje terapeut), pak se mu toto cvičení nezobrazí mezi seznamem dostupných cvičení.

   Editor pro sestavování nových cvičení se skládá ze dvou oken a několika tlačítek. První okno tvoří textové pole, ve kterém dochází k sestavování cvičení, které lze realizovat buď ručním psaním, nebo klikáním do druhého okna. V druhém okně je seznam všech programových modulů. Po kliknutí na požadovaný programový modul dojde k zobrazení nabídky všech úloh, které byly v rámci tohoto programového modulu vytvořeny. Potvrzením výběru konkrétní úlohy dojde k automatickému vložení záznamu do textového pole. Tento elementární záznam je tvořen názvem programového modulu, pořadovým číslem úlohy z daného modulu a názvem úlohy, která byla z tohoto progra-

[11] Ukázka, jak zamezit uživateli přístup k úloze značené textem „abc“: `lock: abc`

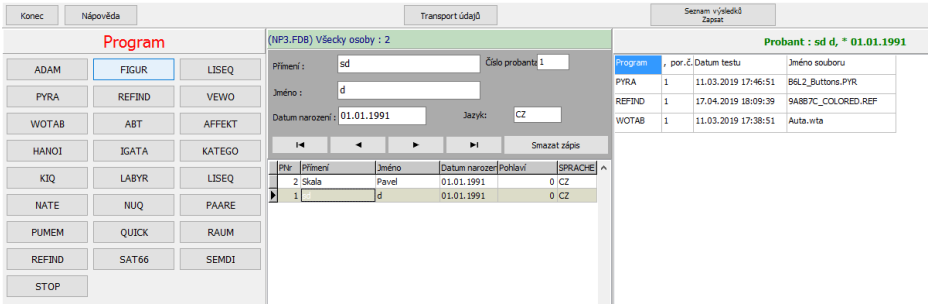
mového modulu vytvořena[12]. Vzhled tohoto editoru se prakticky neliší od editoru pro vytváření nových úloh (viz obr. 2.7).

Tímto způsobem se sestaví cvičení, které se následně uloží. Přiřazení funguje pomocí prvního parametru cvičení, tedy nastavením parametru „lock“.

3. **Prohlížení údajů**
   Prostřednictvím programu pro zobrazení údajů z databáze je terapeutovi umožněno zobrazit následující informace:

   - výsledky jednotlivých pacientů
   - výpis dlouhodobého průběhu veškerých výsledků pacienta v rámci jednotlivých cvičení.

   V programu pro zobrazení výsledků jsou tři základní okna (viz obr. 2.8). V prvním okně je seznam všech dostupných programových mo-



Obrázek 2.8: Ukázka editoru náhled výsledků.

dulů. Druhý obsahuje seznam všech pacientů a třetí je určen pro zobrazení dokončených úloh, odpovídajících zadaným kritériím. Tato kritéria se zavádějí buď kliknutím na programový modul, nebo kliknutím na konkrétního pacienta. Následně se ve třetím okně zobrazí odpovídající dokončené úlohy (jejichž výsledky jsou uloženy v databázi) a kliknutím na libovolnou z nich se zobrazí informace o jejich výsledku. Dále je možnost tyto informace promítnout do grafu.

Pro získání dalších výsledků je zde připravený `SQL-modul` pro přímou komunikaci s databází. Tímto jsme schopni získat mnohem sofistikovanější data.

Dále si představíme funkce, které jsou k dispozici v rámci nástroje `HNP3`, který je určený pacientům pro domácí terapii.

[12]Takový záznam může mít následující podobu: `PRG=HANOI 1, Hanoi_uloha`.

1. **Spuštění připravených cvičení s úlohami**
   Po obdržení paměťového média s cvičeními a přihlášení do programu lze spustit připravené úlohy. Tato operace musí být přístupná s minimálními nároky na obsluhu počítače, protože bude prováděna lidmi s neurologickými poruchami. Po spuštění a splnění zadaného cvičení jsou výsledky uloženy do databáze. Následně má pacient možnost buď spustit další cvičení, nebo ukončit program.

2. **Zasílání zprávy a/nebo údajů terapeutovi**
   Jedná se o jednoduchý komunikační editor, kterým má pacient možnost zaslat zpětnou vazbu týkající se provedených cvičení. V rámci tohoto editoru je možné vložit jednoduchou zprávu, či vyplnit dotazník týkající se náročnosti provedeného cvičení.

2.5.2 Analýza webové verze NP3

Před spuštěním webové aplikace bylo nejprve nutno vytvořit databázi prostřednictvím nástroje MS SQL Server Management Studio 2014[13], kde jsem pomocí inicializačního skriptu vytvořil celou databázi. Následně jsem si otevřel projekt ve vývojovém prostředí MS Visual Studio 2017[14] a nastavil v konfiguračním souboru parametr pro napojení do vytvořené databáze. Po těchto úpravách se mi podařilo aplikaci spustit. Takto spuštěná aplikace umožňovala pouze prezentaci celého produktu bez možnosti spuštění libovolné úlohy, protože přihlášení uživatele bylo nefunkční. Po několika úpravách v databázi se mi podařilo přihlásit, což mi umožnilo spustit celkem 5 typů úloh, z nichž každá obsahovala několik cvičení.

Po spuštění několika úloh jsem zjistil, že tímto veškerá funkcionalita programu končí, protože mi nebyly umožněny provést další operace v aplikaci. Mezi těmito operacemi bych především očekával náhled výsledků jednotlivých pacientů, možnost tvorby nových cvičení nebo náhled na pacienty. Po náhledu do databáze jsem zjistil, že úlohy generovaly mezivýsledky.

Analýza zdrojových kódů

V minulé kapitole jsem naznačil, že popisované řešení nebylo příliš funkční, a tak jsem se rozhodl pro důslednou analýzu zdrojového kódu. Zde jsme zjistil, že většina požadované funkcionality, která byla součástí desktopové aplikace, byla implementována, ovšem po spuštění aplikace v ladícím módu

[13] dále jen MSSMS
[14] Dále jen VS

program vracel velké množství výjimek způsobené zejména při získávání dat z databáze.

Z toho plyne, že inicializační skript, který byl s aplikací dodán, nedokáže vygenerovat korektní podobu databáze. Skript pravděpodobně nebyl dodán v jeho konečné podobě. I přesto databáze obsahovala nějaká data.

Většina zdrojového kódu má spíše prezentační charakter. Jednotlivé spustitelné úlohy jsou vytvářeny následujícím scénářem:

1. Vytvoření HTML šablony pro spuštění úlohy.
2. Vytvoření HTML šablony pro prezentaci výsledků dané úlohy.
3. Souborem napsaným v JavaScriptu pro vykreslení dané úlohy a následnému odeslání výsledku na server pro uložení dat do databáze.

V původním řešení webové aplikace bylo naimplementováno celkem pět programových modulů v jazyce JavaScript. Jelikož se jedná o programy, které jsou součástí desktopové verze, pak jsem se je rozhodl do mého řešení také zařadit, ovšem v upravené podobě.

3 Specifikace požadavků

3.1 Úvod

V rámci této kapitoly si představíme veškeré požadavky na aplikaci, které budou výstupem této diplomové práce.

3.1.1 Předmět specifikace

Výsledkem práce bude webová aplikace, která bude sloužit pro neurorehabilitaci pacientům, kteří si prošli závažnými neurologickými problémy, aby jim byl umožněn co nejsnazší přechod do běžného života. Součástí aplikace bude možnost vytvářet nové programy, ze kterých se budou následně tvořit testovací úlohy určené na míru každému pacientovi s následným vyhodnocením. Pacient tak bude mít k dispozici nástroj, díky kterému bude moci rehabilitovat kdykoli bez omezení a nutnosti přítomnosti terapeuta. Terapeut bude zase moci kdykoli vytvářet nové úlohy, resp. je vyhodnocovat a posílat tak svým pacientům odezvu v podobě krátkých reportů o pacientovo pokroku, čímž bude docházet i k motivaci pacienta do dalších cvičení.

Součástí této aplikace bude také webová prezentace, jejímž cílem bude představit tuto aplikaci potenciálním zákazníkům sérií několika málo úloh, které si budou moci zdarma vyzkoušet.

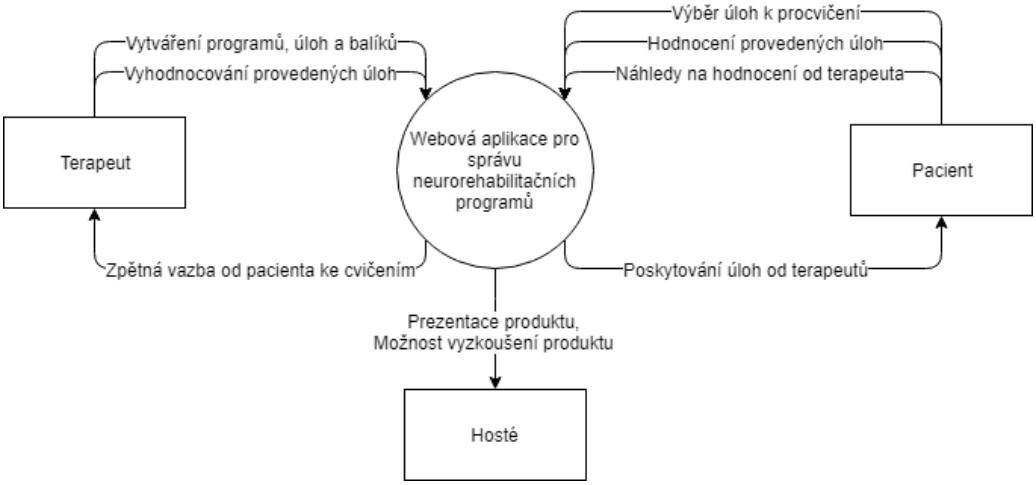
3.2 Obecný popis

3.2.1 Kontext systému

Dosud existovala pouze sada desktopových programů, které byly schopny vytvářet a interpretovat sadu úloh vytvořených terapeutem. Dalším krokem vývoje této aplikace je ji předělat do podoby webové, která bude nasazena na serveru, a tak přístupná i široké veřejnosti. Kontextový diagram systému můžete vidět na obrázku 3.1.

3.2.2 Funkce produktu

Po analýze současných nástrojů na počítačovou neurorehabilitaci jsem se rozhodl v rámci této práce naimplementovat několik programových modulů, které budou zajišťovat hlavně správu programů, úloh a balíků. Dále bude



Obrázek 3.1: Kontextový diagram systému

nutné naimplementovat administrační modul, který bude spravovat uživatele v aplikaci, a modul pro zpracovávání a report výsledků. Pro zajištění vícejazyčného použití jednotlivých programů bude také naimplementován modul překladů, který bude spravovat lokalizace podporovaných jazyků. Pro lepší orientaci v terminologii se doporučuji podívat na tabulku 3.1).

Pojem	**Význam**
Program	Jedná se o tzv. šablonu, ze které se následně mohou vytvářet jednotlivé úlohy. Při vytváření programu dochází k definování této šablony.
Úloha	Každá úloha musí být vytvořena na základě programu. Při procesu vytváření nové úlohy tedy dochází k definování konkrétní množiny hodnot.
Balík / cvičení	Jedná se o pouhý kontejner, do kterého jsou seskupovány jednotlivé úlohy. Pacient si následně spouští celý balík, nikoli samostatné úlohy.

Tabulka 3.1: Vymezení základních pojmů ve webové aplikaci

3.2.3 Třídy uživatelů

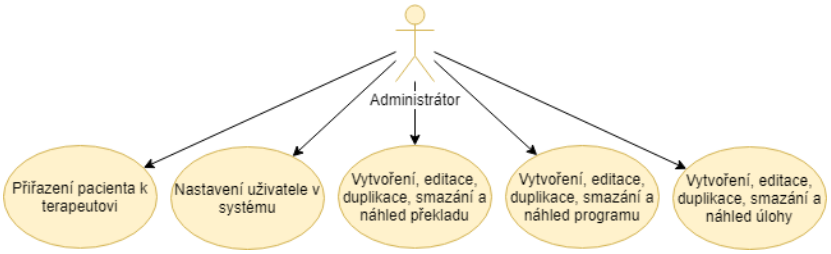
V aplikaci budou uživatelé vystupovat celkem v šesti rolích. Tyto role jsou Super-administrátor, administrátor, super-terapeut, terapeut, pacient a uživatel bez role. Všechny role, jejich účel a hlavní vlastnosti si představíme na následujících řádkách.

Super-administrátor

Uživateli s touto rolí je umožněno využívat veškeré funkcionality, kterou tato webová aplikace nabízí, tj. tento uživatel disponuje právy všech dalších rolí. Tato role je určena především pro vývojáře a uživatele, kteří potřebují veškerou funkcionalitou aplikace bez nutnosti opakovaného přihlašování a odhlašování.

Administrátor

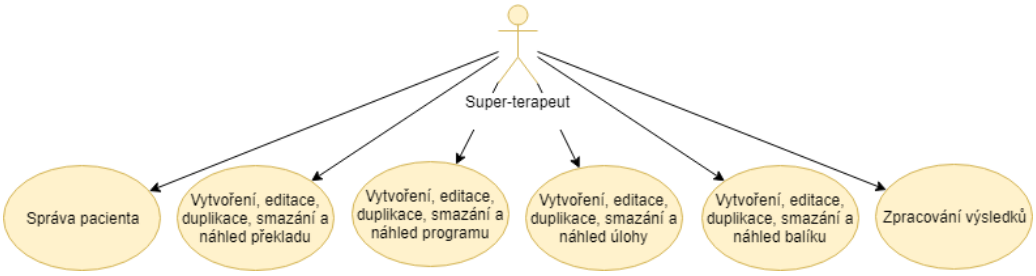
Jedná se o uživatele, který se stará o nastavení prostředí této aplikace a dále má možnost vytvářet nové programy a úlohy, čímž může přispívat k rozšiřování celé aplikace. Zjednodušený diagram užití můžete vidět na obrázku 3.2.



Obrázek 3.2: Zjednodušený diagram užití pro administrátora.

Super-terapeut

Jedná se o roli, která je vytvořena kvůli odlišení základního terapeuta, který nemá povoleno vytvářet nové programy, resp. tuto operace neumí. Tato role může provádět operace zobrazené na diagramu 3.3.

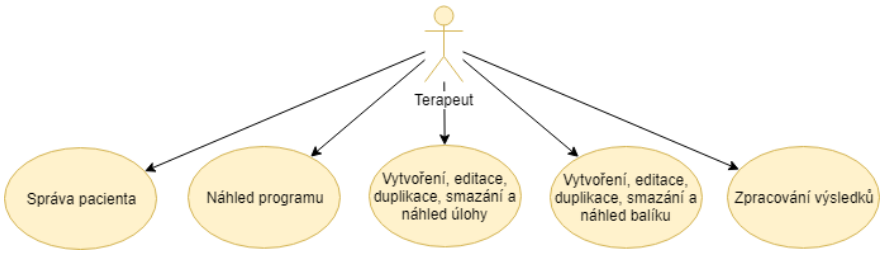


Obrázek 3.3: Zjednodušený diagram užití pro super-terapeuta.

Terapeut

Jedná se o zjednodušenou formu super-terapeuta, který nemá možnost vytvářet nové programy, ovšem má umožněno úlohy vytvářet. Každý terapeut může přiřazovat svým pacientům buď své vlastní úlohy, nebo úlohy, které jsou označeny jako veřejné. Tento příznak nastavuje vždy autor úlohy, čímž umožňuje její využití všemi ostatními terapeuty.

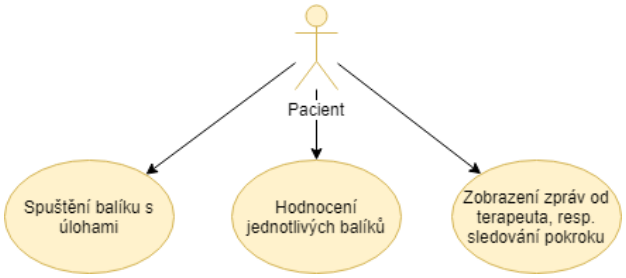
Veškeré operace, které má terapeut povolené, jsou k vidění ve zjednodušeném diagramu 3.4.



Obrázek 3.4: Zjednodušený diagram užití pro terapeuta

Pacient

Jedná se o roli, pro kterou je tato aplikace výhradně určena, ovšem její funkcionalita je velmi omezena. Je to jedním z hlavních požadavků na aplikaci, protože její používání musí být pro pacienta co nejjednodušší. Tyto funkce jsou popsané ve zjednodušeném diagramu 3.5.



Obrázek 3.5: Zjednodušený diagram užití pro roli pacient.

Nepřiřazeno

Tuto roli získá každý nově registrovaný uživatel. Po každé registraci dojde k jejímu schválení administrátorem, který následně novému uživateli přidělí jeho roli.

3.2.4 Uživatelská dokumentace

Součástí diplomové práce je také uživatelská dokumentace, která je součástí přílohy (viz str. 108).

3.3 Funkce systému

V této podkapitole si představíme prostřednictvím uživatelských případů užití veškerou funkcionalitu, kterou tato aplikace bude mít. U jednotlivých scénářů budu popisovat jejich ideální průběh včetně zúčastněných rolí, počátečních i následných podmínek, alternativních i chybových stavů.

Již dříve jsme si představili role super-terapeuta a terapeuta. Pokud budu v textu zmiňovat roli terapeuta, zároveň se tím myslí i role super-terapeuta. Opačně toto tvrzení neplatí. Zároveň jsme si představili roli super-administrátora, který má možnost provádět všechny operace v systému. Z tohoto důvodu je zbytečné jej v jednotlivých scénářích zmiňovat, protože tyto operace mu jsou přístupné.

Jelikož možných scénářů je v této práci velké množství, rozhodl jsem se v této kapitole detailně popsat pouze ty nejdůležitější a u zbývajících uvést pouze jejich popis. Detailní popis všech scénářů bude součástí přílohy.

3.3.1 Registrace uživatele (UC01)

Popis

Každý nově příchozí uživatel má možnost se registrovat do systému pomocí registračního formuláře. Jedná se o základní funkcionalitu, která umožní potenciálním zákazníkům využívat tuto aplikaci.

Standardní průběh

1. Uživatel klikne na tlačítko registrace v pravém horním rohu aplikace.
2. Vyplní všechny povinné údaje ve formuláři a klikne na tlačítko registrovat.
3. Nový uživatel obdrží email, ve kterém je žádán o ověření své emailové adresy.
4. Upozornění administrátora aplikace o novém uživateli v systému prostřednictvím emailu, kvůli nastavení příslušné role.

Zúčastněné role

- Host

Vstupní podmínky

- žádné

Následná podmínka

- Uživatel získá účet v aplikaci

3.3.2 Přihlášení uživatele (UC02)

Popis

Každý zaregistrovaný uživatel má možnost přihlášení se do systému pomocí svých identifikačních údajů.

Standardní průběh

1. Uživatel klikne na tlačítko „přihlášení“ v pravém horním rohu aplikace.
2. Vyplní přihlašovací formulář a klikne na tlačítko „přihlásit se“.
    - Chyba 2a – Byly zadány nesprávné přihlašovací údaje.
3. Uživateli se zobrazí jeho domovská obrazovka.

Zúčastněné role

- Administrátor, terapeut, pacient, nepřiřazeno

Vstupní podmínky

- Uživatel má vytvořený účet.

Následná podmínka

- Uživatel je přihlášen do systému.

Chybový průběh

- 2a – Uživatel zadal nevalidní kombinaci uživatelského jména a hesla, a tak zůstane v přihlašovacím okně a bude o chybě informován. Pokud uživatel své přihlašovací údaje zapomněl, má možnost své heslo obnovit pomocí tlačítka „Zapomenuté heslo“ v dolní části přihlašovacího formuláře.

3.3.3 Přiřazení pacienta k terapeutovi (UC03)

Popis

Aby mohlo dojít k zahájení terapie, musí být nejdříve vytvořena vazba mezi terapeutem a pacientem. Vazba se bude vytvářet výběrem uživatelů ze seznamů a následným potvrzením akce. Po vytvoření vazby je terapeutovi umožněno přidělovat svým pacientům balíky s úlohami.

Standardní průběh

1. Uživatel klikne na kartu „Uživatelé“ v horním menu a následně na položku „Přiřazení pacientů“.
2. Následně se zobrazí seznam existujících přiřazení včetně možnosti přidání nového přiřazení kliknutím na tlačítko „Nové přiřazení pacienta“.
    - Alternativa 2a: Seznam existujících přiřazení může být prázdný.
3. Poté se zobrazí dialogové okno s dvěma rozbalovacími seznamy. V prvním seznamu se vybere terapeut a ve druhém pacient.
    - Alternativa 3a: Některý ze seznamu je prázdný.
4. Potvrzením této akce na tlačítko „Uložit“ dojde k vytvoření nové dvojice terapeut-pacient.

Zúčastněné role

- Administrátor

Vstupní podmínky

- Uživatel musí být přihlášený (viz UC02)

Následná podmínka

- Bude vytvořena vazba mezi terapeutem a pacientem.

Alternativní průběh

- 2a – Seznam může být prázdný z důvodu dosud neexistujícího přiřazení, nebo aktuálně nastaveného filtru.
- 3a – Seznam může být prázdný pouze v případě neexistence uživatele s danou rolí. V tomto případě případ užití končí.

3.3.4 Zobrazení seznamu programů (UC12)

Popis

Uživatelé s příslušnými rolemi mají možnost si zobrazit seznam programů, které sami vytvořili nebo jsou nastaveny jako veřejné.

Standardní průběh

1. Uživatel klikne v menu na kartu „Programy“, čímž se mu rozbalí seznam, a následně klikne na položku „Správa programů“.
2. Následně dojde k zobrazení seznamu uživatelovi dostupných programů.
    - Alternativa 2a: Seznam je prázdný.

Zúčastněné role

- Administrátor, super-terapeut, terapeut

Vstupní podmínky

- Uživatel musí být přihlášený (UC02).

Následná podmínka

- Uživateli se zobrazí seznam programů, ke kterým má přístup.

Alternativní průběh

- 2a – Seznam může být prázdný ze dvou důvodů. Prvním důvodem může být, že danému filtru neodpovídá žádná úloha. Druhým, že uživatel dosud nevytvořil žádnou úlohu, nebo neexistuje úloha, která by byla veřejná. V tomto případě dojde k ukončení případu užití.

3.3.5 Editace programu (UC14)

Popis

Jedná se o uživatelskou akci, při které dochází k nadefinování prototypu programu, ze kterého budou následně vytvářeny úlohy pro pacienty. Základem pro vytvoření prototypu je nadefinování jeho vstupních parametrů, pomocí kterých dojde ke spuštění úlohy, a výstupních parametrů, které se budou z daného programu ukládat a budou sloužit pro analýzu výsledků. Program bude mít dále několik metadat, například jeho název a krátký popis.

Standardní průběh

1. Uživatel si zobrazí seznam dostupných programů (viz UC12).
2. Z tohoto seznamu si uživatel vybere program, který chce editovat, a klikne na tlačítko „Editace“.
    - Alternativa 2a – U požadovaného programu není tlačítko „Editace“
3. Následně je uživatel přesměrován do editoru programu.
4. Uživatel provede potřebné změny a kliknutím na tlačítko „Uložit“ aktualizuje daný program.

Zúčastněné role

- Administrátor, super-terapeut

Vstupní podmínky

- Uživatel musí být přihlášený, viz UC02.
- Uživatel musí mít přístupný některý z programů, který je možné editovat.

Následná podmínka

- Aktualizace programu.

Alternativní průběh

- 2a – Program je editovatelný pouze tehdy, pokud je uživatel jeho autorem a zároveň není dosud použitý. V ostatních případech je dostupné pouze tlačítko „Náhled“.

3.3.6 Zobrazení seznamu úloh (UC18)

Popis

Posloupnost akcí vedoucí k náhledu na seznam úloh, ke kterým má daný uživatel přístup. Jedná se o úlohy, které sám vytvořil nebo jsou veřejné.

Standardní průběh

1. Uživatel klikne do hlavního menu na kartu „Úlohy“ a následně na položku „Správa úloh“.
2. Uživateli se zobrazí seznam dostupných úloh.
    - Alternativa 2a – Seznam je prázdný.

Zúčastněné role

- Administrátor, super-terapeut, terapeut

Vstupní podmínky

- Uživatel musí být přihlášený (viz UC02)

Následná podmínka

- Uživateli se zobrazí seznam úloh, ke kterým má přístup a odpovídají zadanému filtru.

Alternativní průběh

- 2a – Seznam může být prázdný z několika důvodů. Buď uživatel žádnou úlohu nevytvořil, nebo neexistuje úloha, která by byla veřejná. Další možností je, že žádná z úloh neodpovídá zadanému filtru.

3.3.7 Editace úlohy (UC20)

Popis

Operace, při které dojde k definování konkrétní podoby úlohy, tedy převážně k nadefinování vstupních parametrů úlohy. Při vytváření úlohy (UC19) dojde pouze k jejímu založení - vyplnění jejich povinných atributů. V editaci úlohy je možnost nastavit příznak její veřejnosti, tj. zda ji mohou používat i ostatní uživatelé aplikace, nebo pouze její vlastník. Dalším atributem je příznak její

aktivity, tj. zda může být úloha vůbec použita. Neaktivní úlohu vidí pouze její autor a může se rozhodnou, zda ji zaktivuje, nebo ji z aplikace odstraní.

Standardní průběh

1. Uživatel si zobrazí jemu dostupné úlohy (viz UC18)
2. V tomto seznamu si vybere konkrétní úlohu a klikne na tlačítko „Editovat“, čímž se mu zobrazí editor úlohy.
    - Alternativa 2a – Položka neobsahuje tlačítko „Editovat“.
3. V tomto editoru provede potřebné změny a klikne na tlačítko „Uložit“.

Zúčastněné role

- Administrátor, super-terapeut, terapeut

Vstupní podmínky

- Terapeut musí být přihlášen (viz UC02).
- Musí existovat úloha, ke které má uživatel přístup (viz UC18)

Následná podmínka

- Úloha bude aktualizována.

Alternativní průběh

- 2a – Úloha je editovatelná pouze v případě, když ji chce upravit její vlastník a zároveň úloha nesmí být dosud přiřazena pacientovi.

3.3.8 Zobrazení seznamu balíků (UC25)

Případ užití, ve kterém se terapeutovi zobrazí seznam balíků, které vytvořil pro své pacienty.

Standardní průběh

1. Uživatel klikne do hlavního menu na kartu „Balíky“ a následně na položku „Správa balíků“.
2. Uživateli se zobrazí seznam jeho balíků
    - Alternativa 2a – Seznamů balíků je prázdný.

Zúčastněné role

- Super-terapeut, terapeut

Vstupní podmínky

- Uživatel musí být přihlášený (viz UC02)

Následné podmínky

- Uživateli se zobrazí seznam balíků, které vytvořil pro své pacienty.

Alternativní průběh

- 2a – Seznamů balíků může být prázdný buď z důvodu, že terapeut dosud žádný nevytvořil, nebo že žádný z balíku neodpovídá zadanému filtru.

3.3.9 Vytvoření balíku s úlohami (UC26)

Popis

Úlohy jako takové nelze zadávat samostatně, a tak je potřeba úlohy organizovat do balíků. Vytvoření takového balíku se provádí následujícím způsobem.

Standardní průběh

1. Uživatel klikne v horním menu na kartu „Balíky“ a následně na položku „Vytvořit balík“.
    - Alternativa 1a – Možnost založení nového balíku ze seznamu balíků.
2. Následně se uživateli zobrazí okno, kde musí vyplnit název balíku a ze seznamu vybrat pacienta, kterému tento balík bude patřit.
    - Alternativa 2a – Seznam pacientů je prázdný.
3. Poté je terapeut přesměrování do editoru balíku, který je popsán v UC27.

Zúčastněné role

- Super-terapeut, terapeut

Vstupní podmínky

- Uživatel musí být přihlášený (viz UC02).
- Uživatel musí mít nějakého pacienta.

Následná podmínka

- Vytvoření balíku, který je přidružen k pacientovi.

Alternativní průběh

- 1a – Možnost vytvoření balíku je také v pohledu seznamu již vytvořených balíků (viz UC25) v pravém horním rohu pod odkazem „Vytvořit balík“.
- 2a – Pokud terapeut nemá žádného pacienta, pak nemůže vytvářet balík, protože se jedná o povinný údaj. V tomto případě průběh vytváření balíku končí.

3.3.10 Editace balíku s úlohami (UC27)

Pomocí této akce je umožněna terapeutovi manipulace s balíkem. Terapeutovi je zde umožněno vkládat nové úlohy do balíku či je z balíku odebírat. Součástí editoru by měl být i seznam se zpětnou vazbou od pacienta na zadaný balík, kterou pacient má možnost vytvořit po procvičení daného balíku.

Standardní průběh

1. Uživatel si zobrazí seznam vytvořených balíků (viz UC25)
2. U hledaného balíku klikne uživatel na tlačítko „Editovat“.
    - Alternativa 2a – Do editace balíku se lze dostat i přes vytvoření nového balíku (viz UC26)
3. Uživatel v balíku provede potřebné změny a následně je uloží kliknutím na tlačítko „Uložit“.
4. Uživateli se zobrazí upravený balík se všemi změnami.

Zúčastněné role

- Super-terapeut, terapeut

Vstupní podmínky

- Uživatel musí být přihlášený (viz UC02).
- Uživatel musí mít již vytvořený nějaký balík.

Následná podmínka

- Aktualizace existujícího balíku.

Alternativní průběh

- 2a – Jakmile dojde k vytvoření nového balíku (UC26), pak je uživatel automaticky přesměrován do jeho editace.

3.3.11 Analýza výsledků (UC30)

Velice důležitá funkcionalita určena pro vyhodnocení pokroku pacienta. Terapeut má zde možnost náhledu na tabulkový výpis výstupních parametrů z jednotlivých spuštění dané úlohy.

Standardní průběh

1. Přihlášený uživatel klikne v hlavním menu na kartu „Analýzy“ a následně na položku „Analýza“.
2. Následně je uživatel přesměrován do editoru, ve kterém má terapeut možnost filtrace úloh podle typu programu, názvu úlohy a jména pacienta. Tyto filtry se také dají kombinovat. Touto filtrací si dohledáme konkrétní úlohu a kliknutím na tlačítko „Analyzovat“ dojde k přesměrování na analýzu.
    - Alternativa 2a – Analýza je také dostupná přes editor balíku.
3. V tomto okně se nalézá seznam výstupních hodnot z jednotlivých provedení konkrétní úlohy, ze kterých terapeut provádí analýzu.

Zúčastněné role

- Super-terapeut, terapeut

Vstupní podmínky

- Uživatel musí být přihlášený do systému (viz UC02).

Následná podmínka

- Uživatel provedl analýzu výsledků pacienta

Alternativní průběh

- 2a – Jednotlivé úlohy jsou strukturovány do balíků, které jsou zadávány pacientům. Proto se v jeho editoru (viz UC27) nalézá seznam úloh ze který se skládá. Každá úloha v seznam obsahuje tlačítko „Analýza“, která umožní náhled na její výsledky. Případ užití pokračuje bodem 3.

3.3.12 Ostatní případy užití

Výše jsme si uvedli základní scénáře, které jsou v aplikaci nejčastěji používány. V následující podkapitole v krátkosti popíši zbytek případů užití. Detailní přehled všech scénářů bude součástí přílohy (viz strana 82).

Správa uživatelů v aplikaci (UC04)

Jedná se o funkcionalitu, která slouží administrátorům aplikace pro nastavování důležitých informací, hlavně uživatelských rolí, jednotlivým uživatelům.

Tato možnost je dostupná pod kartou „Uživatelé“ a záložkou „Správa uživatelů“. Následně dojde k zobrazení seznamu všech uživatelů a pomocí tlačítek k jejich editaci či náhledu.

Správa profilu uživatele (UC05)

Jedná se o scénář, ve kterém je uživateli umožněno nastavit svůj vlastní profil. Jedná se o osobní informace, kontaktní a registrační údaje. Tato akce je dostupná po kliknutí na kartu, jejíž název se shoduje se jménem aktuálně přihlášeného uživatele v levém horním roku a záložku „Nastavení účtu“. Uživatel zde modifikuje potřebné údaje a klikne na tlačítko „Uložit vše“.

Správa pacientů (UC06)

Akce určená pro terapeuty, která slouží pro detailní náhled svých pacientů. Seznam aktivních pacientů je dostupný pod kartou „Pacienti“ a záložkou „Správa pacientů“.

Po nalezení hledaného pacienta se terapeut dostane do jeho profilu, kde jsou uvedeny základní informace o pacientovi a všechny úlohy, které mu byly

kdy zadány - tyto úlohy lze analyzovat. Dále je seznam všech zpráv[1], které terapeut o pacientovi napsal, včetně možnosti přidání nové zprávy.

Zobrazení seznamu překladů (UC07)

Překlady jsou v aplikaci vytvořeny pro to, aby bylo možné jazykově lokalizovat nově vytvořené programy. Jedná se o lokalizaci vstupních a výstupních parametrů programů, aby terapeut věděl, které parametry co znamenají. Zobrazení seznamu překladů je dostupné v horním menu pod kartou „Překlady“ a následně záložkou „Správa překladů“. Tento případ užití je dostupný všem rolím, které mají možnost vytvářet programy, tedy administrátorovi a super-terapeutovi.

Vytvoření nového překladu (UC08)

Pokud uživatel při vytváření nového programu zjistí, že potřebuje nový překlad pro nadefinování vstupu či výstupu z programu, pak si jej vytvoří a následně použije. Vytvářet překlad jde přes jednoduchý editor dostupný pod kartou „Překlady“ a záložkou „Vytvořit překlad“. Dále ho lze vytvořit v editoru programu, aby se uživatel nemusel přepínat do jiného okna.

Při zakládání nového překladu stačí definovat pouze jeho povinné atributy, tj. výchozí jazykovou mutaci, kterou je čeština.

Editace překladu (UC09)

Při vytváření překladu není povinné ihned uvádět všechny jazykové mutace. Proto je později vhodné tyto údaje doplnit. Modifikace překladu je dostupná přes „Správu překladů“ (viz UC07), kde u jednotlivých záznamů je tlačítko „Editovat“.

Duplikace překladu (UC10)

V některých případech může být užitečné vytvořit nový překlad na základě již existujícího, proto je zde možnost duplikace překladu. Tato možnost je rovněž přístupná přes „Správu překladů“ (viz UC07) přes tlačítko „Duplikovat“.

[1]Jedná se pacientské posudky, které terapeut vytváří na základě analýzy úloh, tedy pokroku pacienta.

Smazání překladu (UC11)

Pokud chce uživatel smazat nehodící se překlad, může ho smazat kliknutím na tlačítko „smazat“ u konkrétního překladu ve „Správě překladů“ (viz UC07). Tato operace je dostupná pouze u překladů, které nejsou nikde použity.

Vytvoření nového programu (UC13)

Oprávnění uživatelé mají možnost vytvářet nové programy. Těmito uživateli jsou administrátoři a super-terapeuti. Vytvořením programu se rozumí jeho založení, tedy vyplnění povinných atributů. Po jeho založení uživatel pokračuje v jeho editaci (viz UC14). Uživatel může založit program kliknutím na kartu „Programy“ a následně na položku „Vytvořit program“. Poté je přesměrován do editoru, kde vyplní povinné údaje a kliknutím na tlačítko „Založit“ bude přesměrován do jeho editace.

Náhled programu (UC15)

Tento scénář je téměř totožný od jeho editace (viz UC14) s tím rozdílem, že zde jsou téměř všechny informace pouze pro čtení. Nastavit se dá příznak aktivity a také se dá nahrát nový skript, který úlohu spouští.

Duplikace programu (UC16)

Operace, která vytvoří kopii existujícího programu, respektive jeho funkčního předpisu. Skript pro spuštění programu se v tomto případě neduplikuje, protože se předpokládá, že program budeme chtít aktualizovat.

Akce je jednoduše přístupná ze správy programů (viz UC12). Duplikovat lze takový program, který je již používaný.

Smazání programu(UC17)

Smazání programu je možné pouze v případě, dokud není v aplikaci použit, resp. z daného programu nebyla dosud vytvořena úloha. Pokud bychom i tak chtěli smazat takový program, musíme nejprve odebrat všechny úlohy, které byly na základě programu vytvořeny.

Zobrazení všech programů bylo vysvětleno v případu užití UC12 a pokud budou splněny podmínky pro jeho smazání, bude zde tlačítko „Smazat“.

Vytvoření nové úlohy (UC19)

Jedná se o scénář, který pouze založí novou úlohu zadáním jejích povinný parametrů. Tato akce je dostupná v hlavním menu pod kartou „Úlohy“ a následně položkou „Vytvořit úlohu“. Po zadání všech parametrů a kliknutí na tlačítko „Založit“ je uživatel přesměrován do editoru úlohy, kde má možnost úlohu dodefinovat (viz UC20).

Duplikace úlohy (UC21)

Jedná se o operaci, při které se bude vytvářet kopie existující úlohy. Funkcionalita bude určena k vytváření úloh, které se od sebe výrazněji neliší, a tak bude ulehčena její tvorba. Tímto se vytvoří nová úloha s totožným zadáním. Tato možnost je přístupná ve správě úloh (viz UC18) kliknutím na tlačítko „Duplikace“ v případě, že požadovaná úloha je již používána.

Smazání úlohy (UC22)

Operace, která je opět dostupná přes „správu úloh“ z UC18 kliknutím na tlačítko „Smazat“. Tuto možnost má pouze vlastník úlohy a to jen v případě, že danou úlohu nemá zadanou žádný z pacientů. Úlohu, která obsahuje nějaké výsledky, již nelze smazat.

Náhled úlohy (UC23)

V případě nedostupné editace úlohy (viz UC20) má uživatel možnost jejího náhledu, kde již není umožněna změna jejího zadání. Uživatel má zde pouze možnost, pokud je jejím autorem, měnit příznak její veřejnosti, tj. zda úlohu mohou zadávat i ostatní terapeuti. Neveřejnou úlohu může zadávat pouze její autor.

Otestování vytvořené úlohy (UC24)

Předtím, než se terapeut rozhodne některou ze svých úloh použít, je dobré tuto úlohu otestovat, tedy zjistit, zda funguje podle očekávání. Tuto možnost má autor úlohy v rámci její editace a náhledu (viz UC20 a UC23) pomocí tlačítka „Otestovat úlohu“ v pravém horním rohu. Následně dojde ke spuštění dané úlohy.

Duplikace balíku (UC28)

Balík v každém případě patří pouze jednomu pacientovi. Pokud se terapeut rozhodne, že by chtěl konkrétní balík včetně všech jeho úloh přiřadit někomu

jinému, pak v rámci jeho editace (viz UC27) klikne na tlačítko „Duplikovat“. Pote bude terapeut dotázán na zadání nového jména pro balík a výběru pacienta, kterému bude balík přiřazen. Následně dojde k vytvoření nového, identického, balíku.

Smazání balíku (UC29)

Každý terapeut musí mít možnost smazat již nepotřebný balík. Tuto operaci může provést přes „správu balíků“ (viz UC25) kliknutím na tlačítko „smazat“ u konkrétního záznamu, nebo v rámci jeho editace (viz UC27) kliknutím na tlačítko „Smazat balík“ v pravém horním rohu.

Smazáním balíku nedojde k odebrání získaných výsledků z jednotlivých cvičení, pouze dojde ke ztrátě seskupení úloh, které byly v balíku obsaženy. Pokud bude úloha přidána do jiného balíku, pak tento balík bude obsahovat i data z předchozích spuštění úlohy.

3.4 Požadavky na vnější rozhraní

3.4.1 Uživatelská rozhraní

Mezi hlavní požadavky na uživatelské rozhraní aplikace patří její intuitivnost a přehlednost, protože je určena zejména pro uživatele s velmi omezenými kognitivními funkcemi či jinými vážnými problémy. Proto musí být veškerý obsah určený pro pacienta přehledně zobrazen na hlavní obrazovce aplikace bez nutnosti hledání v menu. V případě ostatních uživatelů systému to tak být nemusí.

3.4.2 Komunikační rozhraní

Snahou aplikace nebude zahrnout veškerou komunikaci mezi jejími uživateli ve smyslu nahrazení dosavadních komunikačních prostředků, ke kterým patří psaní emailů, telefonní hovory, videokonferencí pomocí aplikace Skype apod.

Aplikace zahrne pouze základní informační notifikace prostřednictví emailových zpráv, které se budou odesílat při různých akcích, od kterých by se to očekávalo. Tyto akce mohou být následujícího charakteru:

- **Registrace nového uživatele** – administrátor bude upozorněn o novém uživateli, kterému nastaví jeho roli.
- **Přiřazení nového balíků s úlohami** – pacientovi se zobrazí na jeho uvítací stránce nový balík a zároveň o jeho existenci bude upozorněn emailem.

- **Přiřazení k novému terapeutovi**.
- **Pacient ohodnotil balík s úlohami**.
- **Terapeut napsal hodnocení pacienta** týkající se jeho pokroku.

Komunikace mezi terapeutem a pacientem tedy zůstane stejná jako doposud, jen bude rozšířena o emailové notifikace z aplikace.

3.5 Další parametrické (mimo-funkční) požadavky

3.5.1 Bezpečnostní požadavky

Velký důraz při používání této aplikace musí být věnován nakládání s osobními údaji, kde nesmí dojít k jejich zneužití. Současně musí být systém navržený takovým způsobem, aby ke zneužití dat vůbec dojít nemohlo. Aby k tomuto nedošlo, tak je v aplikaci použito několik rolí, díky kterým lze povolit přístup pouze k těm datům, ke kterým mají oprávnění. Oprávnění není orientováno pouze na role, ale také na uživatele. Proto je obsah, který je zobrazován, plně personalisovaný.

3.5.2 Multijazyčné prostředí

Každý z programů musí podporovat všechny jazykové lokalizace, aby mohli být pohodlně používány všemi uživateli, především pacienty a terapeuty, v systému. Z tohoto důvodu bude naimplementován mechanismus, který umožní správu překladů s jejich následným použitím při definování programů a úloh.

3.5.3 Kvalitativní parametry

- **požadavky na snadnost používání** - Výše ve specifikaci bylo popsáno, že je tato aplikace určena pro osoby, kteří trpí neurologickými poruchami a jen velmi obtížně komunikují s okolním světem. Je tedy naprosto nezbytné, aby veškeré ovládací prvky byly co nejjednodušší a intuitivní. Proto veškeré ovládací prvky, které slouží hlavně pro neurorehabilitaci, musí být součástí domovské stránky pacienta. Pacientovi pak bude stačit po navštívení aplikace jedním kliknutím spustit cvičení.

Ostatní role budou mít svoji funkcionalitu dostupnou především pomocí menu, ale její určenou podmnožinu budou mít dostupnou taktéž z domovské stránky

- **požadavky na udržovatelnost** - Jedním z cílů této diplomové práce je vytvořit aplikační framework, který bude obsahovat sadu programů s úlohami, které si budou moci pacienti procvičit.

  Je tedy žádoucí, aby bylo v budoucnosti možné přidávat další programy, tj. aby framework poskytl rozhraní pro přidání dalších programů. Každému programu, který bude vytvořen pomocí této aplikace, musí být nadefinovány vstupní a výstupní parametry včetně programového skriptu, který při spuštění aplikace převezme právě tyto vstupní parametry a po dokončení úlohy vrátí data ve formátu výstupních parametrů.

4 Dostupné technologie

Před samotným návrhem a implementací webové aplikace je nutné zohlednit, které technologie a principy mám k dispozici a které z nich bych mohl použít.

Nejprve je nutné zvolit některý z dostupných webových frameworků, ve kterém bude daná aplikace vyvíjena.

Na základě zvolené webové technologie se bude odvíjet výběr technologie databázové. Zde se musím zamyslet především nad charakterem ukládaných dat, tj. zda je výhodné použít relační nebo dokumentovou databázi.

Při vývoji této webové aplikace jsem se rozhodl využít velmi populární architektonický vzor **Model-View-Controller** (dále pouze MVC), díky kterému dojde k výraznému zpřehlednění kódu a zvýší efektivitu vývoje [17]. Tento návrhový vzor je využíván ve většině oblastí softwarového vývoje, a to proto, že výrazně zvyšuje přehlednost a vývoj aplikací.

První části se říká **model**, který je zodpovědný za obchodní logiku aplikace a zapouzdřuje metody pro přístup k datům (databáze, soubory, atd.). Druhé části, která určuje, jak budou data zobrazována, se říká **view**. Dále je to také označováno jako pohled či šablona. Poslední částí, která je zodpovědná za vyřizování požadavků, je **kontroler**. Kontroler přijímá požadavky z okolí a prostřednictvím modelu získává potřebná data, která předá pohledu pro zobrazení.

Všichni níže uvedení kandidáti, ze kterých budu následně vybírat cílový webový framework, musí tedy tuto modulární architekturu podporovat.

4.1 Webového frameworky

V současné době existuje velké množství frameworků pro různé programovací jazyky. Pro ulehčení volby se můžeme řídit pomocí několika výběrových kritériích [4], mezi které patří náročnost naučení se dané technologii, velikosti vývojářské komunity nebo kvalitou dokumentace. Dalšími ukazatelem pro výběr může být zaměření daného frameworku, jeho výkonnost a úroveň zabezpečení. Subjektivní měřítko, které má velkou váhu, je samozřejmě zkušenost s daným frameworkem či programovacím jazykem.

4.1.1 ASP.NET MVC 5

Jedná se o aplikační framework umožňující tvorbu dynamických webových aplikací prostřednictvím programovacího jazyk C#, který implementuje návrhový vzor MVC[14]. Jde o velmi populární framework s velkou vývojářskou komunitou a výbornou dokumentací.

Framework také poskytuje vysokou míru zabezpečení díky technologii ASP.NET Core, která obsahuje nástroje pro zajištění autentizace, autorizace a ochrany dat. ASP.NET společně s EntityFrameworkem (dále jen EF, který bude popsán v podkapitole 4.3) obsahuje ještě další obranné mechanismy před nejběžnějšími útoky, kterými můžou být Cross-site scripting, SQL Injection, Cross-Site Request Forgery a Open redirect útoky. Tyto typy útoků a další budou popsány v kapitole zabývající se testováním (viz kapitola 7).

Autentizace a autorizace v aplikaci je řešena pomocí frameworku **ASP NET Identity**. Tento framework dokáže identifikovat aktuálně přihlášeného uživatele a na základě jeho role mu povolí přístup pouze k jeho datům. Dále tento framework nabízí možnost přihlášení prostřednictvím již vytvořených účtů u společností jako jsou Microsoft, Facebook nebo Twitter.

Díky použití návrhového vzoru MVC je použití frameworku jednoduché a práce efektivní.

4.1.2 Spring MVC

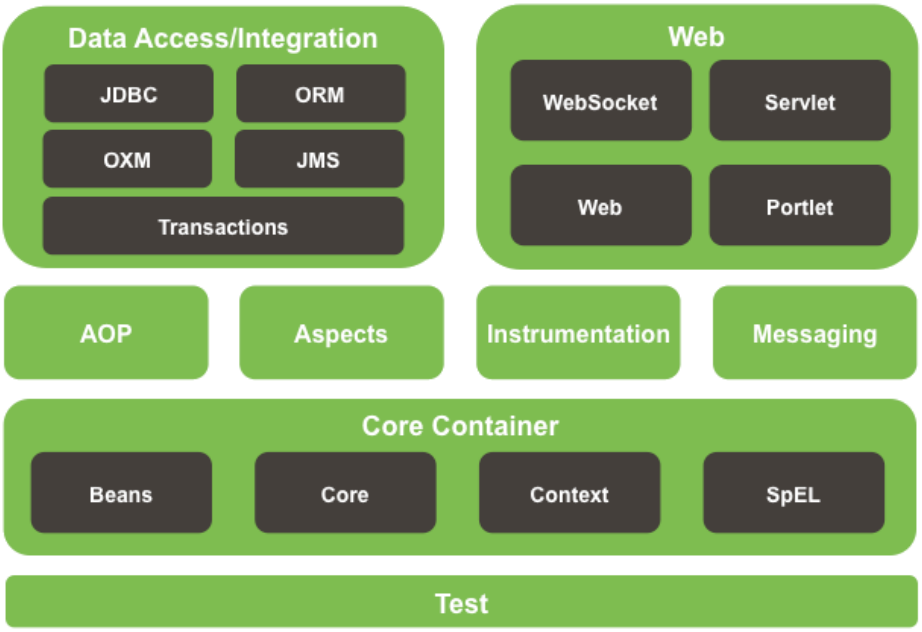
Spring je jedním z nejpopulárnějších frameworků pro vývoj v Javě, resp. v javaEE[1]. Již základní verze Springu se vyznačuje především vysokou mírou modulárnosti a obsahuje celkem osm kategorií modulů (viz obr. 4.1). Spring umožňuje vytváření jak webových, tak i desktopových aplikací.

Mezi jeho velké výhody patří především velikost komunity a kvalitní dokumentace, která je rozsáhlá a popisuje využití jednotlivých modulů. Do Springu lze také integrovat pomocné frameworky pro usnadnění implementace, mezi které patří především Hibernate pro objektově relační mapování databáze nebo Spring Security pro zabezpečení aplikace.

Spring Security je tedy framework, který poskytuje autentizaci, autorizaci a další bezpečnostní prvky pro podnikové aplikace. Dále zajišťuje ochranu proti útokům typu Clickjacking, Cross-site Request Forgery, Cross-

[1]Zkratka pro Javu Enterprise Edition, tj. součásti platformy Java, která je určená pro vývoj a provoz podnikových aplikací a informačních systémů.

site scription, atd[21].



Obrázek 4.1: Typy modulů ve frameworku Spring

4.1.3 Ruby on Rails

Jedná se o volně dostupnou technologii napsanou v programovacím jazyce Ruby, která podporuje návrhový vzor MVC. Jedná se o framework, který má velmi širokou komunitu a mezi začátečníky s webovými aplikacemi je velmi oblíbený, protože obsahuje velké množství tutoriálů a dalších zdrojů. Rails je velmi oblíbený framework, který obsahuje mnoho užitečných knihoven pro zvýšení rychlosti a efektivity vývoje.

Rails lze považovat za velice bezpečný webový framework, jehož bezpečnost je pro něj prioritou. Framework obsahuje velmi podrobnou dokumentaci [19], ve které je uživatel podrobně instruován, jak útokům zabránit.

4.1.4 Django

Django je volně dostupný webový framework, který umožňuje velmi rychlý vývoj webových aplikací v programovacím jazyce Python. Díky velké komunitě vývojářů obsahuje mnoho užitečných knihoven, které mohou vývojáři použít a zjednodušit si tím vývoj své aplikace. Django je vysoce univerzální framework, může být použit od systémů pro správu obsahu, po sociální sítě

až po vědecké počítačové platformy. Součástí frameworku je také rozsáhlá dokumentace.

Django je považován za bezpečný framework, kdy některé bezpečnostní prvky fungují implicitně, jiné je potřeba aktivně použít. Součástí jsou funkce pro autentizaci uživatele a ochrana před známými útoky typu XSS, CSFR, SQL injection, Clickjacking atd.

4.1.5 Volba aplikačního frameworku

V této kapitole dojde k výběru konkrétního frameworku pro tvorbu této práce, kdy máme možnost výběru celkem ze čtyř frameworků, které byly popsány v předchozí podkapitole. Všechny popsané frameworky splňovali základní kritéria pro výběr. Jednalo se o oblíbené a velmi rozšířené frameworky, které mají velké komunity vývojářů a podrobné dokumentace, což může pomoci při vývoji. Dále se jedná o velice bezpečné frameworky, které dokáží ochránit citlivá data.

Dalším kritériem byla osobní znalost programovacích jazyků. Z tohoto důvodu jsem se rozhodl výběr zúžit na frameworky ASP.NET MVC a Spring, protože jak s Ruby, tak s Pythonem mám minimální zkušenosti.

Nakonec jsem se rozhodl dát přednost ve volbě dosavadní zkušenosti a zvolil jsem framework ASP.NET MVC, který jsem si již vyzkoušel v rámci mimoškolního projektu. Tato volba se odrazí i následující kapitole, ve které budou popisovány databázové technologie, které musejí být s tímto frameworkem kompatibilní.

4.2 Volba databáze

V současné době se téměř žádná aplikace neobejde bez ukládání dat do databáze a právě volba databáze je klíčová pro další pokračování vývoje. Databáze se dělí do dvou hlavních kategorií, na relační a nerelační, které jsou označovány jako NoSQL databáze. Jednotlivé kategorie mají své výhody i nevýhody, a proto jim budou věnovány následující dvě podkapitoly. Poté si představíme jednoduchou tabulku (viz tab. 4.1), ve které budou popsány hlavní rozdíly mezi SQL a NoSQL databázemi.

4.2.1 Relační databáze

Tradiční relační databáze (SQL) se používají již od osmdesátých let. Základním prvkem těchto databází jsou relace mezi databázovými tabulkami. Ta-

bulky se skládají ze záznamů (řádků), které jsou pevně definovány atributy (sloupce). Atributy jsou definovány konkrétním datovým typem a rozsahem hodnot, kterých může atribut nabývat. Každý záznam dále musí obsahovat primární klíč, který jednoznačně identifikuje každý řádek v tabulce. Může se jednat o jeden nebo více sloupců, které dohromady tvoří unikátní kombinaci. Další důležitou částí jsou cizí klíče, které slouží právě pro vyjádření oněch relací/vazeb, které mezi sebou jednotlivé záznamy z různých tabulek mají. Jde opět o jeden či více sloupců, které jednoznačně identifikují záznam v jiné tabulce. V neposlední řadě je významným prvkem relačních databází integrita dat.

Tento typ databáze musí využívat tzv. ACID transakcí. Jedná se o zabezpečení databáze skládající se ze 4 předpokladů, kterými jsou atomicita (transakce se buď provede celá, nebo vůbec), konzistence (v každý okamžik jsou data v bezchybném stavu), izolovanost (transakce se mezi sebou neovlivňují) a trvalost (zapsaná data se neztratí).

4.2.2 NoSQL databáze

V posledních letech se začíná hojně využívat systémů, které mají poměrně odlišné vlastnosti od tradičních relačních databází. Jedná se o tzv. NoSQL (nerelační) databáze.

Tento typ databáze se hodí v případě, když nejsme schopni dopředu určit strukturu dat, respektive její podobu v relační databázi. Struktura ukládaných dat se může neustále měnit, přibývají další typy parametrů, případně různé položky jsou u různých parametrů povinné/volitelné [13].

U těchto databází je kladen důraz především na co největší propustnost, ať už při zápisu nebo při čtení. Aby tyto požadavky byly splněny, většinou se rezignuje na některé principy známé z RDBMS[2]. Typicky je to pravidlo třetí normální formy, data nejsou atomická, neexistují transakce a není zaručena konzistence databáze v každé chvíli. V některých případech toto nemusí být problém, např. nevadí, když se některé informace uloží o pár sekund později.

NoSQL databáze se dělí do několika kategorií podle principu manipulace s daty, které byly popsány v tabulce 4.1. Jedná se hlavně o dokumentové, grafové a databáze pracující na principu ukládání hodnot pod konkrétními klíči.

[2]Relation DataBase Management System – Relační databáze

Vlastnost	SQL	NoSQL
Organizace dat	Předem dané, jasně definované schéma	Dynamické schéma nestrukturovaných dat
Styl ukládání dat	Tabulky	Dokumenty, dvojice klíč-hodnota, grafové struktury, ...
Dotazovací jazyk	Standardizovaný SQL	Každá NoSQL databáze má svůj vlastní
Bezpečnost	ACID přístup	ACID nelze zajistit

Tabulka 4.1: Porovnání vlastností SQL a NoSQL přístupu.

4.2.3 Zvolení databázového přístupu

Na základě zvoleného aplikační frameworku, kterým je ASP.NET MVC, je nutné zvolit především relační databázi, kterou pro svoji práci potřebuje ASP.NET Identity framework, který je určen pro autentizaci a autorizaci uživatelů do aplikace. Navíc většina ukládaných dat budou mít pevně danou svoji strukturu, a tak volba relační databáze je určitě správná.

Dále zde ještě existuje možnost využití obou typů databázových přístupu, respektive zvolit jako druhý typ databáze NoSQL databázi dokumentového typu. Tento typ ukládání dat by mohl být použit například pro manipulaci s programy nebo úlohami. Bude se jednat o typ objektů, které nemají pevně daný počet svých atributů, ovšem struktura těchto atributů je stejná. Proto jsem se rozhodl použít v této práci pouze relační databázi.

Jelikož framework ASP.NET MVC používá jako výchozí databázový systém Microsoft SQL server, tak jsem se jej rozhodl v této práci také použít.

4.3 Další použité technologie

V rámci vývoje webové aplikace jsem dále použil následující technologie.

4.3.1 HTML a CSS

Základní technologie pro tvorbu webových stránek. Pomocí značkovacího jazyka HTML a jeho značek se definuje obsah stránky a pomocí technologie CSS zase její obsah. Pomocí technologie CSS jsme schopni jeden obsah prezentovat různými způsoby pro různá zařízení od velkých obrazovek pracovních stanic po malé displeje mobilních telefonů či jako požadovaný výstup z tiskárny. Oddělení struktury dokumentů od jejích prezentačních stylů vý-

razně ulehčuje údržbu stránek a zároveň je tak umožněno sdílení stylů napříč vícero stránkami [20][8].

4.3.2 Bootstrap

Bootstrap je front-end framework pro vytváření responzivních[3] webových stránek. V současnosti se jedná o jeden z nepoužívanějších a nejoblíbenějších frameworků ve své kategorii, který je zároveň volně dostupný a zahrnuje v sobě soubor nástrojů pro vývoj pomocí technologií HTML, CSS a JavaScriptu. Umožňuje rychlé a relativně snadné vytváření prototypů webových stránek, přináší responzivní systém rozvržení elementů na stránce, nové komponenty a mocná rozšíření založená na jQuery [6].

Výhodami tohoto frameworku je především snadnost jeho použití, kdy si vývojář postačí se základní znalostí HTML a CSS, responzivita a kompatibilita se všemi moderními webovými prohlížeči [7].

4.3.3 JavaScript, Ajax a jQuery

JavaScript je dynamický programovací jazyk. Používá jako součást webových stránek, jejichž implementace umožňují klientskému skriptu komunikovat s uživatelem a vytvářet dynamické stránky. Jedná se o interpretovaný programovací jazyk podporující objektově orientovaný přístup [11]. Skript by měl být obsažen v dokumentu HTML nebo na něj odkazovat, aby kód mohl být interpretován prohlížečem. Mechanismus JavaScriptu na straně klienta poskytuje mnoho výhod oproti tradičním skriptům na straně serveru CGI. Můžeme například použít JavaScript k validaci formuláře pro ověření platnosti formátu e-mailové adresy apod. JavaScript lze použít k zachycení událostí iniciovaných uživatelem, jako jsou klepnutí na tlačítko, navigace odkazem a další akce, které uživatel inicializuje explicitně nebo implicitně.

JQuery je výkonná knihovna napsaná v JavaScriptu, jejíž cílem je ušetřit vývojáři práci ve smyslu využití funkcí, který zaobalují danou funkcionalitu v JavaScriptu. Jinak řečeno vývojář napíše méně kódu. JQuery zjednodušuje procházení HTML dokumentů, zpracování událostí, animaci a interakce AJAXu pro rychlý vývoj webu [10].

Další použitou technologií je AJAX, což je zkratka pro Asynchronní JavaScript a XML, která umožňuje vytvářet lepší, rychlejší a interaktivnější

[3]Responzivní design je o vytváření takových webových stránek, které se dokáží automaticky upravit tak, aby vypadaly dobře na všech zařízeních, od malých telefonů po velké displeje pracovních stanic.

webové aplikace pomocí XML, HTML, CSS a JavaScriptu [5]. Pomocí této technologie jsme schopni získat ze serveru data po načtení stránky, aktualizovat stránku bez jejího přenačtení nebo posílat data na server aniž by o tom uživatel věděl.

4.3.4 Entity Framework

Entity Framework (dále jen EF) je volně dostupný ORM[4] framework pro .NET aplikace podporované společností Microsoft. EF vytváří tzv. EDM[5], pomocí kterých získává data z databáze, nebo je do databáze ukládá. Tento model je tedy používán pro dotazování nebo ukládání dat do cílové databáze prostřednictvím tzv. LINQ[6] dotazů [9]. Poskytovatel databáze převede tyto LINQ dotazy do dotazovacího jazyka specifického pro databázi (např. SQL pro relační databázi) a vrátí nám dotazovaná data. EF nám také umožňuje spouštět nad databází dotazy v jazyce SQL.

[4]Objektově relační mapování je programovací technika v softwarovém inženýrství, která zajišťuje automatickou konverzi dat mezi relační databází a objektově orientovaným programovacím jazykem.

[5]Entity Data Model – jednoduché třídy s atributy, které korespondují atributům jednotlivým entitám v databázi.

[6]LINQ (Language Integrated Query) poskytuje jednotné dotazovací rozhraní v jazyce C# a VB.NET pro načtení dat z různých zdrojů a formátů. Eliminuje nesoulad mezi programovacími jazyky a databázemi.

5 Návrh aplikace

V předchozí kapitole jsem uvedl několik technologií, ve kterých se v současnosti vyvíjí webové aplikace. Po zvážení všech kritérií jsem zvolil framework ASP.NET MVC, se kterým již mám nějaké zkušenosti. Jako hlavní databázový systém jsem zvolil SQL Server od společnosti Microsoft. Výběr těchto technologií jsem popsal v kapitole 4.

V následujících kapitolách budu popisovat, jakým způsobem jsem se rozhodl navrhnout a následně naimplementovat jádro této webové aplikace. Při návrhu jsem v prvé řadě dbal na její rozšiřitelnost a modulární architekturu. Jádro celé aplikace se skládá z několika modulů, které budou popsány v kapitole 5.3. Dále bylo nutno implementovat administrační část aplikace pro správu všech uživatelů a další nastavení aplikace.

5.1 Architektura

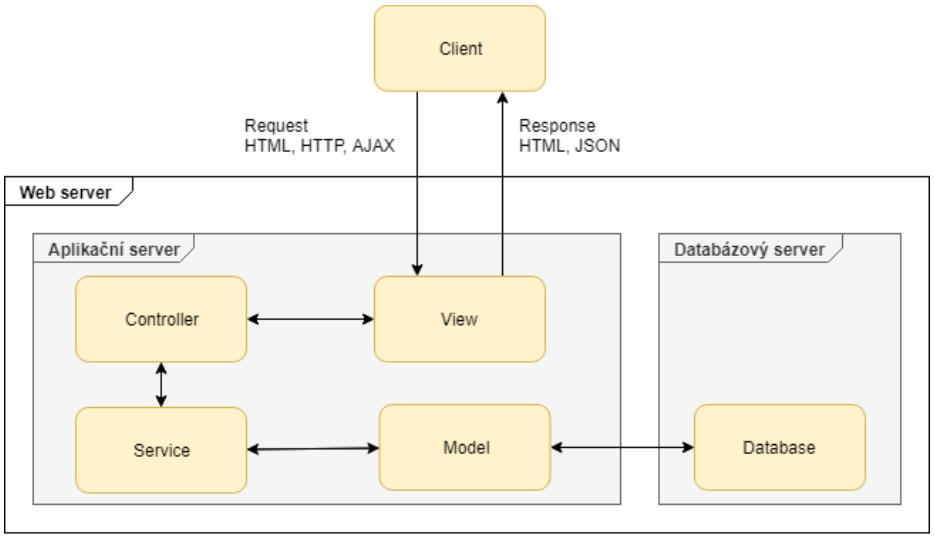
Aplikace je vyvíjena v osvědčené webové technologii ASP.NET MVC, která vynucuje návrhový vzor Model-View-Controller (dále jen MVC). Aplikace se skládá z několika funkčních celků, kdy se o každý z nich stará vlastní kontroler, který je rozšířený o vlastní službu, jenž obstarává veškerou komunikaci s databází. Kontroler tedy pouze přijímá požadavky a odpovídá na ně prostřednictvím přidružené služby, která kontroleru před-připravuje data. Všechny moduly včetně jejich kontrolerů, modelů a pohledů si uvedeme v kapitole 5.3.

Na obrázku 5.1 popisuji základní princip vyřizování uživatelských dotazů na webový server. Uživatel pomocí webového prohlížeče vytvoří požadavek, který následně zachytí kontroler.

Pokud uživatelův dotaz byl charakteru ukládání dat na server, pak kontroler přijme data v tzv. ViewModelu[1], ze kterého vytvoříme objekt databázového modelu, který následně aktualizuje data v databázi.

V případě dotazu typu získání dat ze serveru se pomocí služby získají z databáze potřebná data, které se následně převedou na konkrétní ViewMo-

[1]ViewModel je upravená verze normálního databázového modelu, která obsahuje takové atributy, které se v daném pohledu využívají. Může se tedy jednat pouze o podmnožinu atributů běžného modelu, nebo se naopak může sestavovat z atributů napříč vícero modelů. Tento přístup je vhodný zejména pro formuláře a jejich validaci na základě povinných vstupních hodnot, či jejich formátu.



Obrázek 5.1: Architektura systému

del. Na základě pohledu a tohoto ViewModelu dojde k vygenerování HTML kódu, který se odešle zpět klientovi.

Ve své podstatě bych mohl pro zobrazení cílového HTML kódu použít výchozí model, který provádí mapování databázových tabulek na modely v aplikaci. Tímto způsobem bychom dříve či později narazili na problém nemožnosti přidat do modelu dodatečné informace. Proto se vytváří mapování modelu na ViewModel a dále je velmi doporučováno provádět ještě mapování databázového modelu na cílový pohled[2].

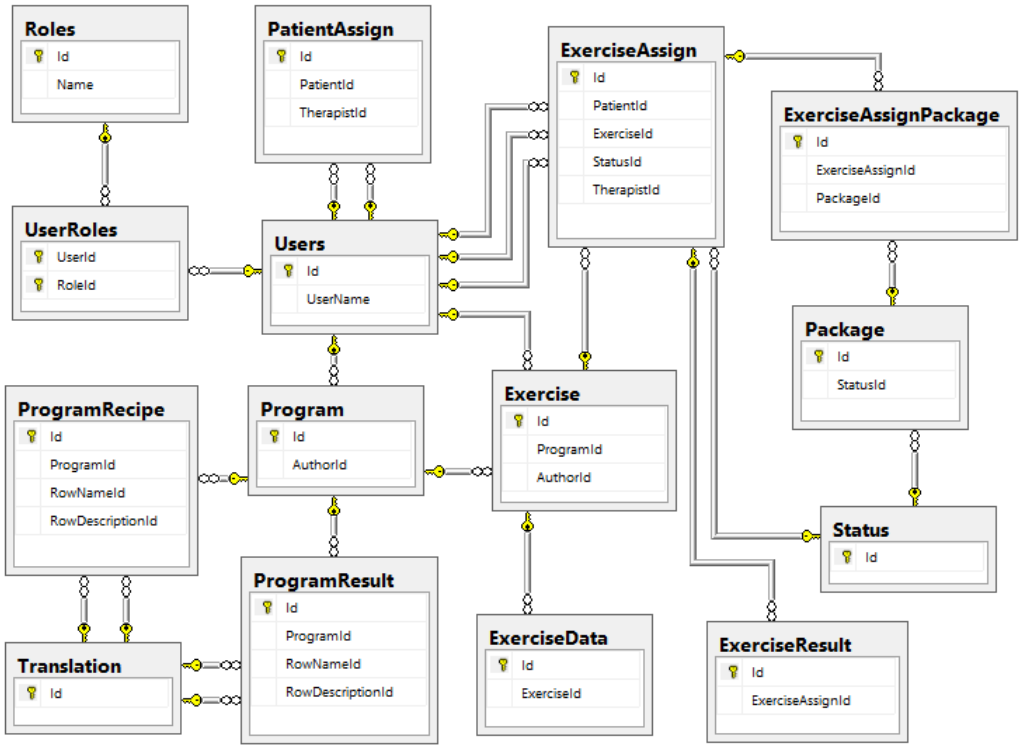
5.2 Databázový model

Po krátkém popisu databázových přístupů v kapitole 4.2 jsem se rozhodl pro relační databázi a navrhl následující množinu tabulek.

V rámci zjednodušeného databázového modelu (viz obrázek 5.2) budu popisovat základní tabulky a jejich význam. V rámci modelu jsou uvedeny pouze primární a cizí klíče, aby byl model přehledný vzhledem k velikosti místa na stránce. Celý model je k dispozici v příloze tohoto dokumentu (viz str. 81) a na nosiči spolu s touto aplikací.

Nyní si uvedeme význam jednotlivých tabulek spolu s jejich důležitými atributy a vazbami na další tabulky, které budou postupně detailně popsány.

[2]Toto mapování znamená, že každá možná akce by měla mít svůj vlastní ViewModel. Tento ViewModel bude unikátní v rámci celé aplikace, a proto v případě jeho modifikace nedojde k ovlivnění ostatních pohledů.



Obrázek 5.2: Zjednodušený databázový model

U jednotlivých atributů bude v případě vazby v závorce uvedena tabulka, tj. záznam z této tabulky, na kterou tento cizí klíč ukazuje. Pokud atribut nepředstavuje cizí klíč, hodnota v závorce bude obsahovat datový typ atributu.

Pokud v rámci nějaké tabulky zmíním atribut „IsActive“, pak v rámci této tabulky není umožněno mazání záznamů. Mazání záznamů není obecně dobrá volba, protože mohou být v budoucnu ještě dále využity. Příkladem takového scénáře může být například možnost obnovení vazby mezi terapeutem a pacientem, kdy by bylo žádoucí vidět výsledky pacienta z předchozího přiřazení. Proto se u těchto záznamů pouze nastaví vlajka indikující jejich platnost.

5.2.1 Users

Jedná se o základní tabulku, ve které jsou udržovány informace o jednotlivých uživatelích v systému, tj. všech terapeutech, pacientech, administrátorech a dalších uživatelích. Data z této tabulky jsou využívána napříč celou aplikací například z bezpečnostních důvodů (přístup k editorům, oprávnění k akcím, zobrazení citlivých dat apod.) a mají následující charakter.

- **Id** (text) – identifikátor záznamu
- **FirstName** (text) – křestní jméno
- **LastName** (text) – příjmení
- **Email** (text) – emailová adresa
- **PhoneNumber** (text) – telefonní číslo
- **PasswordHash** (text) – otisk uživatelského hesla

PatientAssign

Tato tabulka je určena pro ukládání vazeb mezi terapeuty a jejich pacienty. V tabulce jsou ukládány pouze jejich identifikátory a umožňují tak spojit konkrétního terapeuta s konkrétním pacientem.

- **Id** (text) – identifikátor záznamu v tabulce
- **PatientId** (Users) – identifikátor pacienta
- **TherapistId** (Users) – identifikátor terapeuta
- **IsActive** (bool) – indikátor aktivní vazby[3]

5.2.2 Translation

Jedná se o tabulku, ve které jsou uchovávány uživatelské překlady, které mohou být využity pro definování názvů vstupních a výstupních parametrů pro programy. Tyto překlady se na základě nastavené lokalizace zobrazí uživatelům při vyplňování jednotlivých parametrů úlohy. Tabulka obsahuje identifikátor překladu a jeho verze v podporovaných jazycích. Tyto překlady se v jednotlivých editorech - především v editoru programu - dají využít tím způsobem, že budou nabízeny na základě shody napsaného uživatelského textu s hodnotou uloženou v databázi v nastaveném jazyce. Zjednodušeně řečeno uživatel bude zadávat text a aplikace mu bude nabízet vytvořené překlady, které obsahují daný podřetězec v nastavené jazykové mutaci.

Tato tabulka neobsahuje žádné vazby na okolní tabulky a obsahuje následující atributy:

- **Id** (číslo) – jednoznačný identifikátor záznamu v tabulce

[3]Tento typ atributu byl popsán v posledním odstavci úvodu do kapitoly 5.2.

- **Cz** (text) – český překlad
- **En** (text) – anglický překlad
- **De** (text) – německý překlad
- **DbColumnName** (text) – textová hodnota určená pro orientaci v analýze výsledků[4].

5.2.3 Program

Jedná se o tabulku, do které jsou přidávány záznamy o nově vytvořených programech[5]. Jedná se o velmi jednoduchou tabulku, která má následující atributy.

- **Id** (číslo) – identifikátor záznamu v tabulce
- **Name** (text) – název programu
- **IsActive** (bool) – indikátor aktivity programu[6]
- **Description** (text) – krátký popis programu vystihující jeho podstatu
- **AuthorId** (User) – vazba na autora programu[7]
- **IsPublic** – indikátor veřejnosti programu[8]

ProgramRecipe

Jedná se o velmi důležitou tabulku, jejíž záznamy obsahují informace o tom, z kterých částí se jednotlivé programy skládají. Záznamy z této tabulky vytvářejí funkční předpis, který se následně inicializuje při vytváření nové úlohy. Jinak řečeno, co záznam, to jeden vstupní parametr do konkrétní úlohy.

Každý záznam obsahuje tyto atributy:

[4]Tento atribut je uveden z důvodu, abychom při analýze výsledků věděli, co který záznam v databázi znamená.

[5]Pojem program v našem případě znamená funkční prototyp, nebo-li šablona, pomocí které dochází k vytvoření nových úloh.

[6]Pouze aktivní programy mohou být využity pro tvorbu nových úloh. Pokud je program neaktivní, pak to neznamená, že není funkční, mohl být například nahrazen novější verzí.

[7]Identifikátor autora je důležitý zejména při jeho modifikaci, protože pouze autor může měnit jeho podobu.

[8]V návrhu se počítá s tím, že každý program bude veřejný. V rámci spolehlivosti je ovšem důležité, aby program nebyl dostupný veřejnosti v případě, pokud nebyl dostatečně otestován.

- **Id** (číslo) – identifikátor záznamu v tabulce
- **ProgramId** (Program) – vazba na záznam, který uvádí, k jakému programu jsou data přidružena.
- **RowNumber** (číslo) – uvádí pořadí parametrů v rámci jednoho programu, které je důležité zejména při předávání vstupních parametrů výkonnému skriptu.
- **RowNameId** (Translation) – vazba na záznam s překladem dané řádky.
- **RowDescriptionId** (Translation) – vazba na záznam s překladem, který má detailněji popsat význam konkrétního vstupu.
- **DataTypeId** (číslo) – hodnota z číselníku, která je schopna identifikovat typ vstupních dat dané řádky[9]
- **Value** (text) – tento parametr je použit, pokud se parametr úlohy vybírá ze seznamu[10]

ProgramResult

Jedná se o principiálně stejnou tabulku jako předchozí s tím rozdílem, že se používá pro definici výstupu z konkrétnímu programu. Tabulka tedy obsahuje téměř stejné vazby a atributy jako tabulka `ProgramRecipe`, a proto je zde nebudu uvádět. Tabulka neobsahuje pouze atribut „Value“, protože tato hodnota bude získána po procvičení dané úlohy, která bude navíc uchovávána v jiné tabulce.

5.2.4 Exercise

Tabulka pro uchování základních informací o vytvořené úloze. Tabulka obsahuje následující vazby a atributy:

- **Id** (číslo) – jednoznačný identifikátor záznamu v tabulce
- **Name** (text) – název úlohy
- **Description** (text) – krátký popis úlohy

[9] Mezi základní podporované typy patří text, číslo, pravdivostní hodnota, výběr jedné nebo více položek ze seznamu, obrázek a zvuk.

[10] Obsahuje například sekvenci možných výběru ze seznamu, kdy se vybírá jedna nebo několik položek. V ostatních případě je tento atribut nepřístupný.

- **ProgramId** (Program) – typ programu, ze kterého byla úloha vytvořena
- **CreatedDate** (datum) – datum vytvoření úlohy
- **AuthorId** (Users) - vazba na autora úlohy
- **IsActive** (bool) – indikátor aktivity úlohy
- **IsPublic** (bool) – informace, zda je úloha veřejná[11]

ExcerciseData

Každá vytvořená úloha potřebuje pro svoje spuštění množinu vstupních parametrů (obtížnost úlohy, velikost herního pole, atd.). Z tohoto důvodu byla vytvořena tato tabulka, která tyto parametry udržuje. Aby bylo uchováno pořadí těchto vstupů, tak obsahují informaci o jejich pořadovém čísle. Tabulka tedy obsahuje vstupní data do všech úloh, které byly v systému vytvořeny a mají následující podobu.

- **Id** (číslo) - primární klíč záznamů v tabulce
- **ExerciseId** (Exercise) – vazba na konkrétní úlohu, které vstupní parametry patří
- **Row** (číslo) – pořadí vstupního parametru v rámci dané úlohy
- **Value** (text) – konkrétní hodnota daného vstupního parametru

ExcerciseAssign

V systému je žádoucí, aby jednu úlohu bylo možné přiřadit několika pacientům. Z tohoto důvodu byla vytvořena právě tato tabulka, která provádí mapování pacientů k jejich úlohám. Založení těchto vazeb ještě neznamená, že pacienti mohou cvičit dané úlohy. Tyto vazby jsou dále strukturovány do balíků, které se následně pacientům uvolňují.

V aplikaci je dokonce umožněno, aby měl pacient jednu konkrétní úlohu přiřazenou vícekrát, ovšem od různých terapeutů.

Tabulka má následující atributy:

- **Id** (číslo) – jednoznačný identifikátor záznamu v tabulce
- **PatientId** (Users) – vazba na uživatele, kterému je úloha přiřazena

[11] Tento parametr bude blíže přestaven v kapitole 5.3.3

- **TherapistId** (Users) – vazba na terapeuta, který úlohu zadal
- **ExerciseId** (Exercise) – vazba na úlohu, která byla uživateli přiřazena
- **StatusId** (Status) – vazba na stav konkrétní úlohy
- **AssignDate** (datum) – datum přiřazení úlohy pacientovi
- **LastAccessDate** (datum) – datum posledního přístupu pacienta k úloze
- **IsActive** (bool) – indikátor aktivního přiřazení úlohy

Package

Ve specifikaci požadavků je uvedeno, že se úlohy budou pacientům přiřazovat prostřednictvím balíků, které můžou obsahovat libovolné množství úloh. Z tohoto důvodu byla vytvořena rozkladová tabulka „ExerciseAssignPackage“, která uchovává identifikátory z tabulek „ExerciseAssign“ a „Package“. Každý balík v aplikaci je definován následující množinou atributů.

Pacienti mají viditelné pouze takové balíky k procvičení, které jsou nastaveny jako aktivní a zároveň se nalézají ve stavu *nové* nebo *zpracovávané*.

- **Id** (číslo) – jednoznačný identifikátor balíku
- **Name** (text) – název balíku
- **StatusId** (Status) – identifikátor stavu balíku
- **TherapistId** (text) – identifikátor terapeuta, který balík založil
- **PatientId** (text) – identifikátor pacienta, kterému je balík určený
- **CreateTimeUtc** (datum) – datum vytvoření balíku
- **LastAccessTimeUtc** (datum) – datum posledního přístupu k balíku
- **IsActive** (bool) – indikátor aktivity balíku.

ExcerciseResult

Tabulka určená pro ukládání výsledků z jednotlivých cvičení, která neobsahuje vazbu přímo na určitou úlohu, ale na záznam z tabulky „ExerciseAssign“, kde je úloha přiřazena konkrétnímu pacientovi. Pokud je daná úloha provedena pacientem vícekrát, pak jsou od sebe odlišeny sekvenčním číslem provedení. Tímto způsobem ukládání je pak nad daty umožněno provádět různé analýzy. Tabulka obsahuje následující atributy.

- **Id** (číslo) – primární klíč v tabulce
- **ExerciseAssignId** (ExerciseAssign) – vazba na záznam, který jednoznačně identifikuje přiřazení úlohy konkrétnímu pacientovi
- **Row** (číslo) – pořadové číslo výstupního parametru z úlohy[12]
- **Value** (text) – konkrétní realizace výstupního parametru z úlohy
- **Sequence** (číslo) – pořadové číslo ukládané úlohy[13]
- **TimeUtc** (datum) – datum provedení úlohy

5.2.5 Ostatní tabulky

V této podkapitole si uvedeme zbývající tabulky, které jsou s těmi předchozími propojeny vazbami.

ExerciseAssignPackage

Jedná se o typ rozkladové tabulky uchovávající informaci o balících a jejich úlohách. Součástí této tabulky je také pořadové číslo úlohy v rámci balíku, aby mohlo dojít k jejich řazení.

PatientFeedback

Jedná se o tabulku, ve které budou ukládány odezvy pacientů na jednotlivé balíky. Tímto způsobem bude pacientům umožněno hodnocení právě dokončeného balíku, které poslouží terapeutovi pro tvorbu dalších balíků. Každá zpráva obsahuje informaci o náročnosti, počtu úloh a krátkou zprávu s hodnocením balíku.

AdministratorData, TherapistData a PatientData

Jedná se o tabulky, které udržují dodatečná data, která jsou přidružena ke konkrétním rolím. Tabulky obsahují pouze atributy s primitivními datovými typy a vazbu na tabulku Users. Jedinou výjimkou je tabulka PatientData, která obsahuje dále vazby na tabulky TherapistReport a PatientDiagnosis, které si představíme v další podkapitole.

[12] Z každé úlohy ukládáme několik výstupních parametrů. Tento Atribut uvádí jeho pořadové číslo.

[13] V případě opakovaného provádění úlohy dochází k iterování této hodnoty.

TherapistResport a PatientDiagnosis

Jedná se o tabulky, které jsou spjaty s konkrétní terapeutem a pacientem. Terapeut tak má možnost do systému přidat pacientovu diagnosu a krátký report o pokroku pacienta.

Roles a Status

Jedná se o velmi jednoduché tabulky, které převážně představují výčtový typ, kterých mohou role a statuty nabývat. K naplnění těchto tabulek dochází při sestavování aplikace a jejich data jsou v průběhu celého běhu aplikace neměnná.

ListFilter

Jakmile dojde k nasazení aplikace do ostrého provozu, dá se předpokládat, že množství jednotlivých objektů v rámci všech tabulek rychle poroste. Z tohoto důvodu jsem se rozhodl pro všechny editory, které mají za úkol zobrazit seznam svých objektů, naimplementovat filtry. Každý uživatel v systému bude mít v těchto tabulkách jeden záznam s jejich posledním nastavením. Nyní uvedu seznam tabulek, které jsou pro filtry určeny. Jejich konkrétní realizaci popíši v kapitole 5.3.

- TranslationListFilter
- ProgramListFilter
- ExerciseListFilter
- PackageListFilter
- PatientAssignListFilter

Comment, ErrorLog, ChangeLog

Jedná se o tabulky, které slouží pro vývojářské účely. Do tabulky *Comment* přispívají přihlášení uživatelé hlášením chyb, které v průběhu práce se systémem našli nebo návrhem na možná vylepšení aplikace. Administrátor aplikace je na tyto záznamy upozorňován prostřednictvím e-mailu. Tabulka *ErrorLog* je určena pro ukládání chybových průběhů, které v aplikaci nastanou a tabulka *ChangeLog* je určena pro vkládání zpráv o novinkách v aplikaci pro její uživatele.

5.3 Popis jednotlivých modulů

V této kapitole si popíšeme všechny důležité moduly, ze kterých se tato webová aplikace skládá. Programovací styl vývoje této aplikace je založený na návrhovém vzoru MVC, proto jsem se rozhodl jednotlivé moduly popisovat v rámci těchto tří vrstev.

Abych se v rámci každého modulu neopakoval ve vysvětlování principu jeho fungování, tak nejprve popíši obecný modul. V jednotlivých podkapitolách budu dále vysvětlovat pouze důležité části a odchylky od obecného modulu.

5.3.1 Popis obecného modulu

Kontroler

Základem každého modulu je jeho kontroler, který přijímá požadavky z okolí, tedy z uživatelského vstupu nebo programově vyvolaných akcí, a odpovídá na ně prostřednictvím jeho služby. Každý kontroler má právě jednu službu, ale zároveň je umožněno, aby jednotlivé služby byly využívány napříč celou aplikací.

Při získávání dat ze serveru prostřednictvím kontroleru dochází k zavolání služby, která v případě potřeby získá z databáze požadovaná data, které následně převede do formy, ve které se předají do pohledu, tzv. „ViewModelu“, který byl popsán výše. V případě dotazu na server z důvodu manipulace s daty je službě předán právě tento ViewModel, ze kterého služba získá databázový model a provede následnou modifikaci.

V rámci webových aplikací je velmi důležité zajistit, aby nedocházelo ke zpřístupnění nepovolených operací neautorizovaným osobám. Pro tento účel v aplikaci používám „Identity Framework“, který zajišťuje přihlašování do aplikace a kontrolu přístupových práv. Všechny kontrolery jsou zabezpečeny přístupovými rolemi a pouze některá volání na server (například přístup k volně dostupným informacím/datům) jsou přístupná i pro anonymní uživatele. Autorizace neprobíhá pouze na úrovni kontroleru, ale také na úrovní pohledů (tj. co nemá být přístupno, je skryto) a služeb (ověřování identity aktuálně přihlášeného uživatele před manipulací s daty).

Model

Model je základní entita pro jakoukoli manipulaci s daty. Modely jsou ve své podstatě mapovány na jednotlivé tabulky uvnitř databáze. Stejným způsobem, jako jsou v rámci databáze vytvořeny vazby mezi jednotlivými ta-

bulkami, tak jsou vytvořeny stejné vazby mezi modely. Takovému přístupu se říká „code first“ a je umožněn `EntityFrameworkem`, který v této práci používám pro přístup k datům v databázi.

Očekávaným výsledkem dotazu na server je ve většině případů nějaký HTML kód, který nám zobrazí požadovaný obsah. V technologii ASP.NET MVC dochází k vygenerování HTML kódu prostřednictvím pohledu[14], do kterého vstupuje model konkrétního typu. Na základě vstupního modelu získáme konkrétní podobu požadovaného pohledu (HTML kód).

Ve většině případů si v pohledech nevystačíme s obecnými modely, tj. s daty/záznamy, které jsou uloženy přímo v databázi. Z tohoto důvodu je pro vývojáře výhodné vytvořit zcela nový model, tzv. ViewModel, který v sobě bude mít veškeré atributy, které v rámci pohledu potřebuje. Tyto atributy mohou být podmnožinou požadovaného modelu, nebo mohou obsahovat data z vícero modelů. Vytvářením ViewModelů také dochází k úspoře datových prostředků, protože si z databáze získáme pouze taková data, která nás zajímají. V případě potřeby přidání některého z atributů jej jednoduše přidáme a neovlivníme tak žádnou jinou část v aplikaci.

Dále je doporučováno vytvářet ViewModely nejen pro jednotlivé pohledy, ale také pro všechny databázové tabulky. Toto platí hlavně z důvodu využití EntityFrameworku, který při dotazu na získání konkrétního záznamu v databázi získá nejen požadovaný záznam, ale také objekty, se kterými má nějakou vazbu. Tyto dodatečné objekty ve většině případů jsou zasílány zbytečně, čímž zpomalují celý systém.

V rámci jednotlivých modulů nebudu uvádět detailní popis jednotlivých modelů, protože jejich význam byl již popsán (viz podkapitola 5.2).

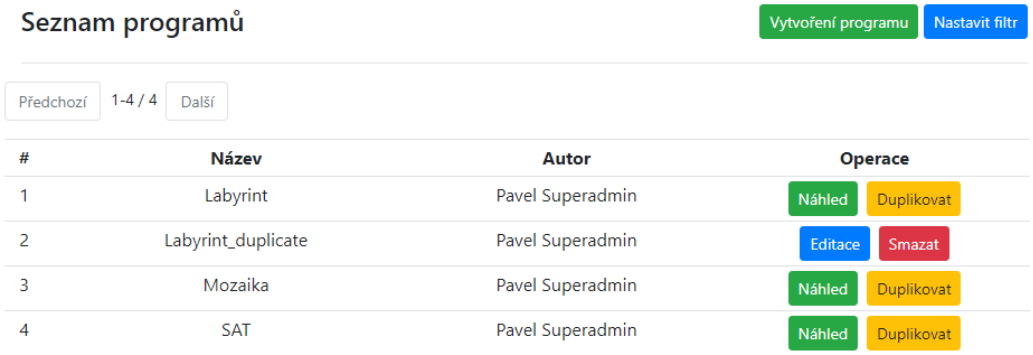
Pohled

Hlavní úkolem webových technologií je poskytovat uživatelům pomocí webového prohlížeče požadované vizuální výsledky. Tohoto dosahuji prostřednictvím pohledů, které získávají svoji konkrétní podobu prostřednictvím přidruženého modelu. Každý pohled může - ale nemusí - tento model obsahovat. Pokud jej postrádá, pak se z pohledu stává pouze statická stránka.

Každý modul se skládá minimálně ze tří pohledů, tj.:

- **Vytvoření nové entity**, při které ve většině případů dochází pouze k vyplnění povinných údajů. Ostatní atributy je možné doplnit v její

[14]Pohled je ve své podstatě HTML šablona, do které se doplňují informace obsažené ve vstupním modelu.



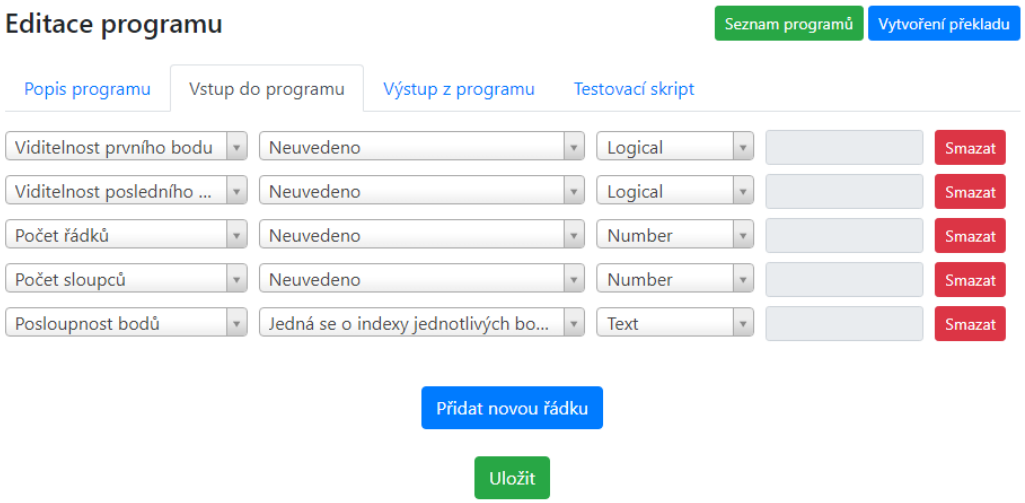
Obrázek 5.3: Seznam vytvořených programů.

editaci.

- **Seznam entit**, které byly dosud v daném modulu vytvořeny (např. seznam všech programů viz obr. 5.3). Nad tímto seznamem bude umožněno několik operací, mezi které patří náhled, editace, duplikace, mazání, skrývání či aktivace daného objektu. Nad tímto seznamem je umožněno provádět filtraci napříč všemi atributy dané entity. Proto byly v databázi vytvořeny tabulky, které u každého uživatele uchovávají poslední nastavení filtru. Filtr je dostupný pod tlačítkem „Nastavit filtr“ v levém horním rohu editoru. Po nastavení filtru dojde k jeho aktualizaci v databázi a zobrazení příslušných záznamů. Nastavení filtru je individuální pro každého uživatele a při opětovném navštívení editoru se aplikuje jeho aktuální verze. Pokud bychom chtěli vyfiltrovat položky, které odpovídají vícero podmínkám v rámci jednoho uživatelského vstupu, pak jednotlivé podmínky oddělíme středníkem[15]
- **Náhled/editace** konkrétního objektu. Ve své podstatě se jedná o dvě totožné akce s tím rozdílem, že v případě editace je uživateli umožněno provádět nad konkrétním modelem změny. V případě náhledu je model zobrazen pouze pro čtení. Pro ukázku jsem se rozhodl ukázat jednu ze záložek editoru programu, resp. editor vstupních parametrů do daného programu (viz obr. 5.4).

Tímto jsme si představili způsob implementace základního modulu. Nyní si popíšeme konkrétní moduly. Nejdříve zjednodušeně popíši základní modely, které mají své ekvivalenty v podobě databázových tabulek, jak bylo popisováno výše (kapitola 5.2). Následně si popíšeme důležité pohledy a na

[15]Pokud chci pouze uživatele, jejichž jméno je Pavel nebo Petr, pak filtr na jméno nastavím na hodnotu „Pavel; Petr“.



Obrázek 5.4: Editor vstupních parametrů do programu.

závěr popíši kontroler a jeho odpovědi na uživatelské či programově vyvolané dotazy.

5.3.2 Správa neurorehabilitačních programů

V této kapitole si popíšeme tvorbu a správu neurorehabilitačních programů (dále jen programů) napříč celou MVC architekturou.

Jedním z hlavních důvodu vývoje této aplikace je, aby v rámci ní bylo možné co nejjednodušeji vytvářet nové programy, ze kterých se následně vytvářejí úlohy pro pacienty. Proto jsem vytvořil modul, který toto umožní s nulovým zásahem do programového kódu a bez nutnosti následného opětovného sestavení aplikace na webovém serveru.

Každý program má přímo definovaného svého vlastníka, ovšem počítá se s tím, že veškeré programy budou volně přístupné všem uživatelům s právem zadávat nové úlohy. Program má stavový bit, který rozhoduje o tom, zda je veřejný (může jej používat kdokoli) nebo je privátní (může jej použít pouze jeho autor). Při vytvoření nového programu je program neaktivní a pouze jeho autor jej může používat. Zároveň je mu také povoleno vytvořit z tohoto programu novou úlohu a následně ji otestovat. Po otestování programu jej autor označí za aktivní a veřejný, čímž jej zpřístupní ostatním uživatelům. Ti poté mohou na základě tohoto programu vytvářet nové úlohy.

Modely

Základními modely v rámci tohoto modulu jsou `Program`, `ProgramRecipe`, `ProgramResult`.

- `Program` – Atributy tohoto modelu jsou jeho identifikátor, název, popis, vlajka aktivity a tři kolekce pro instance modelů `ProgramRecipe`, `ProgramResult` a `Exercise`[16]

- `ProgramRecipe` – Pomocí tohoto modelu můžeme dynamicky přidávat vstupní parametry do testovacího skriptu, který je jádrem konkrétního programu. Dále obsahuje dvě reference na objekt typu `Translation`, který bude popsán v kapitole 5.3.6.

- `ProgramResult` – Jedná se o model podobný předchozímu, který je ovšem určený pro definování výstupních parametrů z testovacího skriptu. Pomocí tohoto modelu jsme schopni určit počet výstupních parametrů, jejich význam a datové typy, které jsou důležité pro následnou analýzu dat. Model obsahuje také dvě reference na objekty typu `Translation`.

Pohledy

Základním pohledem tohoto modulu je pohled na seznam všech vytvořených programů (viz obr. 5.3), který umožňuje filtraci přes jeho atributy pomocí modálního okna. Tento pohled umožňuje na základě jejich stavu několik základních operací:

- **Vytváření** nového programu je operace, kdy je nutné definovat pouze název nového programu a jeho popis. Oba atributy jsou zároveň validovány, zda obsahují dostatečný popis znaků.

- **Editace** je základní operace, ve které dochází k sestavení konkrétního programu. Program je editovatelný pouze jeho vlastníkem, a to do té doby, dokud z něj není vytvořena úloha, resp. neexistuje aktivní přiřazení úlohy k pacientovi[17]. Poté se dá úloha pouze duplikovat, nebo na ní nahlížet.

- **Náhled** nedovoluje žádnou manipulaci s programem. Slouží pouze pro nahlížení vstupních a výstupních parametrů.

- **Duplikace** je operace, která vytvoří nový program, jehož funkční předpis bude totožný. Při pohledu do modelu programu zjistíme, že obsahuje i kolekci úloh, které na základě duplikovaného programu vznikly.

[16] Bude popsáno v rámci modulu správy úloh (viz kapitola 5.3.3)

[17] Může existovat i neaktivní přiřazení, které ovšem obsahuje data z provedení této úlohy. Tento stav může nastat v případě, že terapeut odebere pacientovi již splněnou úlohu. V budoucnu toto přiřazení můžeme chtít obnovit, čímž bychom obnovili i původní data.

Tato kolekce se tedy pochopitelně do nového programu nezduplikuje. Součástí nového programu také nebude původní testovací skript, protože se bude jednat o nový program, který se starým skriptem nebude fungovat.

- Poslední operací je **mazání** vytvořených programů. Tuto operaci lze provést pouze v případě, když je daný program editovatelný. Editovatelnost programu jsem uvedl výše. Z toho plyne nemožnost odebrání programu, ze kterého již byla dokončena libovolná úloha[18].

Dalším pohledem správy programů je jeho editor, který se skládá ze čtyř záložek. Tyto čtyři záložky obsahují okna, díky kterým lze dodefinovat existující program.

První záložka je určena k popisu programu. Uživatel má zde možnost nastavit jeho název, krátký popis jeho účelu a stavové hodnoty, které určují zda jej mohou použít ostatní terapeuti.

Ve druhé záložce definujeme vstupní parametry programu (viz obr. 5.4), které představují vstupní hodnoty do testovacího skriptu. Zde můžeme dynamicky přidávat či mazat řádky, tj. jeden řádek představuje jeden vstupní parametr do skriptu. Každý řádek je složen z *názvu řádky*, jejího *popisu*, *typu dat* a v případě výběru dat ze seznamu, tak *hodnoty*, což představuje výčet možných prvků. Název řádku a její popis je realizovaný pomocí *překladů*, které budou představeny později, viz kapitola 5.3.6.

Ve třetí záložce definujeme podobným způsobem výstup z testovacího skriptu, který slouží především pro definování dat, které budeme po dokončení úlohy ukládat do databáze. Nad těmito daty budou následně prováděny různé analýzy pro ohodnocení pokroku pacienta.

Čtvrtá záložka slouží pro nahrání testovacího skriptu, který přijímá vstupní hodnoty ze druhé záložky a na výstupu jsou data ze třetí záložky. Vytvoření testovacího skriptu je jediná část, která vyžaduje programátorské zkušenosti. Nahrávání testovacího skriptu je umožněno jeho autorovi provádět kdykoli, a to i v tom případě, že daný program již není editovatelný. Nově nahraný skript ovšem musí splňovat nadefinované vstupní a výstupní parametry.

Součástí editoru je také možnost přidání nového překladu, který by nám v případě definování jednoho ze vstupních či výstupních polí chyběl. Toto přidání se provede pomocí modálního okna, a tak se nemůže stát, že bychom vinou přesměrování do jiného editoru přišli o rozpracovaný program. Celá

[18]Tímto se nepočítá provedení úlohy při jejím testování autorem, ze které se data neukládají.

operace se provádí pomocí ajaxového dotazu na server, ve kterém dochází k předání jednotlivých lokalizací.

Kontroler

Tento modul má charakter obecného kontroleru, který jsem popisoval výše v rámci této kapitoly. Co bych mohl zmínit, je způsob přidávání nových řádek jak u přidávání vstupních, tak výstupních parametrů. Toto se děje pomocí ajaxového volání, během kterého dojde k vytvoření modelu této řádky a následnému vygenerování HTML kódu. Do editoru se tak pošle pouze segment HTML kódu, který se propíše do databáze až při konečném uložení celého programu. Způsob vyplňování textů jednotlivých řádků probíhá také pomocí AJAXové technologie, tj. uživatel zadá část hledaného překladu a kontroler vrátí seznam všech překladů z databáze, které obsahují zadanou posloupnost znaků. Po nalezení hledaného překladu se do modelu uloží jeho identifikátor a na základě nastavené lokalizace se zobrazí při tvorbě úlohy daný překlad.

5.3.3 Správa úloh

Jedná se o modul, pomocí kterého terapeuti vytvářejí svým pacientům sady testovacích úloh. Před vytvořením nové úlohy musí být prototyp této úlohy vytvořen ve správě programů. Následně může dojít k využití předpisu programu a po vyplnění příslušných vstupních polí dojde k vytvoření úlohy. Přiřazení úlohy pacientovi pak probíhá prostřednictvím balíků, které budou popsány ve správě balíků (viz podkapitola 5.3.4).

Modely

Základními modely tohoto modulu jsou:

- `Exercise` – Jedná se o základní model, který představuje úlohu.
- `ExerciseAssign` – Model pro ukládání přiřazení úloh k pacientům.
- `ExeciseData` – Model, jehož záznamy jsou konkrétními vstupními parametry do vytvořené úlohy, tj. co zadaná úloha, to množina těchto parametrů.
- `ExerciseResult` – Model pro ukládání výsledků z provedených úloh. Každé jedno elementární spuštění a dokončení úlohy má v této tabulce záznam. Každý záznam obsahuje informaci o pořadovém čísle konkrétního výsledku a výsledku samotnému.

- `Package` - Model, do kterého jsou ukládány úlohy přiřazené pacientovi. Pacientovi se následně přidělí ke cvičení celý balík, nikoli jednotlivé úlohy.

Pohledy

Aby mohlo dojít k vytvoření nové úlohy, je nejprve nutné pomocí příslušného editoru tuto úlohu založit. Při zakládání úlohy je nutné definovat její název, popis a vybrat ze seznamu její typ - program. Základním pohledem je u tohoto modulu opět seznam všech vytvořených entit, tedy úloh, které jsou přístupné danému uživateli. Nad tímto seznamem jsou dostupné základní filtrace, které byly popsány u obecného modulu. Nad jednotlivými položkami seznamu jsou přístupné podle stavu jednotlivých záznamů následující operace:

- **Editace** je základní operací, při které dochází k dodefinování úlohy, která byla založena nad jedním z programů. Editace je dostupná pouze v případě, že ji chce provést její autor a zároveň nebyla dosud přiřazena žádnému z uživatelů, resp. dokud neexistují z úlohy nějaké výsledky[19].
- Další operací nad úlohou je **náhled**, kdy je možné měnit pouze příznak veřejnosti a aktivity úlohy. Ostatní možnosti jsou nedostupné. Tato operace se zpřístupní v případě nemožnosti editace.
- **Smazání** úlohy je možné pouze, pokud daná úloha je editovatelná. Tyto důvody vyplývají z popisu editovatelnosti, kterou jsem popsal výše.
- **Duplikovat** úlohu je přístupné v případě, když je daná úloha v systému již využívána. Tímto vznikne nová, editovatelná úloha.

Dalším pohledem ve správě úloh jsou seznamy určené pro pacienty, ve kterých mají možnost sledovat svoje úlohy napříč všemi jejich stavy. Tyto seznamy jsou zobrazeny v samostatných záložkách. Jedná se tedy o seznamy úloh, které jsou v následujících stavech a mají umožněny tyto operace.

- **Nová a probíhající úloha**
  - Spustit

[19]Může nastat situace, že existuje záznam o neaktivním přiřazení, které ovšem bude mít výsledky v tabulce `ExerciseResult`. Poté je nežádoucí úlohu editovat, protože již obsahuje výsledky.

- **Dokončená úloha**
- **Zkontrolovaná a uzavřená úloha**
  - Zobrazit výsledky

Dalším pohledem pro správu úloh je jejich editor. Editor se skládá celkem ze dvou záložek, kdy první je určena pro náhled základních informací o úloze a zda je úloha veřejná či nikoli. Mezi tyto informace patří její název, typ programu, krátký popis a datum vytvoření. Veřejná úloha je taková, kterou může využít jakýkoli terapeut, aniž by ji musel sám vytvořit. Ve druhé záložce definujeme zadání samotné úlohy.

5.3.4 Správa balíků

Pokud chce uživatel přiřadit pacientovi v aplikaci libovolnou z úloh, pak je mu to umožněno pouze prostřednictvím balíku, do kterého úlohu vloží. Spouštění úloh bez použití balíku je přípustné pouze pro autory úloh z důvodu jejich otestování nebo spuštění ukázkových úloh určených primárně pro neregistrované uživatele.

Modely

Základními modely v rámci tohoto modulu jsou:

- **Package** – Jedná se o popis konkrétního balíku pro jeho jednoznačnou identifikaci a určení, komu náleží.
- **ExerciseAssignPackage** – Model, pomocí kterého jsme schopni zjistit, které úlohy se v daném balíku nalézají včetně jejich pořadí spouštění.
- **PatientFeedback** – Model, který umožní ukládat pacientovo hodnocení konkrétního balíku.

Pohledy

Abychom mohli balíky používat, tak je nejprve nutné je založit. Toto je možné pomocí **editoru pro vytváření balíku**, ve kterém je nutné specifikovat jejich název a pacienta, kterému je balík přiřazen. Potvrzením těchto vstupů dojde k vytvoření balíku, který je zpočátku neaktivní. Po jeho naplnění úlohami jej terapeut označí za aktivní, čímž dojde k jeho zpřístupnění konkrétnímu pacientovi.

Další důležitý pohled je určený pro zobrazení **seznamu všech balíků**, které patří konkrétnímu terapeutovi. Nad tímto seznamem je dostupná filtrace podle jména balíku, jména pacienta a aktivity balíku. Dále jsou nad jednotlivými položkami umožněny následující operace:

- Hlavní operací je **editace**, která bude popsána v samostatném odstavci níže.
- Balík je také možné **duplikovat**, čímž se vytvoří nový balík pro jiného pacienta, kterého během duplikace uživatel zadá. Takto vytvořený balík bude zpočátku neaktivní. Po úspěšné duplikaci bude terapeut přesunut do jeho editace.
- Terapeut má také možnost balík **smazat**. Tímto krokem nedojde ke smazání konkrétních výsledků, ale pouze této struktury (balíku), která úlohy obaluje.

Editace balíku se skládá celkem ze tří záložek. V **první záložce** je terapeutovi umožněno nastavit jeho aktuální stav a příznak jeho aktivity. Pacientovi je umožněno vidět pouze balíky, které jsou aktivní a zároveň se nalézají ve stavu *nový* nebo *zpracovávaný*. V rámci **druhé záložky** je terapeutovi umožněno přidávat do balíku nové úlohy a odebírat, měnit pořadí či analyzovat stávající úlohy. V rámci poslední **třetí záložky** má terapeut k dispozici zpětnou vazbu od pacientů k danému balíku, prostřednictvím seznamu a tlačítka pro náhled jednotlivých záznamů pomocí modálního okna.

5.3.5 Analýza výsledků

Jedná se o modul, pomocí kterého je terapeutům umožněno vyhodnocení splněných úloh od svých pacientů. V rámci tohoto modulu nebudu popisovat žádné modely, jelikož zde dochází k použití již popsaných modelů z kapitoly správy úloh (viz 5.3.3).

Pohledy

Základní pohledem tohoto modulu je **seznam všech úloh**, které daný terapeut přiřadil svým pacientům. Aby terapeut mohl najít úlohu, která ho zajímá, tak je v tomto pohledu umožněna filtrace pomocí tří seznamů, které obsahují následující prvky.

- **Seznam programů**, jejichž úlohy mají být zobrazeny v seznamu.
- **Seznam úloh**, které mají být zobrazeny.

- **Seznam pacientů**, jejichž výsledky chceme analyzovat.

Jednotlivé seznamy jsou na sobě závislé, tzn. že pokud vybereme nějakou hodnotu v seznamu programů, tak v seznamu úloh budou zobrazeny pouze úlohy příslušné této množině programů. Stejným způsobem jsou na sobě závislé seznamy úloh a pacientů. V rámci každého seznamu lze vybírat více hodnot. Tímto způsobem dojde k zobrazení požadované množiny úloh a kliknutím na tlačítko analyzovat u konkrétního záznamu dojde k přesměrování do analýzy úlohy.

Další pohled realizuje samotnou **analýzu konkrétní úlohy**, kterou pacient splnil. Uživatel má úlohu přístupnou do té doby, dokud jí terapeut nenastaví status o jejím dokončení. Do této doby může pacient tuto úlohu spouštět a terapeut sledovat průběh jejího plnění.

Pohled pro analýzu výsledků se skládá z tabulkového výpisu výstupních parametrů jednotlivých spuštění dané úlohy. V dalším vývoji aplikace bude žádoucí, aby byl tento textový výpis rozšířen o grafické zobrazení.

5.3.6 Správa překladů

V průběhu návrhu bylo potřeba počítat s tím, že aplikace bude podporovat více jazyků. V rámci základní implementace, tj. veškerých editorů, informačních textů apod., které jsou součástí aplikace při sestavení nového řešení aplikace, není problém tyto překlady předem definovat. Problémem je zavedení nových překladů v průběhu životního cyklu aplikace, kdy musel být navrhnut jiný princip jejich přidávání, které se týká hlavně definování nových testovacích programů.

Princip vytváření nového překladu, zobrazení seznamu překladů a operace nad jednotlivými položkami v seznamu nevykazuje téměř žádné odchylky od obecného modulu.

Při **vytváření** je potřeba dávat pozor na to, aby došlo k vyplnění hlavní jazykové mutace a textovému atributu, který jednoznačně identifikuje daný překlad[20] určený převážně pro analýzu výsledků. Na povinnost vyplnění musí být uživatel případně upozorněn.

Operace nad jednotlivými záznamy vycházejí z obecného modelu. **Editovat** lze libovolný překlad, ale v případě jeho aktivního použití je dobré editora upozornit, aby nedošlo ke změně jeho významu. **Duplikovat** lze

[20]Tímto se rozumí atribut „DbColumnName“ u modelu „Translation“ pro slovní definování překladu

libovolný již použitý překlad a **smazat** pouze překlad, který nebyl dosud nikde použit.

5.3.7 Správa uživatelů

Mezi další nezbytnou funkcionalitu patří správa uživatelů. V rámci tohoto modulu je umožněno nastavovat uživatelům aplikace jejich role a také provádět přiřazování pacientů k jejich terapeutům. V tomto modulu jsou vyřizovány i ostatní požadavky, které se týkají uživatelů aplikace.

Pokud vezmu v úvahu **administrační část** tohoto modulu, tj. správu uživatelských rolí a vytváření vazeb mezi terapeuty a pacienty, pak modul opět vychází z obecného popisu modulu. V prvním případě se administrátorovi zobrazí seznam uživatelů, které si může na základě jména filtrovat a následně modifikovat. V druhém případě se mu zobrazí seznam s vazbami s dynamickou možností odebírání[21] existující a přidávání nových vazeb.

5.3.8 Nastavení uživatelského účtu

V neposlední řadě je v aplikaci možnost nastavení svého vlastního účtu prostřednictvím editoru. Zde má uživatel možnost nastavení nového hesla, aktualizace osobních údajů a dalších nepovinných voleb.

[21] Ve skutečnosti se tato vazba nikdy neodebere, pouze se nastaví jako neaktivní a v případě opětovného přiřazení se zaktivuje.

6 Implementace

V této kapitole se blíže podíváme na způsob implementace aplikace. Bude zde popsána základní adresářová struktura projektu, která bude užitečná zejména pro vývojáře, kteří by případně v tomto projektu pokračovali.

6.1 Adresářová struktura projektu

Adresářová struktura projektu vychází z aplikačního frameworku ASP.NET MVC, jehož hlavními adresáři jsou `Models, Views, Controllers`, které si zde v krátkosti popíšeme. Poté popíši i zbývající adresáře.

6.1.1 Controllers

Jedná se o adresář, ve kterém jsou udržovány veškeré kontrolery v aplikaci. Aplikace je napsána stylem, že každý modul, resp. funkčně samostatná část aplikace, má svůj vlastní kontroler, který zpracovává dotazy. Modulů je celkem devět, které si popíšeme v tabulce 6.1. Jednotlivé kontrolery mají v průměru 210 řádek programového kódu.

6.1.2 Services

V tomhle adresáři jsou umístěny služby, které jsou mapovány na své kontrolery. Tento přístup umožňuje vytvářet přehledné kontrolery, jejichž funkcionalita je obsažena v konkrétní službě, která má přístup k datům v databázi. Služby obsahují v porovnání s kontrolery mnohem více programového kódu a to průměrně 550 řádek.

6.1.3 Models

V tomto podadresáři se nacházejí veškeré modely, které jsou v aplikaci použity. Modely jsou dvojího typu a jsou podle toho rozděleny ve složkách. Prvním typem jsou modely, které jsou mapovány na databázové tabulky. Ty jsou používány pro získávání a ukládání dat. Druhým typem jsou ViewModely, které vstupují do pohledů. Tyto pohledy se nalézají v adresáři ViewModels.

Celkový počet všech modelů je 85.

Název	Určení
Account	Administrační záležitosti typu registrace, přihlášení, nastavení hesla, změna osobních údajů, ověření e-mailu, ...
User	Kontroler zabývající se operacemi nad uživateli typu vyhledávání uživatelů s danou rolí, vytváření vazeb terapeut-pacient, ...
Home	Jedná se o obecný kontroler, který slouží převážně pro zobrazení prezentace webové aplikace a funkcionality, která přímo nepatří do ostatních kontrolerů.
Analysis	Kontroler pro vyřizování požadavků související s analýzou výsledků.
Test	Kontroler, jehož úkolem je spouštět cvičení pro pacienty a ukládat jejich výsledky.
Program Exercise Package Translation	Kontrolery zajišťující převážně akce pro vytvoření programů, úloh, balíků a překladů.

Tabulka 6.1: Tabulka kontrolerů

6.1.4 Views

Zde jsou obsaženy pohledy, které jsou dále strukturovány do složek podle názvu jejich kontrolerů. Každý kontroler využívá také tzv. „PartialViews“ (pohledy, které generují pouze fragment HTML kódu, nikoli celou stránku), které jsou dále organizovány ve složce „EditorTemplates“ Celkový počet pohledů v aplikaci je 104.

6.1.5 Další adresáře

Ostatní důležité adresáře jsem se rozhodl popsat prostřednictvím přehledné tabulky (viz tab. 6.2).

6.2 Zdrojové kódy

Při psaní programového kódu byl dodržován kódovací standard jazyka C# a jmenné konvence pro co nejsnazší orientaci vývojářů v aplikaci. Všechny důležité části zdrojových kódů byly dostatečně okomentovány, aby byl u rozsáhlejších částí zřejmý jejich význam.

Název	Určení
Common	Adresář tříd, ve kterém jsou pomocné statické metody a výčtové typy, které se mohou používat kdekoli v aplikaci.
Content	Adresář určený pro soubory s kaskádovými styly a obrázky.
Resources	Adresář pro správu lokalizačních souborů
Scripts	Adresář pro správu JavaScriptových souborů.

Tabulka 6.2: Tabulka vedlejších adresářů v projektu

Zdrojové kódy této práce byly spravovány pomocí webového Git repositáře GitLab a jsou dostupné na adrese `https://gitlab.com/Skala/NeuroRehab.git`. Na této adrese je dostupná i dokumentace a soubory, které byly použity pro vývoj aplikace či jako zdroje do dokumentace.

7 Ověření funkčnosti

Tato kapitola se zaměřuje na poslední bod zadání, tj. otestování funkčnosti a bezpečnosti aplikace. Proto si v této kapitole uvedeme jakými způsoby se webové aplikace testují, jakým útokům mohou být vystaveny a jak jim lze předejít. Na závěr popíši výsledky tohoto testování.

7.1 Typy útoků na webové aplikace

V této podkapitole si představíme několik typů nejběžnějších útoků na webové aplikace, kterým se v rámci této práce snažím zabránit [18]. Dále si kapitole popíšeme jak těmto útokům v rámci frameworku ASP.NET zamezit.

- **DDOS útoky**
  Typ útoku, jehož cílem je vyřadit poskytovanou službu z provozu. DDOS (Distributed Denial of Service) útok přichází od více zdrojů, které posílají na server tak velký počet požadavků, že jej server není schopen vyřídit. Tím dojde k zahlcení a následnému pádu.

- **Cross-site Scripting (XSS)**
  Metoda napadení webových stránek využitím bezpečnostních chyb ve skriptech (především neošetřené vstupy), kdy útočník díky těmto chybám může do stránek podstrčit svůj vlastní kód.

- **Cross–site request forgery (CSRF)**
  Jedná se o typ útoku, ve kterém dojde k podvržení formuláře, který následně odešle data uživatele útočníkovi. Uživateli se tak může stát, že spolu s přihlášením se do aplikace odešle své přihlašovací údaje útočníkovi.

- **SQL injection**
  Typ útoku, při kterém se útočník může zmocnit databáze a spouštět nad ní libovolné dotazy. Útok spočívá v úpravě SQL dotazu, který je následně odeslán do databázového serveru. Tímto způsobem může dojít ke zneužití dat nebo jejich modifikaci.

Nyní si v jednotlivých odstavcích popíšeme, jak se dá těmto útokům zabránit prostřednictvím vybrané technologie.

Napadení databáze prostřednictvím **SQL Injection** je zabráněno díky použití Entity Frameworku, který sestavuje tzv. LINQ dotazy, které veškeré uživatelské vstupy převádějí do interních parametrů, které neumožní spuštění škodlivého kódu. Tímto je tento typ útoku vyloučen.

Útokům typu **DDOS** je zabráněno explicitně technologií ASP.NET díky úpravě nastavení v konfiguračním souboru. Jedním z jeho parametrů můžeme nastavit časnost příchodu požadavků od jednoho uživatele.

Třetím typem útoku je **CSRF**, kterému je zabráněno prostřednictvím validačních tokenů, které jsou přidávány do každého formuláře a následně validovány s příchodem každého požadavku do kontroleru. Pokud kontroler přijme nevalidní token, pak se kód v kontroleru nevykoná.

Posledním představovaným útokem je **XSS**, který lze vyřešit prostřednictvím kódováním nebezpečných znaků. Při implementaci pohledů pomocí tzv. Razor syntaxe dochází k tomuto kódování automaticky, čím se tento typ útoku vyřeší.

Výše jsem si představili přehled nejčastějších útoků na webové aplikace současnosti[18], které byly v rámci mé implementace vyřešené a otestované.

7.2 Uživatelské akceptační testy

Jedná se o typ testování, které je prováděno jejími reálnými uživateli. Tento typ testování má za úkol najít potenciální chyby především v jejích nejdůležitějších částech.

Na začátku vývoje této webové aplikace jsem implementoval mechanismus, který jejím uživatelům umožňuje vkládat do systému reporty o nalezených chybách nebo požadavcích na možná vylepšení aplikace. V průběhu jejího vývoje byl na tyto požadavky brán zřetel a po jejich vyhodnocení došlo k jejich implementaci či zamítnutí.

Testování bylo zaměřeno především na hlavní funkcionalitu aplikace, tedy na proces správy programů, úloh a balíků se cvičeními. Cílem je otestování potenciálních logických chyb (například, aby privátní úlohu mohl použít pouze její vlastník, zamezení použití neaktivního programu/úlohy a dalších chyb). V průběhu testování může nastat stav, kdy bude potřeba pozměnit návrh aplikace[1]. Cílem testování je také otestovat přehlednost a prezentaci aplikace. Za nejdůležitější scénáře v aplikaci považuji následující funkcionalitu.

[1]Funkcionalita byla sice naimplementována podle specifikace, ale její použitelnost není vhodná.

1. **Náhled na seznamy objektů**
   Jak programy, tak úlohy mají své příznaky, které určují, zda jsou aktivní a veřejné. Proto je důležité otestovat, zda každý z uživatelů vidí pouze takové objekty, ke kterým má přístup a zároveň odpovídají zadaným filtrům.

2. **Vytváření a editace objektů**
   Jedná se o otestování hlavních funkcionalit v aplikaci, mezi které patří správa programů, úloh, balíků a překladů. Při tomto testování je možné vycházet z uživatelské příručky (viz příloha C str. 108) a jejím cílem je odhalit co nejvíce chyb především v následujících scénářích.

   - U testování **programů** se jedná především o přidávání a odebíráním řádek - vstupních parametrů do programu - při definici vstupu do programu
   - Při testování **úloh** jde hlavně o správné vyplnění zadání úloh ve smyslu datových typů jednotlivých parametrů a ověření příznaku o aktivitě a přístupnosti programu/úlohy. Dále je dobré otestovat po nadefinování úlohy, zda jde úloha spustit.
   - Dále je vhodné otestovat **vytvoření a přidání úloh do balíku**. Součástí funkcionality je také duplikace balíku, kterou má terapeut povolenou pouze pro své pacienty.

Toto byl základní seznam nabízené funkcionality vytvořené aplikace, které jsou pro aplikaci klíčové. Uživatelé mohou podávat reporty na libovolné části aplikace od funkcionality až po vzhled jednotlivých elementů.

Při odevzdání této práce byly veškeré požadavky na změny vyřízené, opravené, čímž aplikaci považuji za funkční. Reportovací mechanismus zůstává i nadále aktivní, tudíž je možné přijímat nové požadavky na změny, které mohou být následně zapracovány do současného řešení.

7.3 Logování nestandardních stavů

Akceptační testování z předchozí kapitoly je pro odladění potenciální chyb a nevhodného grafického rozhraní nejlepším možným řešením. Pokud ovšem dojde k implementační chybě, pak si nevystačíme pouze s reportem od uživatelů. V tomto případě je nutné tento chybový stav zachytit a následně opravit bez nutnosti jejího zdlouhavého hledání ve zdrojovém kódu. Proto

jsem implementoval mechanismus, který každou zachycenou výjimku v aplikaci uloží do databáze. Vývojář pak jednoduše najde zdroj chyby, kterou následně opraví.

7.4 Další typy testů

Pro testování webových aplikací existuje celá řada nástrojů a přístupů, které sice nepoužiji, ale chtěl bych je pouze v krátkosti představit.

7.4.1 Automatizované funkční testování

Pro tento typ testování existuje velmi populární framework **Selenium**, se kterým již mám předchozí zkušenost. Framework umožňuje vytvořit sady testovacích skriptů, které dokáží simulovat reálného uživatele klikáním na elementy na stránce. Dle mého názoru není tento typ testu vhodný, protože dokáže projít pouze nadefinované scénáře, čímž není možné nalézt nové chyby, které se mohou vyskytovat v detailech, či upozornit na nevhodný návrh grafického rozhraní.

7.4.2 Testování databáze

Testy, které mají za úkol zajistit platnost vlastností jako je ACID (viz kap. 4.2), mapování dat, integritu dat, atd. Dále se dají otestovat uložené procedury a triggery.

Pokud v aplikaci dojde k chybě způsobené přístupem do databáze, pak jsme o ni informováni pomocí mechanismu logování nestandardních stavů. Administrátor je o této chybě automaticky informován a zajistí její nápravu.

7.5 Zhodnocení testů

V této části si popíšeme, jakým způsobem probíhalo uživatelské akceptační testování. Na testování se podílely celkem čtyři lidé a probíhalo ve dvou fázích.

První fáze testování probíhala v rámci raného vývoje jejím autorem, tedy mnou. V této fázi bylo nalezených chyb pochopitelně nejvíce.

Jakmile byly tyto chyby opraveny, tak se mohlo přejít do druhé fáze, tedy testování ostatními uživateli. V této fázi došlo k nasazení aplikace na webový server, aby byla přístupná i ostatním uživatelům.

Po odevzdání této práce následuje třetí fáze, kdy bude aplikace podrobena ostrému provozu.

Testování přístupu k funkcionalitám dle jednotlivých rolí dopadlo dle očekávání. Uživatelům, kteří se chtěli dostat modifikací URL adresy k nepovoleným akcím, byl přístup zamítnut a došlo k přesměrování na přihlašovací formulář, což je žádaný scénář. Otestovaný byl i případ, že uživatel sice požadovanou roli má, ale chce přistoupit k cizím datům. V tomto případě se mu nezobrazí žádný obsah.

Dále byla aplikace podrobena testu na výše popsané útoky, jejichž výsledky byly pozitivní, tedy že aplikace útoky odrazila a škodlivý kód se nevykonal.

7.5.1 Zjištěné problémy

Jakmile došlo k nasazení aplikace na webový server, tak se u seznamů úloh a programů zobrazovaly objekty, které daný uživatel nevytvořil. Jednalo se o chybu v implementaci při filtrování objektů pro daného uživatele. Většina nalezených chyb, které uživatelské testy prokázaly, byly tohoto charakteru, protože při implementaci docházelo k její ověření pouze jedním uživatelem, vývojářem.

8 Závěr

Cílem této diplomové práce bylo prozkoumat a analyzovat současné nástroje zabývající se neurorehabilitací. Na základě této analýzy jsem specifikoval nové řešení, které bylo konzultováno se specialistou v oboru neurorehabilitace. Dalším cílem bylo navrhnout a za pomoci vhodného webového frameworku toto řešení realizovat a otestovat.

V první části práce jsou popsány a porovnány čtyry řešení zabývající se neurorehabilitací, které odpovídali dvěma kategoriím. Do první kategorie patří nástroje Lumosity a HAPPYneuron, které se orientují na pacienty prostřednictvím algoritmů, předchozí zkušenosti a osobním preferencím. Do druhé kategorie spadá systém NP3 a její webové řešení, které upřednostňuje individuální přístup terapeutů ke každému pacientovi. Jelikož pacienti s vážnými neurologickými poruchami potřebují hlavně individuální péči, tak bude tato práce vycházet ze systému NP3.

Další části práce se zabývám její specifikací a návrhem. Návrh aplikace považuji za stěžejní část diplomové práce, protože bylo potřeba navrhnout aplikaci, která bude snadno rozšiřitelná a hlavně bezpečná, protože v ní budou spravována citlivá data pacientů. Proto bylo vytvořeno několik samostatných modulů, které pokrývají veškerou specifikovanou funkcionalitu.

Výstupem této diplomové práce je tedy analýza současného stavu neurorehabilitace a aplikace, která tuto neurorehabilitaci umožňuje prostřednictvím webového prohlížeče. Řešení se skládá ze dvou částí, které na sebe navazují. První částí je vytvoření samotného programu, kde je nutné nejen vytvořit předpis tohoto programu, ale také dodat skript, který dokáže daný program spustit. Tj. programy mohou vytvářet i neprogramátoři, ale musí jim být poskytnut testovací skript. Druhou částí je vytvoření úloh, které jsou vytvořeny cílovým pacientům na míru. Poté je pacientům umožněno provádět sady cvičení.

Testování webové aplikace probíhalo podle očekávání. Uživatelé se podíleli na vývoji řadou připomínek o drobných chybách a nedostatcích, které byly následně opraveny.

Myslím si, že se mi podařilo úspěšně splnit všechny body zadání. Vytvořil jsem robustní a funkční webovou aplikaci, která bude dobře sloužit jak terapeutům, tak hlavně pacientům.

Literatura

[1] *HAPPYneuron* [online]. 2019. [cit. 2019/04/15]. Dostupné z: http://www.happy-neuron.com/.

[2] *HAPPYneuron Pro: Rehabilitation Program – expert review* [online]. Dostupné z: https://psyberguide.org/expert_review/scientific-brain-training-sbt-pro-rehabilitation-program-expert-review/.

[3] Enhancing Nervous System Recovery through Neurobiologics, Neural Interface Training, and Neurorehabilitation. *Frontiers in Neuroscience*. 2016, 10, s. 584. ISSN 1662-453X. doi: 10.3389/fnins.2016.00584. Dostupné z: https://www.frontiersin.org/article/10.3389/fnins.2016.00584.

[4] *How to select a web framework* [online]. 2019. [cit. 2019/04/17]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Web_frameworks#How_to_select_a_web_framework.

[5] *What is AJAX?* [online]. Tutorials Point, 2019. [cit. 2019/05/08]. Dostupné z: https://www.tutorialspoint.com/ajax/what_is_ajax.htm.

[6] *Bootstrap* [online]. 2019. [cit. 2019/05/08]. Dostupné z: https://getbootstrap.com/.

[7] *Bootstrap Get Started* [online]. W3Schools.com, 2019. [cit. 2019/05/08]. Dostupné z: https://www.w3schools.com/bootstrap/bootstrap_get_started.asp.

[8] *What is CSS* [online]. W3C, 2019. [cit. 2019/05/07]. Dostupné z: https://www.w3.org/standards/webdesign/htmlcss#whatcss.

[9] *What is Entity Framework?* [online]. Entity Framework Tutorial, 2019. [cit. 2019/05/08]. Dostupné z: https://www.entityframeworktutorial.net/what-is-entityframework.aspx.

[10] *jQuery - Overview* [online]. Tutorials Point, 2019. [cit. 2019/05/08]. Dostupné z: https://www.tutorialspoint.com/jquery/jquery-overview.htm.

[11] *JavaScript - Overview* [online]. Tutorials Point, 2019. [cit. 2019/05/08]. Dostupné z: https://www.tutorialspoint.com/javascript/javascript_overview.htm.

[12] Angerová, Y. et al. Neurorehabilitace. *Česká a Slovenská neurologie a neurochirurgie*. 2010, 73, 106, s. 2. Dostupné z: http://www.csnn.eu/pdf?id=33805.

[13] Bydžovský, M. *Relační a nerelační modelování pro portál elektrofyziologických experimentů*. ZČU, 2014.

[14] Galloway, J. *Professional ASP.NET MVC 5*. John Wiley & Sons, Inc., 2014. ISBN 978-1-118-79475-3.

[15] Gaál, L. *Příručka k programu NP3, SAMCO*, 2015. Dostupné z: www.neurop.de.

[16] Physiopedia. *The effectiveness of gaming technology in neurological rehabilitation — Physiopedia,* [online]. 2019. [cit. 2019/05/01]. Dostupné z: https://www.physio-pedia.com/index.php?title=The_effectiveness_of_gaming_technology_in_neurological_rehabilitation&oldid=204961.

[17] Pop, D.-P. – Altar, A. Designing an MVC Model for Rapid Web Application Development. *Procedia Engineering*. 2014, 69, s. 1172 – 1179. ISSN 1877-7058. doi: https://doi.org/10.1016/j.proeng.2014.03.106. Dostupné z: http://www.sciencedirect.com/science/article/pii/S187770581400352X. 24th DAAAM International Symposium on Intelligent Manufacturing and Automation, 2013.

[18] Priya, J. *Top 5 Most Common Web Application Attacks That Affecting Websites* [online]. 2018. [cit. 2019/05/11]. Dostupné z: https://gbhackers.com/web-application-attacks/.

[19] *Securing Rails Application* [online]. 2019. [cit. 2019/05/13]. Dostupné z: https://guides.rubyonrails.org/security.html#cross-site-scripting-xss.

[20] Shannon, R. *What is HTML* [online]. 2012. [cit. 2019/05/07]. Dostupné z: https://www.yourhtmlsource.com/starthere/whatishtml.html.

[21] *Spring Security* [online]. 2019. [cit. 2019/05/12]. Dostupné z: https://spring.io/projects/spring-security.

[22] Van De Ven, R. M. et al. Brain training improves recovery after stroke but waiting list improves equally: A multicenter randomized controlled trial of a computer-based cognitive flexibility training. *PloS one*. 2017, 12, 3, s. e0172993.

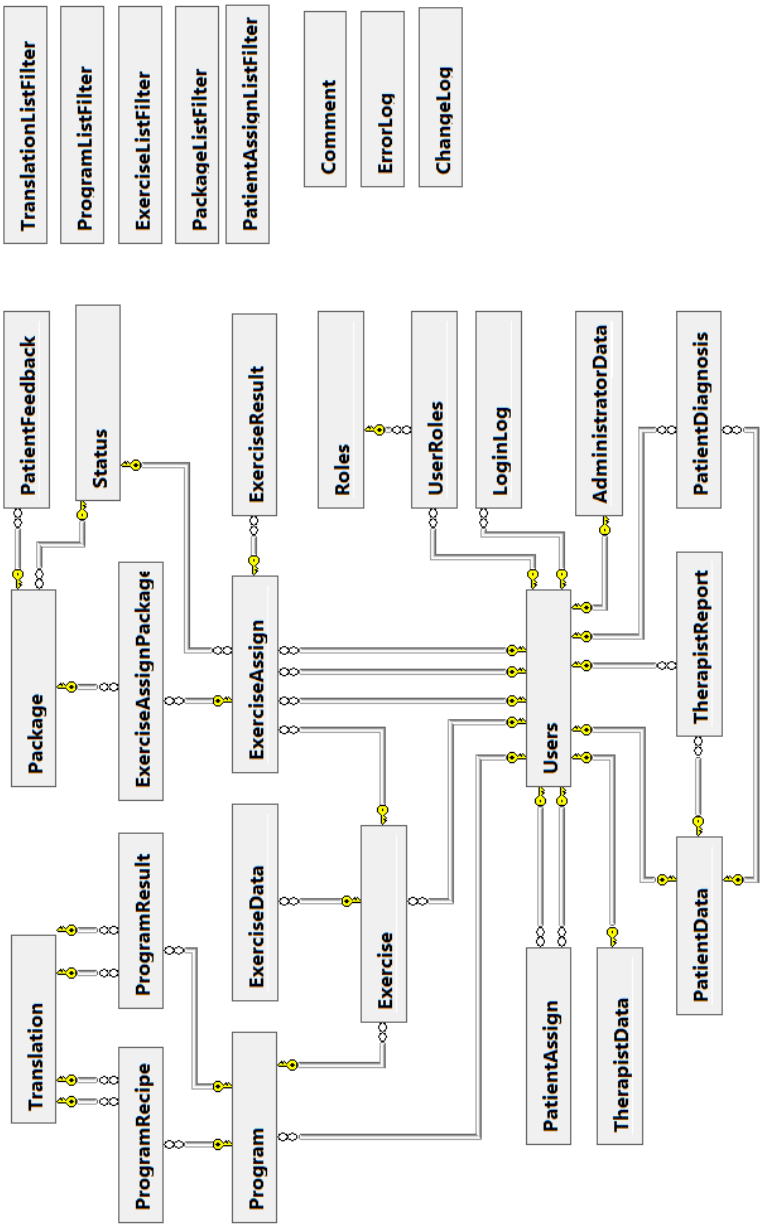
Seznam použitých zkratek a výrazů

Zkratka	Význam
ACID	Atomicity, Consistency, Isolation, Durability
AOP	Aspect Oriented Programming
ASP	Active Server Pages
CMP	Cévní mozková příhoda
CSFR	Cross-site request forgery
DDOS	Distributed Denial of Service
EF	Entity Framework
ER - model	Entity Relation - model
Framework	Software pro podporu programování
Front-End	část, kterou vidí návštěvník stránek
GIT	Distribuovaný systém správy verzí
HNP3	Home NeuroP 3
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
JavaEE	Java Platform, Enterprise Edition
JDBC	Java Database Connectivity
JMS	Java Messaging Services
JS	JavaScript
JSON	JavaScript Object Notation
LINQ	Language Integrated Query
MS	Microsoft
MSSMS	Microsoft SQL Server Management Studio
MVC	Model View Controller
NP3	Neurop 3
ORM	Object-Relational Mapping
OXM	Object XML Mapping
RDBMS	Relation DataBase Management Systém
SQL	Structured Query Language
THNP3	Therapist Home NeuroP 3
VS	Visual Studio
XML	eXtensible Markup Language

XSS	Cross-site scripting

Tabulka 8.1: Tabulka zkratek a výrazů

A Databázový model



Obrázek A.1: Kompletní pohled na databázové tabulky a jejich vazby.

B Případy užití

B.1 Registrace uživatele (UC01)

Popis

Každý nově příchozí uživatel má možnost se zaregistrovat do systému pomocí registračního formuláře. Jedná se o základní funkcionalitu, která umožní potenciálním zákazníkům využívat tuto aplikaci.

Standardní průběh

1. Uživatel klikne na tlačítko registrace v pravém horním rohu aplikace.
2. Vyplní všechny povinné údaje ve formuláři a klikne na tlačítko registrovat.
3. Nový uživatel obdrží email, ve kterém je žádán o ověření své emailové adresy.
4. Upozornění administrátora aplikace o novém uživateli v systému prostřednictvím emailu, aby došlo k přenastavení příslušné role.

Zúčastněné role

- Host

Vstupní podmínky

- žádné

Následná podmínka

- Uživatel získá účet v aplikaci

B.2 Přihlášení uživatele (UC02)

Popis

Každý zaregistrovaný uživatel má možnost přihlášení se do systému pomocí svých identifikačních údajů.

Standardní průběh

1. Uživatel klikne na tlačítko „přihlášení“ v pravém horním rohu aplikace.
2. Vyplní přihlašovací formulář a klikne na tlačítko „přihlásit se“.
    - Alternativa 2a – Byly zadány nesprávné přihlašovací údaje.
3. Uživateli se zobrazí jeho domovská obrazovka.

Zúčastněné role

- Administrátor, super-terapeut, terapeut, pacient, nepřiřazeno

Vstupní podmínky

- Uživatel musí mít vytvořený účet.

Následná podmínka

- Uživatel je přihlášen do systému.

Alternativní průběh

- 2a – Uživatel zadal nevalidní kombinaci uživatelského jména a hesla, a tak zůstane v přihlašovacím okně a bude o chybě informován. Pokud uživatel své přihlašovací údaje zapomněl, má možnost své heslo obnovit pomocí tlačítka „Zapomenuté heslo“ v dolní části přihlašovacího formuláře.

B.3 Přiřazení pacienta k terapeutovi (UC03)

Popis

Aby mohlo dojít k zahájení terapie, musí být nejdříve vytvořena vazba mezi terapeutem a pacientem. Vazba se bude vytvářet výběrem uživatelů ze seznamů a následným potvrzením akce. Po vytvoření vazby je terapeutovi umožněno přidělovat svým pacientům balíky s úlohami.

Standardní průběh

1. Uživatel klikne na kartu „Uživatelé“ v horním menu a následně na položku „Přiřazení pacientů“.
2. Následně se zobrazí seznam existujících přiřazení včetně možnosti přidání nového přiřazení kliknutím na tlačítko „Nové přiřazení pacienta“.
    - Alternativa 2a: Seznam existujících přiřazení může být prázdný.
3. Poté se zobrazí dialogové okno s dvěma rozbalovacími seznamy. V prvním seznamu se vybere terapeut a ve druhém pacient.
    - Alternativa 3a: Některý ze seznamu je prázdný.
4. Potvrzením této akce na tlačítko „Uložit“ dojde k vytvoření nové dvojice terapeut-pacient.

Zúčastněné role

- Administrátor

Vstupní podmínky

- Uživatel musí být přihlášený (viz UC02)

Následná podmínka

- Bude vytvořena vazba mezi terapeutem a pacientem.

Alternativní průběh

- 2a – Seznam může být prázdný z důvodu dosud neexistujícího přiřazení, nebo aktuálně nastaveného filtru.

- 3a – Seznam může být prázdný pouze v případě neexistence uživatele s danou rolí. V tomto případě případ užití končí.

B.4 Správa uživatelů v aplikaci (UC04)

Popis

Jedná se o funkcionalitu, která slouží administrátorům aplikace pro nastavování důležitých informací, hlavně uživatelských rolí, jednotlivým uživatelům.

Standardní průběh

1. Uživatel klikne na kartu „Uživatelé“ v horním menu a následně na položku „Správa uživatelů“.
2. Poté se zobrazí seznam všech uživatelů v aplikaci.
3. Každý záznam obsahuje tlačítko „náhled“ a „editace“.
4. Při editaci je uživateli umožněno nastavit konkrétnímu uživateli nové role.
    - Alternativa 4a – zobrazení náhledu

Zúčastněné role

- Administrátor

Vstupní podmínky

- Uživatel musí být přihlášený (viz UC02).

Následná podmínka

- Role konkrétního uživatele budou změněny.

Alternativní průběh

- 4a – Při kliknutí na tlačítko „Náhled“ dojde pouze k zobrazní informací o uživateli bez možnosti jeho editace.

B.5 Správa profilu uživatele (UC05)

Popis

Jedná se o scénář, ve kterém je uživateli umožněno nastavit svůj vlastní profil. Jedná se o osobní informace, kontaktní a registrační údaje.

Standardní průběh

1. Uživatel klikne na kartu, jejíž název se shoduje se jménem aktuálně přihlášeného uživatele v levém horním roku a záložku „Nastavení účtu“.
2. Uživateli se zobrazí okno se třemi záložkami, kde podle potřeby upraví svoje nastavení.
3. Po provedení změn klikne na tlačítko „Uložit vše“.

Zúčastněné role

- Administrátor, super-terapeut, terapeut a pacient

Vstupní podmínky

- Uživatel musí být přihlášený (viz UC02).

Následná podmínka

- Změna osobních údajů uživatele.

B.6 Správa pacientů (UC06)

Popis

Akce určená pro terapeuty, která slouží pro detailní náhled svých pacientů. V tomto náhledu jsou uvedeny základní informace o pacientovi a možnost zadávání krátkých zpráv o pokroku pacienta v rehabilitaci. Dále jsou v tomto editoru seznamy všech úloh, které byly pacientovi zadány. Tyto úlohy lze následně analyzovat.

Standardní průběh

1. Terapeut klikne na kartu „Pacienti“ a záložku „Správa pacientů“.
2. Poté se zobrazí seznam všech aktivních pacientů daného terapeuta.
    - Alternativa 2a - Seznam je prázdný
3. Každý záznam v seznamu obsahuje tlačítko „náhled“ pro přesměrování do profilu daného pacienta.
4. Následně se terapeutovi zobrazí náhled pacienta.

Zúčastněné role

- Super-terapeut, terapeut

Vstupní podmínky

- Uživatel musí být přihlášený (viz UC02)

Alternativní průběh

- 2a – Seznam je prázdný, pokud terapeut nemá přiřazeného žádného pacienta.

B.7 Zobrazení seznamu překladů (UC07)

Popis

Překlady jsou v aplikaci vytvořeny pro to, aby bylo možné jazykově lokalizovat nově vytvořené programy. Jedná se o lokalizaci vstupních a výstupních parametrů programů, aby terapeut věděl, které parametry co znamenají. Tento případ užití je dostupný všem rolím, které mají možnost vytvářet programy.

Standardní průběh

1. Uživatel klikne na kartu „Překlady“ a záložku „Správa překladů“
2. Poté se zobrazí seznam všech překladů
    - Alternativa 2a – Seznam je prázdný.

Zúčastněné role

- Administrátor, Super-terapeut

Vstupní podmínky

- Uživatel musí být přihlášený (viz UC02).

Alternativní průběh

- 2a – Seznam může být prázdný z důvodu špatně nastaveného filtru.

B.8 Vytvoření nového překladu (UC08)

Popis

Pokud uživatel při vytváření nového programu zjistí, že potřebuje nový překlad pro nadefinování vstupu či výstupu z programu, pak si jej vytvoří a následně použije.

Standardní průběh

1. Uživatel klikne na kartu „Překlady“ a následně na záložku „Vytvořit překlad“.
    - Alternativa 1a – Vytvoření je také možné z editace programu.
2. Uživateli se zobrazí editor s formulářem pro vytvoření nového překladu.
3. Po potvrzení formuláře je uživatel přesměrován do seznamu překladů.

Zúčastněné role

- Administrátor, super-terapeut

Vstupní podmínky

- Uživatel musí být přihlášený (viz UC02).

Následná podmínka

- Dojde k vytvoření nového překladu.

Alternativní průběh

- 1a – Vytváření nového překladu je umožněno i z editace programu, aby ji uživatel nemusel opouštět.

B.9 Editace překladu (UC09)

Popis

Při vytváření překladu není povinné ihned uvádět všechny jazykové mutace. Proto je později vhodné tyto údaje doplnit.

Standardní průběh

1. Uživatel si nechá zobrazit seznam všech překladů (viz UC07).
2. V seznamu najde konkrétní záznam a klikne na tlačítko „Editace“
3. V editoru provede uživatel potřebné změny a klikne na tlačítko „Uložit“.
4. Uživatel bude následně přesměrován do seznamu překladů.

Zúčastněné role

- Administrátor, super-terapeut

Vstupní podmínky

- Uživatel musí být přihlášený (viz UC02).

Následná podmínka

- Požadovaných překlad bude upraven.

B.10 Duplikace překladu (UC10)

Popis

V některých případech může být užitečné vytvořit nový překlad na základě již existujícího, proto je zde možnost duplikace překladu, který se následně v jeho editaci upraví.

Standardní průběh

1. Uživatel si nechá zobrazit seznam všech překladů (viz UC07).
2. Po nalezení konkrétního překladu uživatel klikne na tlačítko „Duplikovat“
    - Alternativa 2a – Tlačítko zde není dostupné.
3. Uživatel je přesměrován do editace překladu, kde má možnost upravit nový (zduplikovaný) překlad (viz UC09).

Zúčastněné role

- Administrátor, super-terapeut.

Vstupní podmínky

- Uživatel musí být přihlášený (viz UC02).

Následná podmínka

- Dojde k vytvoření nového překladu.

Alternativní průběh

- 2a – Tlačítko je dostupné pouze u překladů, které již jsou v aplikaci použity. Případ užití zde tedy končí.

B.11 Smazání překladu (UC11)

Popis

Aby nedocházelo k hromadění nevyužitých překladů v aplikaci, pak je zde možnost takové překlady smazat.

Standardní průběh

1. Uživatel si nechá zobrazit seznam překladů (viz UC07).
2. V tomto seznamu najde požadovaný překlad a klikne na tlačítko „Smazat“.
    - Alternativa 2a – Tlačítko smazat u konkrétního překladu není.
3. Po smazání záznamu dojde k obnovení seznamu překladů.

Zúčastněné role

- Administrátor, super-terapeut

Vstupní podmínky

- Uživatel musí být přihlášený (viz UC02).

Následná podmínka

- Požadovaný záznam bude odebrán.

Alternativní průběh

- 2a – Smazat lze pouze takový překlad, který nebyl dosud nikde v aplikaci použit.

B.12 Zobrazení seznamu programů (UC12)

Popis

Uživatelé s příslušnými rolemi mají možnost si zobrazit seznam programů, které sami vytvořili nebo jsou nastaveny jako veřejné.

Standardní průběh

1. Uživatel klikne v menu na kartu „Programy“, čímž se mu rozbalí seznam, a následně klikne na položku „Správa programů“.
2. Následně dojde k zobrazení seznamu uživatelovi dostupných programů.
    - Alternativa 2a: Seznam je prázdný.

Zúčastněné role

- Administrátor, super-terapeut, terapeut

Vstupní podmínky

- Uživatel musí být přihlášený (UC02).

Následná podmínka

- Uživateli se zobrazí seznam programů, ke kterým má přístup.

Alternativní průběh

- 2a – Seznam může být prázdný ze dvou důvodů. Prvním důvodem může být, že danému filtru neodpovídá žádná úloha. Druhým, že uživatel dosud nevytvořil žádnou úlohu, nebo neexistuje úloha, která by byla veřejná. V tomto případě dojde k ukončení případu užití.

B.13 Vytvoření nového programu (UC13)

Popis

Oprávnění uživatelé mají možnost vytvářet nové programy. Tímto se rozumí pouze jeho založení, tedy vyplnění povinných atributů programu. Následně dojde k přesměrování do jeho editace.

Standardní průběh

1. Uživatel klikne na kartu „Programy“ a následně na záložku „Vytvořit nový program“.
2. Poté dojde k přesměrování do formuláře, kde vyplní povinné údaje.
3. Potvrzením formuláře dojde k založení úlohy a přesměrování do editoru, kde se program dodefinuje (viz UC14).

Zúčastněné role

- Administrátor, super-terapeut

Vstupní podmínky

- Uživatel musí být přihlášený (viz UC02).

Následná podmínka

- Dojde k založení nového programu.

B.14 Editace programu (UC14)

Popis

Jedná se o uživatelskou akci, při které dochází k nadefinování prototypu programu, ze kterého budou následně vytvářeny úlohy pro pacienty. Základem pro vytvoření prototypu je nadefinování jeho vstupních parametrů, pomocí kterých dojde ke spuštění úlohy, a výstupních parametrů, které se budou z daného programu ukládat a budou sloužit pro analýzu výsledků. Program bude mít dále několik metadat, například jeho název a krátký popis.

Standardní průběh

1. Uživatel si zobrazí seznam dostupných programů (viz UC12).
2. Z tohoto seznamu si uživatel vybere program, který chce editovat, a klikne na tlačítko „Editace“.
    - Alternativa 2a – U požadovaného programu není tlačítko „Editace“
3. Následně je uživatel přesměrován do editoru programu.
4. Uživatel provede potřebné změny a kliknutím na tlačítko „Uložit“ aktualizuje daný program.

Zúčastněné role

- Administrátor, super-terapeut

Vstupní podmínky

- Uživatel musí být přihlášený, viz UC02.
- Uživatel musí mít přístupný některý z programů, který je možné editovat.

Následná podmínka

- Aktualizace programu.

Alternativní průběh

- 2a – Program je editovatelný pouze tehdy, pokud je uživatel jeho autorem a zároveň není dosud použitý. V ostatních případech je dostupné pouze tlačítko „Náhled“.

B.15 Náhled programu (UC15)

Popis

Tento scénář je téměř totožný od jeho editace (viz UC14) s tím rozdílem, že zde jsou téměř všechny informace pouze pro čtení. Nastavit se dá příznak aktivity, jeho dostupnost a také se dá nahrát nový skript, který úlohu spouští. Oproti editace je náhled umožněn i obyčejným terapeutům.

Standardní průběh

1. Uživatel si nechá zobrazit seznam programů (viz UC12).
2. V tomto seznamu si uživatel vybere konkrétní program a klikne na tlačítko „Náhled“
    - Alternativa 2a – Tlačítko náhled u požadovaného záznamu není.
3. Uživatel je přesměrován do náhledu programu.

Zúčastněné role

- Administrátor, super-terapeut, terapeut

Vstupní podmínky

- Uživatel musí být přihlášený (viz UC02).

Následná podmínka

- U programu může být nastavena jeho aktivita, dostupnost a nový spouštěcí skript.

Alternativní průběh

- 2a – Tlačítko náhled je dostupný tehdy, až když je používaný. Program je tedy buď editovatelný, nebo určený pouze pro náhled.

B.16 Duplikace programu (UC16)

Popis

Operace, která vytvoří kopii existujícího programu, respektive jeho funkčního předpisu. Skript pro spuštění programu se v tomto případě neduplikuje, protože se předpokládá, že program budeme chtít aktualizovat.

Standardní průběh

1. Uživatel si nechá zobrazit seznam všech programů (viz UC12).
2. V tomto seznamu si uživatel vybere konkrétní program pro duplikaci a klikne na tlačítko „Duplikovat“.
    - Tlačítko u požadovaného záznamu chybí.
3. Následně dojde k duplikaci programu a přesměrování do jeho editace (viz UC14).

Zúčastněné role

- Administrátor, super-terapeut

Vstupní podmínky

- Uživatel musí být přihlášený (viz UC02).

Následná podmínka

- Dojde k vytvoření kopie existujícího programu.

Alternativní průběh

- 2a – Duplikovat lze pouze takový program, který je již používaný.

B.17 Smazání programu(UC17)

Popis

V případě nepoužívaného programu je možné jej ze systému smazat.

Standardní průběh

1. Uživatel si nechá zobrazit seznam všech programů (viz UC12).
2. V tomto seznamu najde požadovaný program a klikne na tlačítko „Smazat“.
    - Alternativa 2a – Tlačítko u konkrétního záznamu není.
3. Po odebrání programu je uživatel přesměrován do seznamu programů.

Zúčastněné role

- Administrátor, super-terapeut

Vstupní podmínky

- Uživatel musí být přihlášený (viz UC02).

Následná podmínka

- Program bude odebrán ze systému.

Alternativní průběh

- 2a – Smazání programu je možné pouze v případě, dokud není v aplikaci použit, resp. z daného programu nebyla dosud vytvořena úloha. Další požadavkem je, že daný uživatel musí být autorem programu.

B.18 Zobrazení seznamu úloh (UC18)

Popis

Posloupnost akcí vedoucí k náhledu na seznam úloh, ke kterým má daný uživatel přístup. Jedná se o úlohy, které sám vytvořil nebo jsou veřejné.

Standardní průběh

1. Uživatel klikne do hlavního menu na kartu „Úlohy“ a následně na položku „Správa úloh“.
2. Uživateli se zobrazí seznam dostupných úloh.
    - Alternativa 2a – Seznam je prázdný.

Zúčastněné role

- Administrátor, super-terapeut, terapeut

Vstupní podmínky

- Uživatel musí být přihlášený (viz UC02)

Následná podmínka

- Uživateli se zobrazí seznam úloh, ke kterým má přístup a odpovídají zadanému filtru.

Alternativní průběh

- 2a – Seznam může být prázdný z několika důvodů. Buď uživatel žádnou úlohu nevytvořil, nebo neexistuje úloha, která by byla veřejná. Další možností je, že žádná z úloh neodpovídá zadanému filtru.

B.19 Vytvoření nové úlohy (UC19)

Popis

Jedná se o scénář, kdy dojde pouze k založení nové úlohy zadáním jejích povinných parametrů.

Standardní průběh

1. Uživatel klikne na kartu „Úlohy“ a následně na záložku „Vytvořit úlohu“.
2. Poté dojde k přesměrování do formuláře, kde dojde k vyplnění povinných parametrů a jeho potvrzení kliknutím na tlačítko „Vytvořit“.
3. Po vytvoření úlohy dojde k přesměrování do její editace (viz UC20).

Zúčastněné role

- Administrátor, super-terapeut, terapeut.

Vstupní podmínky

- Uživatel musí být přihlášený (viz UC02).

Následná podmínka

- Bude vytvořena nová úloha.

B.20 Editace úlohy (UC20)

Popis

Operace, při které dojde k definování konkrétní podoby úlohy, tedy převážně k nadefinování vstupních parametrů úlohy. Při vytváření úlohy (UC19) dojde pouze k jejímu založení - vyplnění jejich povinných atributů. V editaci úlohy je možnost nastavit příznak její veřejnosti, tj. zda ji mohou používat i ostatní uživatelé aplikace, nebo pouze její vlastník. Dalším atributem je příznak její aktivity, tj. zda může být úloha vůbec použita. Neaktivní úlohu vidí pouze její autor a může se rozhodnou, zda ji zaktivuje, nebo ji z aplikace odstraní.

Standardní průběh

1. Uživatel si zobrazí jemu dostupné úlohy (viz UC18).
2. V tomto seznamu si vybere konkrétní úlohu a klikne na tlačítko „Editovat“, čímž se mu zobrazí editor úlohy.
    - Alternativa 2a – Položka neobsahuje tlačítko „Editovat“.
3. V tomto editoru provede potřebné změny a klikne na tlačítko „Uložit“.

Zúčastněné role

- Administrátor, super-terapeut, terapeut

Vstupní podmínky

- Terapeut musí být přihlášen (viz UC02).
- Musí existovat úloha, ke které má uživatel přístup (viz UC18).

Následná podmínka

- Úloha bude aktualizována.

Alternativní průběh

- 2a – Úloha je editovatelná pouze v případě, když ji chce upravit její vlastník a zároveň úloha nesmí být dosud přiřazena pacientovi.

B.21 Duplikace úlohy (UC21)

Popis

Jedná se o operaci, při které se bude vytvářet kopie existující úlohy. Funkcionalita bude určena k vytváření úloh, které se od sebe výrazněji neliší, a tak bude ulehčena její tvorba. Tímto se vytvoří nová úloha s totožným zadáním.

Standardní průběh

1. Uživatel si nechá zobrazit seznam úloh (viz UC18).
2. V seznamu nalezne požadovanou úlohu a klikne na tlačítko „Duplikovat“.
    - Alternativa 2a – Tlačítko duplikovat u požadovaného záznamu není.
3. Po vytvoření kopie je uživatel přesměrován do editace nové (zduplikované) úlohy (viz UC20).

Zúčastněné role

- Administrátor, super-terapeut, terapeut

Vstupní podmínky

- Uživatel musí být přihlášený (viz UC02).

Následná podmínka

- Dojde k vytvoření nové úlohy.

Alternativní průběh

- 2a – Zduplikovat lze pouze takovou úlohu, která je již používána.

B.22 Smazání úlohy (UC22)

Popis

V případě potřeby odebrání úlohy ze systému použije uživatel tento scénář. Odebírat lze pouze úlohy, které nebyly dosud použity.

Standardní průběh

1. Uživatel si zobrazí seznam úloh v systému (viz UC18).
2. U požadovaného záznamu klikne na tlačítko „Smazat“.
    - Alternativa 2a – Tlačítko u požadovaného záznam není.
3. Po odebrání úlohy je uživatel přesměrován do seznamu úloh.

Zúčastněné role

- Administrátor, super-terapeut, terapeut

Vstupní podmínky

- Uživatel musí být přihlášený (viz UC02)

Následná podmínka

- Dojde k odebrání požadované úlohy.

Alternativní průběh

- 2a – Smazat lze pouze úlohu, která nebyla dosud spuštěna některým z pacientů. Úlohy může mazat pouze jejich autor.

B.23 Náhled úlohy (UC23)

Popis

V případě nedostupné editace úlohy (viz UC20) má uživatel možnost jejího náhledu, kde již není umožněna změna jejího zadání. Uživatel má zde pouze možnost, pokud je jejím autorem, měnit příznak její dostupnosti ostatním uživatelům, tj. zda úlohu mohou zadávat i ostatní terapeuti. Soukromou úlohu může zadávat pouze její autor.

Standardní průběh

1. Uživatel si nechá zobrazit seznam vytvořených úloh (viz UC18)
2. U požadovaného záznamu v tabulce klikne na tlačítko „Náhled“.
    - Alternativa 2a – Tlačítko u požadovaného záznamu není.
3. Uživatel je přesměrován do editoru náhledu.

Zúčastněné role

- Administrátor, super-terapeut, terapeut

Vstupní podmínky

- Uživatel musí být přihlášený (viz UC02).

Alternativní průběh

- 2a – Pokud je úloha editovatelná, potom je toto tlačítko nedostupné. Dalším případem může být, že úloha je neaktivní, cizí a soukromá. V tomto případě není dostupné žádné tlačítko.

B.24 Otestování vytvořené úlohy (UC24)

Popis

Předtím, než se terapeut rozhodne některou ze svých úloh použít, je dobré tuto úlohu otestovat, tedy zjistit, zda funguje podle očekávání.

Standardní průběh

1. Uživatel se dostane do editace nebo náhledu dané úlohy (viz UC20 a UC23).
2. Poté uživatel klikne na tlačítko „Otestovat úlohu“ v pravém horním rohu.
3. Následně dojde ke spuštění dané úlohy

Zúčastněné role

- Administrátor, super-terapeut, terapeut

Vstupní podmínky

- Uživatel musí být přihlášený (viz UC02).
- Musí existovat úloha, kterou chceme testovat.

Následná podmínka

- Dojde k otestování dané úlohy.

B.25 Zobrazení seznamu balíků (UC25)

Popis

Případ užití, ve kterém se uživateli zobrazí seznam balíků, které vytvořil pro své pacienty.

Standardní průběh

1. Uživatel klikne do hlavního menu na kartu „Balíky“ a následně na položku „Správa balíků“.
2. Uživateli se zobrazí seznam jeho balíků
    - Alternativa 2a – Seznamů balíků je prázdný.

Zúčastněné role

- Super-terapeut, terapeut

Vstupní podmínky

- Uživatel musí být přihlášený (viz UC02).

Následné podmínky

- Uživateli se zobrazí seznam balíků, které vytvořil pro své pacienty.

Alternativní průběh

- 2a – Seznamů balíků může být prázdný buď z důvodu, že terapeut dosud žádný nevytvořil, nebo že žádný z balíku neodpovídá zadanému filtru.

B.26 Vytvoření balíku s úlohami (UC26)

Popis

Úlohy jako takové nelze zadávat samostatně, a tak je potřeba úlohy organizovat do balíků. Vytvoření takového balíku se provádí následujícím způsobem.

Standardní průběh

1. Uživatel klikne v horním menu na kartu „Balíky“ a následně na položku „Vytvořit balík“.
    - Alternativa 1a – Možnost založení nového balíku ze seznamu balíků (viz UC25).
2. Následně se uživateli zobrazí okno, kde musí vyplnit název balíku a ze seznamu vybrat pacienta, kterému tento balík bude patřit.
    - Alternativa 2a – Seznam pacientů je prázdný.
3. Poté je terapeut přesměrování do editoru balíku, který je popsán v UC27.

Zúčastněné role

- Super-terapeut, terapeut

Vstupní podmínky

- Uživatel musí být přihlášený (viz UC02).
- Uživatel musí mít nějakého pacienta.

Následná podmínka

- Vytvoření balíku, který je přidružen k pacientovi.

Alternativní průběh

- 1a – Možnost vytvoření balíku je také v pohledu seznamu již vytvořených balíků (viz UC25) v pravém horním rohu pod odkazem „Vytvořit balík“.

- 2a – Pokud terapeut nemá žádného pacienta, pak nemůže vytvářet balík, protože se jedná o povinný údaj. V tomto případě průběh vytváření balíku končí.

B.27 Editace balíku s úlohami (UC27)

Popis

Pomocí této akce je umožněna terapeutovi manipulace s balíkem. Terapeutovi je zde umožněno vkládat nové úlohy do balíku či je z balíku odebírat. Součástí editoru by měl být i seznam se zpětnou vazbou od pacienta na zadaný balík, kterou pacient má možnost vytvořit po procvičení daného balíku.

Standardní průběh

1. Uživatel si zobrazí seznam vytvořených balíků (viz UC25)
2. U hledaného balíku klikne uživatel na tlačítko „Editovat“.
    - Alternativa 2a – Do editace balíku se lze dostat i přes vytvoření nového balíku (viz UC26)
3. Uživatel v balíku provede potřebné změny a následně je uloží kliknutím na tlačítko „Uložit“.
4. Uživateli se zobrazí upravený balík se všemi změnami.

Zúčastněné role

- Super-terapeut, terapeut

Vstupní podmínky

- Uživatel musí být přihlášený (viz UC02).
- Uživatel musí mít již vytvořený nějaký balík.

Následná podmínka

- Aktualizace existujícího balíku.

Alternativní průběh

- 2a – Jakmile dojde k vytvoření nového balíku (UC26), pak je uživatel automaticky přesměrován do jeho editace.

B.28 Duplikace balíku (UC28)

Popis

Balík v každém případě patří pouze jednomu pacientovi. Pokud se terapeut rozhodne, že by chtěl konkrétní balík včetně všech jeho úloh přiřadit někomu jinému, pak použije tento případ užití.

Standardní průběh

1. Uživatel se dostane do editace balíku (viz UC 27).
2. V pravém horním rohu se nalézá tlačítko „Duplikovat“, na které uživatel klikne.
3. Následně je uživatel vyzván pro vložení nového názvu balíku a výběru pacienta, kterému bude balík určený.
4. Potvrzení akce kliknutím na tlačítko „Duplikovat“ dojde k vytvoření nového balíku a přesměrování do jeho editace

Zúčastněné role

- Super-terapeut, terapeut

Vstupní podmínky

- Uživatel musí být přihlášený (viz UC02).
- Uživatel musí mít pacienta, který daný balík dosud nemá přiřazený.

Následná podmínka

- Dojde k vytvoření nového balíku pro konkrétního pacienta.

B.29 Smazání balíku (UC29)

Popis

Jedná se o posloupnost akcí, která umožní terapeutovi smazat již nepotřebný balík. Smazáním balíku nedojde k odebrání získaných výsledků z jednotlivých cvičení, pouze dojde ke ztrátě seskupení úloh, které byly v balíku obsaženy. Pokud bude úloha přidána do jiného balíku, pak tento balík bude obsahovat i data z předchozích spuštění úlohy.

Standardní průběh

1. Uživatel přejde do editace požadovaného balíku (viz UC27)
    - Alternativa 1a – Smazat balík lze také ze seznamu balíků
2. Poté klikne na tlačítko „Smazat“, které se nachází v pravém horním rohu okna.
3. Následně dojde k odebrání balíku a přesměrování na seznam dostupných balíků.

Zúčastněné role

- Super-terapeut, terapeut

Vstupní podmínky

- Uživatel musí být přihlášený (viz UC02).

Následná podmínka

- Dojde k odebrání požadovaného balíku.

Alternativní průběh

- 1a – V seznam všech balíků (viz UC25) je u každého záznamu tlačítko „Smazat“, které také provede smazání balíku.

B.30 Analýza výsledků (UC30)

Popis

Velice důležitá funkcionalita určena pro vyhodnocení pokroku pacienta. Terapeut má zde možnost náhledu na tabulkový výpis výstupních parametrů z jednotlivých spuštění dané úlohy.

Standardní průběh

1. Přihlášený uživatel klikne v hlavním menu na kartu „Analýzy“ a následně na položku „Analýza“.

2. Následně je uživatel přesměrován do editoru, ve kterém má terapeut možnost filtrace úloh podle typu programu, názvu úlohy a jména pacienta. Tyto filtry se také dají kombinovat. Touto filtrací si dohledáme konkrétní úlohu a kliknutím na tlačítko „Analyzovat“ dojde k přesměrování na analýzu.

    - Alternativa 2a – Analýza je také dostupná přes editor balíku.

3. V tomto okně se nalézá seznam výstupních hodnot z jednotlivých provedení konkrétní úlohy, ze kterých terapeut provádí analýzu.

Zúčastněné role

- Super-terapeut, terapeut

Vstupní podmínky

- Uživatel musí být přihlášený do systému (viz UC02).

Následná podmínka

- Uživatel provedl analýzu výsledků pacienta

Alternativní průběh

- 2a – Jednotlivé úlohy jsou strukturovány do balíků, které jsou zadávány pacientům. Proto se v jeho editoru (viz UC27) nalézá seznam úloh ze který se skládá. Každá úloha v seznam obsahuje tlačítko „Analýza“, která umožní náhled na její výsledky. Případ užití pokračuje bodem 3.

C Uživatelská dokumentace

Uživatelská příručka obsahuje kompletní informace o možnostech, které v této webové aplikaci uživatelé mají. Jednotlivé možnosti budou popisovány z pohledu uživatelských rolí. Pokud se budou možnosti v rámci rolí překrývat, pak na již uvedenou možnost pouze odkáži. V podkapitole „Obecná funkcionalita“ jsou operace, které mají umožněny všechny role. Následně dojde k popisu konkrétních rolí.

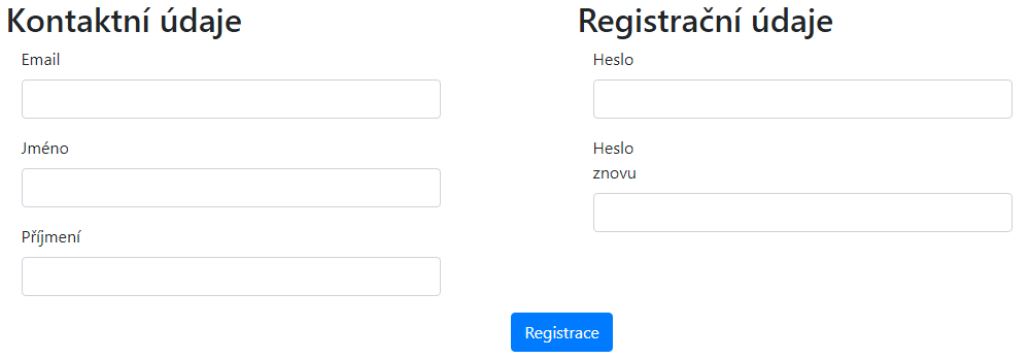
Součásti papírové verze této diplomové práce je pouze její zkrácená verze, která obsahuje popis funkcionalit, které jsou přístupné terapeutovi. Kompletní verze uživatelské příručky je pouze v elektronické podobě, která se nalézá na odevzdávaném CD a v repositáři spolu s projektem.

C.1 Obecná funkcionalita

V této sekci budou popisovány operace, které mají dostupné všechny role v aplikaci.

C.1.1 Registrace

Do registračního formuláře se uživatel dostane kliknutím na tlačítko „registrace“ v pravém horním rohu aplikace. Při registraci (viz obr. C.1) je uživatel povinen vyplnit všechny povinné atributy, které jsou v průběhu vyplňování automaticky validovány. Proto lze provést registraci až po jejich korektním vyplnění. Po dokončení registrace přijde uživateli e-mail s instrukcemi, jak potvrdit svoji e-mailovou adresu.



Obrázek C.1: Registrace do systému

C.1.2 Přihlášení

Formulář pro přihlášení se nalézá pod odkazem v pravém horním rohu aplikace na uvítací obrazovce. Pro přihlášení uživatele je nutno vyplnit přihlašovací formulář. Po správném zadání e-mailu, hesla a kliknutím na tlačítko přihlásit dojde k přihlášení uživatele.

C.1.3 Obnovení hesla

Obnovení zapomenutého hesla se provádí v rámci přihlašovacího formuláře (viz Přihlášení) kliknutím na tlačítko „Zapomenuté heslo“. Uživatel je následně přesměrován do formuláře, kam vloží svoji e-mailovou adresu, na kterou mu přijde zpráva s instrukcemi.

C.1.4 Připomínkování

Jedná se o funkcionalitu, pomocí které je umožněno přihlášeným uživatelům nahlašovat nalezené chyby nebo návrhy na vylepšení aplikace. Tato možnost je dostupná přes hlavní menu pod kartou „Připomínky“, nebo pod kartou v pravém horním rohu, která obsahuje text se jménem aktuálně přihlášeného uživatele. Zde se nalézá položka „Vytvořit připomínku“ a „seznam připomínek“. Vytvoření připomínky dojde vyplněním příslušných polí ve formuláři.

C.1.5 Nastavení účtu a jazyka

Tyto možnosti se nalézají pod kartou v pravém horním rohu aplikace kliknutím na jméno aktuálně přihlášeného uživatele. Poté se zobrazí odkazy pro nastavení účtu a jazyka.

V nastavení svého účtu je uživateli umožněno změnit svoje kontaktní a registrační údaje včetně změny stávajícího hesla. Kliknutím na záložku „Nastavení jazyka“ se uživateli zobrazí seznam dostupných lokalizací, kdy při kliknutí na libovolný z nich dojde ke změně lokalizace.

C.2 Administrátor

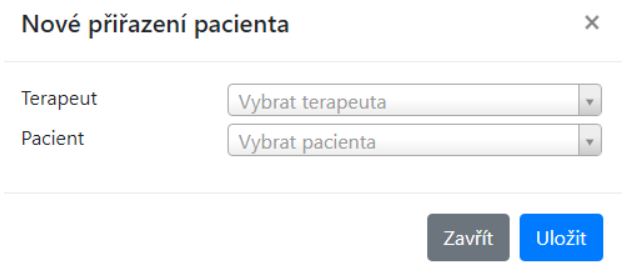
V této části bude popsána funkcionalita, která je určená pro administrátory aplikace.

C.2.1 Správa uživatelů

Jednou z povinností administrátora je správa všech uživatelů v aplikaci. V této správě jde primárně o **nastavení uživatelských rolí** jednotlivým uživatelům. Tato akce je dostupná pod kartou „Uživatelé“ a následně pod záložkou „Správa uživatelů“. Zde se administrátorovi zobrazí seznam všech uživatelů v aplikaci. Tento seznam může být vyfiltrován pomocí textového filtru „Hledat uživatele“. Po nalezení konkrétního uživatele a kliknutí na tlačítko „Editovat“ je administrátor přesměrován do správy uživatele, kde může danému uživateli nastavení jeho role.

C.2.2 Přiřazení pacienta k terapeutovi

Aby mohlo dojít k zahájení rehabilitace, musí být nejprve vytvořena vazba mezi terapeutem a pacientem. Tato akce je dostupná kliknutím na kartu „Uživatelé“ v hlavním menu a následně kliknutím na záložku „Přiřazení pacienta“. Následně dojde k přesměrování do okna s tabulkou již existujících přiřazení, nad kterou se dá filtrovat pomocí přiloženého filtru. Součástí okna je tlačítko „Nové přiřazení pacienta“, které umožňuje vytvořit novou vazbu prostřednictvím dvou seznamů (viz obr. C.2).



Obrázek C.2: Modální okno pro vytvoření vazby mezi terapeutem a pacientem

C.2.3 Správa překladů

Překlady jsou v aplikaci použity z důvodu ulehčení práce při vytváření nových úloh. Vstupní parametry těchto úloh budou v jazykových lokalizacích, které si uživatel zvolí, a tak bude přesně vědět, co který parametr znamená.

Vytvoření překladu

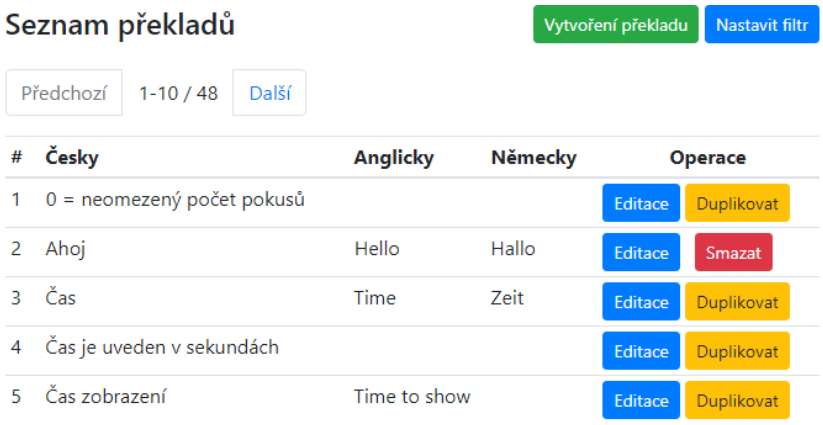
Akce pro vytvoření nového překladu je dostupná z hlavního menu pod kartou „Překlady“ a následně záložkou „Vytvořit překlad“. Poté je uživatel přesmě-

rován na formulář, který vyplní a potvrdí. Následně bude přesměrován na seznam všech překladů.

Zobrazení seznamu překladů

Akce, pomocí které jsou umožněny základní operace na překlady typu **editace**, **duplikace** a **mazání**. Seznam si uživatel zobrazí kliknutím na kartu „Překlady“ a následně na záložku „Správa překladů“. Poté se uživateli zobrazí seznam všech překladů (viz obr. C.3), které jsou aktuálně v aplikaci vytvořeny. Tento seznam se dá filtrovat na základě filtru, který je pro každého uživatele unikátní. Tento filtr je umístěný v pravém horním rohu obrazovky. V tomto editoru je také možnost vytvoření nového překladu pomocí modálního okna.

Při použití tlačítka **duplikace** dojde k vytvoření nového překladu, který bude obsahovat náhodnou příponu. Při kliknutí na tlačítko **smazat** dojde k odebrání překladu a obnovení seznamu s překlady.



Obrázek C.3: Náhled na okno se seznamem překladů

Editace překladu

Akce dostupná pomocí zobrazení seznamu překladů (viz podkapitola Zobrazení seznamu překladů) kliknutím na tlačítko „Editace“. Poté je uživatel přesměrování do editoru, ve kterém může upravovat požadovaný překlad. V případě, že je překlad již používán, je o tomto faktu uživatel informován, aby nedošlo ke změně jeho významu.

Součástí editoru je odkaz pro zobrazení seznamu překladů. Po uložení formuláře je uživatel přesměrován do seznamu překladů.

C.2.4 Správa programů

Program je struktura, která definuje vstupní a výstupní parametry do skriptu, který spustí danou úlohu. Na základě vytvořeného programu lze tedy zakládat nové úlohy.

Vytvoření nového programu

Nový program lze vytvořit kliknutím do hlavního menu na kartu „Programy“ a dále na záložku „Vytvoření programu“. Následně je uživatel přesměrován do formuláře, kam zadá *název* a *popis programu*. Pokud budou vstupy validní, dojde k založení nového programu a přesměrování do jeho **editace**.

Zobrazení seznamu programů

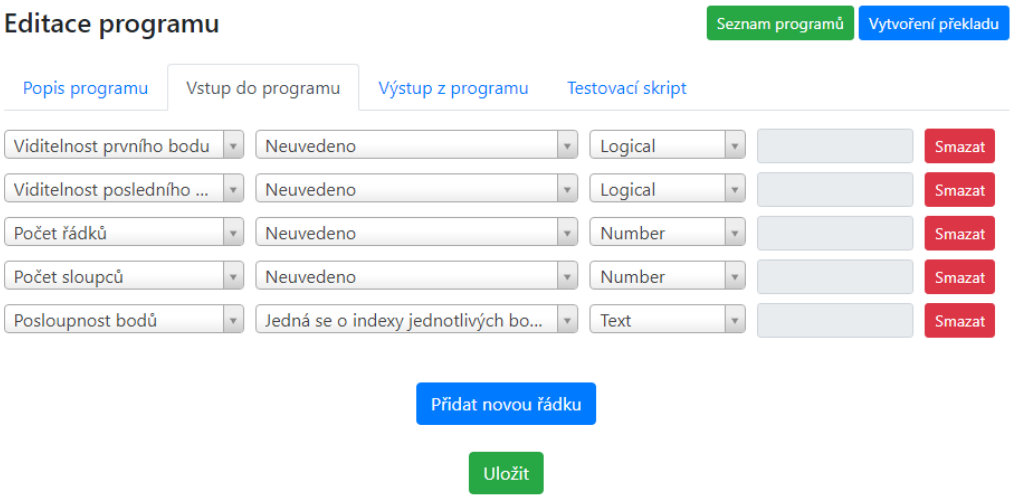
Akce vedoucí k tabulkovému zobrazení všech programu, ke kterým má uživatel přístup. Uživatel má pochopitelně dostupné všechny svoje úlohy a zároveň takové úlohy, které jsou aktivní a nejsou soukromé. Nad tímto výpisem je možné použít filtr, který lze nastavit v pravém horním rohu okna. Nastavení tohoto filtru se ukládá do databáze, a tak při další návštěvě se použije jeho poslední konfigurace. V tomto okně je možné pomocí tlačítka vedle filtru vytvořit novou úlohu.

Každý ze zobrazených programů můžeme pomocí příslušných tlačítek **editovat**, **nahlížet**, **duplikovat** nebo **mazat**. Tlačítka se zobrazuje podle práva uživatele k danému programu. **Duplikace** vytvoří kopii programu včetně jeho typů vstupních a výstupních parametrů. Takový program je označen jako aktivní a privátní. Jeho název obdrží koncovku *_duplicate* a skript pro spuštění úlohy se nekopíruje. Tlačítkem **smazat** dojde k odebrání daného programu ze systému.

Editace programu

Tato akce je dostupná pomocí zobrazení seznamu programů (viz předchozí podkapitola) prostřednictvím tlačítka „Editovat“ u příslušného záznamu. Zde se nalézají čtyry záložky (viz obr. C.4).

V **první záložce** se definuje název, popis, aktivita a přístup k programu. **Ve druhé záložce** se definují vstupní parametry do programu, kdy se nejprve zadá název parametru, poté jeho popis, datový typ a eventuálně seznam hodnot oddělené středníkem. Podobným způsobem se definují i výstupní parametry v rámci třetí záložky. **Poslední záložka** slouží pro nahrání skriptu, který program spouští. Zde se klikne na tlačítko „Nahrát nový skript“ a pomocí modálního okna dojde k jeho uložení na server. Tento skript musí splňo-



Obrázek C.4: Editace vstupních parametrů do programu

vat několik požadavků, aby mohlo dojít k jeho spuštění a řádnému ukončení včetně předání a uložení výsledků do databáze. **Ukázku takového skriptu** můžete vidět na straně 121.

C.2.5 Správa úloh

Úloha je struktura, která byla vytvořena na základě existujícího programu. Vytvořenou úlohu lze přiřadit do některého z balíků, které následně jsou předány pacientovi k procvičení.

Vytvoření úlohy

Vytvoření nové úlohy je možné kliknutím na kartu „Úlohy“ a záložku „Vytvořit novou úlohu“ v hlavním menu. Následně dojde k přesměrování do formuláře, kde vyplní název nové úlohy, její popis a typ programu. Poté se úloha založí a přejde se do její **editace** (viz dále).

Zobrazení seznamu úloh

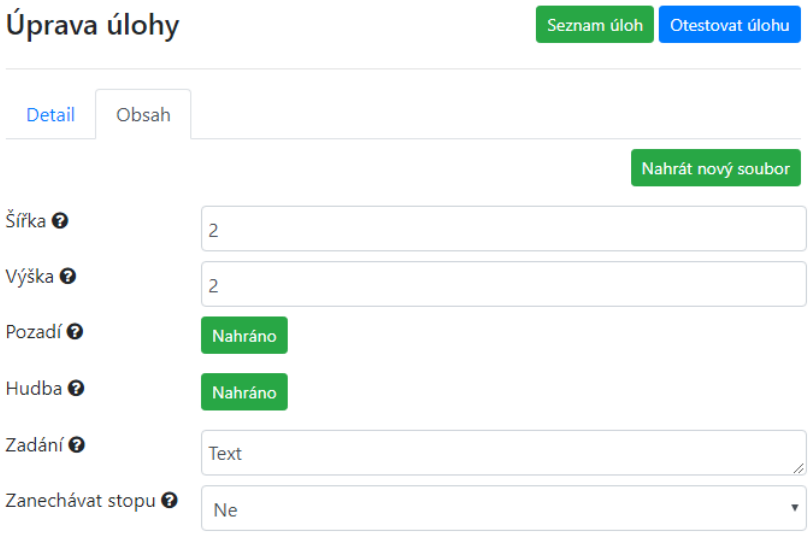
Akce vedoucí k seznamu úloh, ke kterým má daný uživatel přístup. Do tohoto tabulkového výpisu se lze dostat pomocí hlavního menu kliknutím na kartu „Úlohy“ a dále na záložku „Správa úloh“. Nad zobrazeným seznamem lze provádět základní filtrace, které jsou dostupné pod filtrem v pravém horním rohu editoru. Nad záznamy v seznamu jsou povoleny základní operace jako **duplikace**, **editace**, **náhled**, a **smazání** úlohy. Povolené operace závisí na vztahu uživatele k úlohám. Editovat úlohu může pouze její autor a smazat

ji lze pouze v případě, pokud nebyla dosud přiřazena některému z pacientů (resp. přiřazena do balíku).

Editace úlohy

V editaci úlohy může její autor měnit její název, popis, stavové informace zda ji mohou používat i ostatní terapeuti a zda je úloha aktivní. Dále zde dochází k vyplnění vstupních parametrů (viz obr. C.5), které byly nadefinovány v editaci programu.

Součástí editoru jsou tlačítka pro otestování vytvořené úlohy a odkaz na seznam vytvořených úloh.



Obrázek C.5: Editace úlohy

C.2.6 Obecná funkcionalita

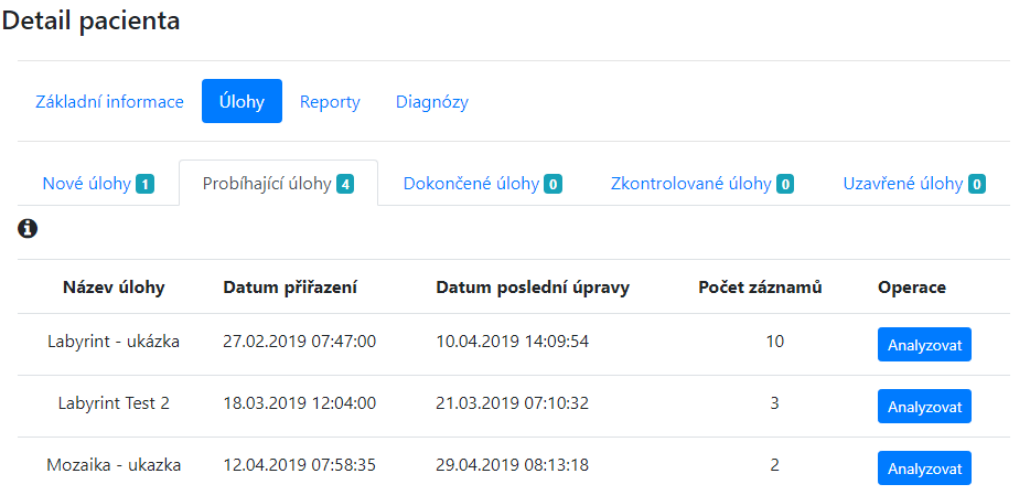
Každý administrátor má také umožněny operace, které byly uvedeny v kapitole obecná funkcionalita (viz kap. C.1, str. 108). Mezi tyto operace patří přihlášení, obnovení hesla, připomínkování, nastavení účtu a jazyka.

C.3 Terapeut

V této části bude popsána funkcionalita, kterou má povolenou každý terapeut.

C.3.1 Správa pacientů

Každý terapeut má možnost zobrazení svých pacientů včetně jejich veřejných údajů a všech úloh, které jim byly zadány. Tato operace je dostupná z hlavního menu kliknutím na kartu „Pacienti“ a následně na záložku „Správa pacientů“. Poté se zobrazí jednoduchý **seznam pacientů**, u kterých si může nechat zobrazit jejich detail. V tomto detailu (viz obr. C.6) jsou veškeré pacientovy úlohy, jeho základní informace, reporty a diagnózy.



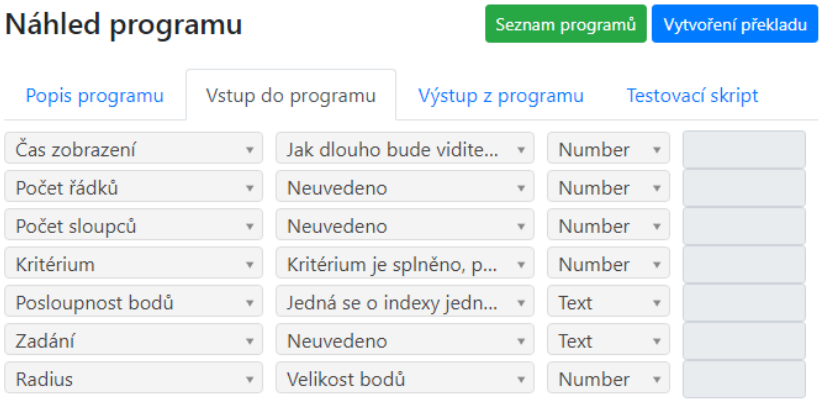
Obrázek C.6: Detail pacienta

Přidání a zobrazení reportu

Každý terapeut má možnost přidávat hodnocení svých pacientů, která si pacienti mohou následně prohlížet. Tato funkcionalita je přístupná pomocí zobrazení „Detailu pacient“, která byla vysvětlována ve správě pacientů. Zde se nalézá záložka **Reporty**, kde je tabulkový výpis všech dosud vytvořených reportů, které terapeut pacientovi vytvořil. Tabulka obsahuje základní informace o reportu včetně informace, zda pacient daný report už viděl. Report je možné si nechat **zobrazit** nebo **smazat**.

C.3.2 Náhled programu

Terapeut má na rozdíl od administrátor a super-terapeuta možnost pouze nahlížet na vytvořené programy, kam se dostane pomocí hlavního menu kliknutím na kartu „Programy“ a následně na záložku „Správa programů“. Poté je uživatel přesměrován do náhledu programu (viz obr. C.7), kde má k dispozici náhled všech vstupní a výstupních parametrů včetně jejich popisu. Dále zde uživatel může najít popis programu.



Obrázek C.7: Náhled programu

C.3.3 Správa úloh

Jedná se o funkcionalitu, kterou mají terapeuti společnou s administrátory a super-terapeuty. Tato funkcionalita byla již popsána v kapitole Správa úloh na straně 113.

C.3.4 Správa balíku

Balík je tzv. kontejner, do kterého jsou přidávány úlohy, které obsahují vazbu na konkrétního pacienta. Všechny aktivní balíky, které jsou nové nebo právě prováděné pacientem, jsou pacientovi dostupné k procvičení.

Vytvoření balíku

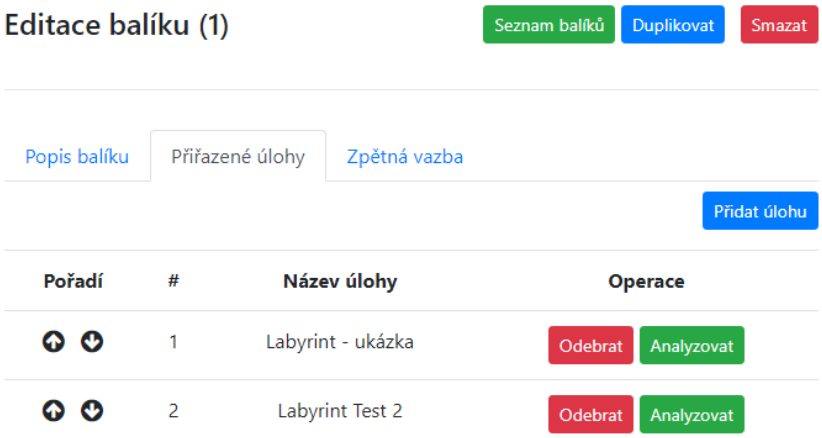
Tato možnost je dostupná z hlavního menu pod kartou „Balíky“ a poté záložkou „Vytvořit balík“. Dále je uživatel přesměrován do formuláře, kde vyplní název balíku a ze seznamu vybere terapeuta, kterému bude balík přiřazen. Následně dojde k jeho vytvoření a přesměrování do editace (viz obr. C.8).

Zobrazení seznamu balíků

Zobrazení tabulkového výpisu všech balíků daného terapeuta je dostupné z hlavního menu pod kartou „Balíky“ a dále záložkou „Správa balíků“. Zde se terapeutovi nabídne seznam vytvořených balíků, které odpovídají zadanému filtru v pravém horním rohu obrazovky. V rámci tohoto okna je umožněno vytvářen nové balíky. V rámci tabulkového výpisu jsou dostupná tlačítka pro **editaci** a **mazání** konkrétních balíků.

Editace balíku

Do editace balíku se lze dostat prostřednictvím akce zobrazení všech balíků, která byla popsána v předchozí kapitole. Zde je uživateli umožněno nastavit jeho název, stav a aktivitu (pacient má viditelné pouze aktivní a nové/prováděné balíky). V jeho editaci je mu umožněno přidávat/odebírat nové úlohy do/z balíku, provádět analýzy nad jednotlivými úlohami nebo celý balík duplikovat pro jiného pacienta. Při **přidávání úloh** se terapeutovi zobrazí pouze ty úlohy, ke kterým má přístup - aktivní úlohy, které sám vytvořil a veřejné úlohy jiných terapeutů. Při **duplikaci** je nutno vybrat v modálním okně pacienta, kterému tento balík duplikujeme. Následně jsme přesměrovány do nového balíku.



Obrázek C.8: Editace balíku

C.3.5 Analýza výsledků

Do analýzy úlohy se lze dostat kliknutím na kartu „Analýzy“ a záložku „Analýza“ v hlavním menu. Následně je uživateli umožněno pomocí tří filtrů dohledat cílovou úlohu a kliknutím na tlačítko „Analyzovat“ se přepnout do její analýzy. Do analýzy úlohy se lze také dostat pomocí již popisovaných akcí „Správa pacientů“ a „Správa balíku“, kdy v editaci balíku, resp. seznamu přiřazených úloh, klikneme na tlačítko analyzovat.

Následně dojde k přesměrování do editoru s tabulkovým výpisem výstupních parametrů z jednotlivých spuštění dané úlohy.

C.3.6 Obecná funkcionalita

Každý terapeut má také umožněny operace, které byly uvedeny v kapitole obecná funkcionalita (viz kap. C.1, str. 108). Mezi tyto operace patří přihlá-

šení, obnovení hesla, připomínkování, nastavení účtu a jazyka.

C.4 Super-terapeut

Tato role je určena pro mírné odlišení od klasického terapeuta, od kterého veškerou funkcionalitu přejímá. Proto se na tuto funkcionalitu pouze odkážu na kapitolu C.3 na straně 114.

Super-terapeut má na rozdíl do terapeuta umožněno **vytvářet nové programy**. Z toho vyplývá, že má také umožněno **spravovat překlady** v aplikaci.

C.4.1 Správa překladů

Tato funkcionalita byla také popsána v rámci role administrátor v kapitole C.2.3 na straně 110.

C.4.2 Správa programů

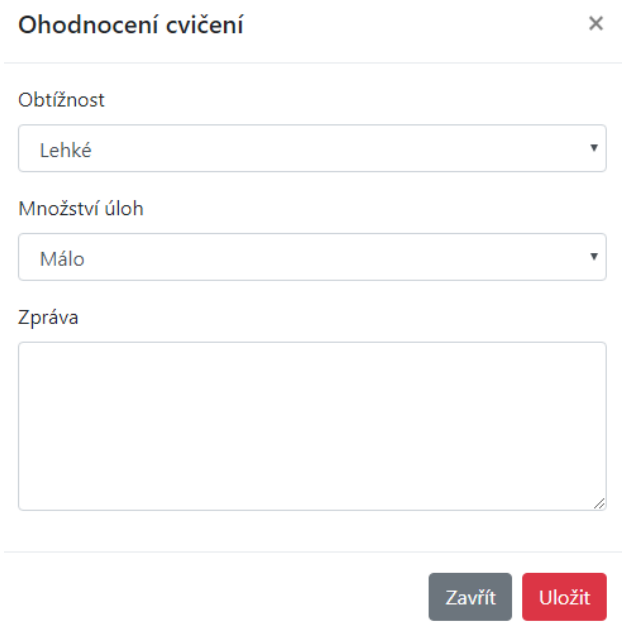
Správa programů byla představena v kapitole popisující funkcionalitu administrátora (viz kap. C.2.4, strana 112).

C.5 Pacient

Jedná se o roli, pro kterou je tato aplikace určena. Tato funkcionalita není příliš obsáhlá. Pacientovi je umožněno pouze **spouštět připravené balíky** od terapeutů a zobrazovat si **zprávy od terapeutů**, které se týkají jejich pokroku v rehabilitaci. Dále je umožněno provádět veškeré operace, které byly popsány v kapitole obecná funkcionalita (viz kap. C.1, strana 108).

C.5.1 Spouštění připravených balíků

Tato akce je pacientovi dostupná z domovské stránky kliknutím na některý z balíků. Následuje jeho spuštění, kdy uživatel plní jednu úlohu za druhou. Úlohy lze přeskakovat (tlačítko „další“) nebo celý balík ihned ukončit kliknutím na tlačítko „Zpět na domovskou stránku“ vlevo nahoře. Po dokončení všech úloh se pacientovi zobrazí formulář, ve kterém může ohodnotit dokončený balík (viz obr. C.9).



Obrázek C.9: Formulář pro ohodnocení dokončeného cvičení.

C.5.2 Zobrazení zpráv od terapeuta

Tato možnost je dostupná pomocí hlavního menu kliknutím na kartu „Zprávy od terapeuta“ a následně na záložku „Reporty“. Poté se zobrazí jednoduchá tabulka, kde každý záznam obsahuje tlačítko „Zobrazit report“, které zobrazí požadovaný náhled přehledně v modální okně. Tento náhled obsahuje informaci o názvu reportu, popisu aktuálního stavu/pokroku pacienta a datum jeho vytvoření.

C.5.3 Obecná funkcionalita

Každý pacient má také umožněny operace, které byly uvedeny v kapitole obecná funkcionalita (viz kap. C.1, str. 108). Mezi tyto operace patří přihlášení, obnovení hesla, připomínkování, nastavení účtu a jazyka.

C.6 Uživatel bez role

Jedná se o roli, která je uživateli nastavena ihned po registraci do systému. Po registraci dojde k odeslání e-mailu administrátorovi aplikace, který danému uživateli nastaví jeho roli. Tento uživatel má povolenou pouze obecnou funkcionalitu, kterou jsem popisoval v kapitole C.1 na straně 108. Mezi tyto operace patří přihlášení, obnovení hesla, připomínkování, nastavení účtu a

jazyka.

D Zdrojové kódy

```
1  //Funkce pro spusteni ulohy
2  function runExercise(param) {
3      //Kod pro ziskani vstupnich parametru z textoveho retezce
           a spusteni ulohy
4      .
5      .
6      .
7      showModalAfter() //Funkce, ktera se vola po ukonceni
           ulohy pro prechod na dalsi ulohu. Tato funkce vola
           funkci finish().
8  }
9
10 //Funkce, ktera provede odeslani vysledku na server
       nastavenim konkretnich elementu (Finished, OutpuParameters
       ) potvrzenim formulare s daty.
11 function finish() {
12     var resultString = ""; //Vystupni retezec, ktery bude na
           serveru zpracovan a ulozen
13     var items = 0; //Pocet spusteni dane ulohy (pocita se s
           jednim spustenim)
14
15     for (var i = 0; i < mResults.length; i++) {
16         resultString += mResults[i].fault;
17         resultString += "##"; //oddelovac parametru v ramci
               ulohy
18         resultString +=  mResults[i].time;
19         resultString += ";;"; //oddelovac vysledku v ramci
               vicero spusteni ulohy (pocita se s jednim
               spustenim)
20     item ++;
21     }
22     if(item != 0){
23         document.getElementById("Finished").value = true; //
               hodnota, ktera rozhoduje, zda dojde k zapisu
               vysledku do databaze
24     }
25     document.getElementById("OutputParameters").value =
           resultString; //Vystup z programu
26     document.getElementById("ExerciseSubmit").submit(); //
           Odeslani vysledku na server
27 }
```

Listing D.1: Šablona pro vytvoření vstupní a výstupní funkce programu.

E Obsah CD

Přiložené CD obsahuje tři adresáře a jeden textový soubor **Obsah_CD.txt**, kde je popis odevzdávané adresářové struktury. V prvním adresáři **Poster** jsou umístěny soubory s posterem ve formátu `pdf` a `pub`. Ve druhém adresáři **src** je umístěný celý projekt. Poslední adresář (**text**) obsahuje text diplomové práce ve formát `PDF` spolu s adresářem `Latex`, kde jsou jeho zdrojové LaTeXsoubory