FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

# Zadání bakalářské práce

| | |
|---|---|
| **Název:** | Informační systém pro fyzioterapeuty - serverová část |
| **Student:** | Jakub Volák |
| **Vedoucí:** | Ing. Filip Glazar |
| **Studijní program:** | Informatika |
| **Obor / specializace:** | Webové a softwarové inženýrství, zaměření Webové inženýrství |
| **Katedra:** | Katedra softwarového inženýrství |
| **Platnost zadání:** | do konce letního semestru 2022/2023 |

## Pokyny pro vypracování

Cílem práce je navrhnout a realizovat webový informační systém pro fyzioterapeuty. Systém by měl především umožňovat správu klientů, kontrolu nad provedenými úkony jednotlivých pacientů a komplexní přehled o léčbě. Tato práce se zabývá především návrhem samotného informačního systému. Klientskou část paralelně zpracovává bakalářská práce Michala Kováře.

Postupujte dle následujících kroků:

1) Analyzujte požadavky možných klientů aplikace
2) Na základě požadavků navrhněte informační systém
3) Konzultujte návrh s Michalem Kovářem
4) Zvolte vhodné technologie pro realizaci projektu
5) Implementujte prototyp aplikace
6) Otestujte implementaci vhodnými automatizovanými testy
7) Propojte softwarové řešení s klientskou částí od Michala Kováře

---

*Elektronicky schválil/a Ing. Jaroslav Kuchař, Ph.D. dne 17. prosince 2021 v Praze.*

Bakalářská práce

# INFORMAČNÍ SYSTÉM PRO FYZIOTERAPEUTY - SERVEROVÁ ČÁST

**Jakub Volák**

Fakulta informačních technologií
Katedra softwarového inženýrství
Vedoucí: Ing. Filip Glazar
4. května 2022

# Obsah

# Seznam obrázků



# Seznam tabulek

# Seznam výpisů kódu

*Chtěl bych poděkovat především panu Ing. Filipu Glazarovi za vedení mé bakalářské práce a za jeho zkušenosti a rady, které mi pomohly při implementaci systému. Dále bych chtěl poděkovat mému kolegovi Michalovi Kovářovi za možnost podílet se na vývoji systému pro fyzioterapeuty a za perfektní týmovou spolupráci. Poslední obrovské poděkování patří celé mé rodině a přátelům, kteří mě podporovali po celou dobu mého studia.*

# Prohlášení

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

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. Dále prohlašuji, že jsem s Českým vysokým učením technickým v Praze uzavřel dohodu, na jejímž základě se ČVUT vzdalo práva na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona. Tato skutečnost nemá vliv na ust. § 47b zákona č. 111/1998 Sb., o vysokých školách, ve znění pozdějších předpisů.

V Praze dne 4. května 2022 .....

# Abstrakt

Tato bakalářská práce se zabývá návrhem a vývojem serverové části webového informačního systému pro soukromé fyzioterapeuty. Na začátku práce je proveden sběr informací a požadavků od fyzioterapeutů pomocí internetového dotazníku a jejich analýza. Podle této analýzy je proveden návrh systému. Ten se věnuje rozboru vhodných technologií pro vývoj moderní webové aplikace. Návrh definuje strukturu a části systému, včetně databázového modelu. Součástí návrhu je analýza obecné směrnice pro ochranu osobních údajů. Práce obsahuje porovnání navržené aplikace s konkurenčními systémy. Závěr práce se věnuje detailnímu popisu nejdůležitějších a nejzajímavějších částí implementace systému, včetně popisu způsobu testování aplikace. Poslední kapitola práce zmiňuje nasazení systému a možné vylepšení systému v budoucnosti.

Výsledkem je funkční prototyp informačního systému, který zjednoduší práci fyzioterapeutům a urychlí léčbu klientů. Systém může klient i fyzioterapeut používat v českém a anglickém jazyce. Systém implementuje bezpečnostní požadavky směrnice o ochraně osobních údajů.

**Klíčová slova** webový informační systém, prototyp, fyzioterapeut, API, Laravel

# Abstract

This bachelor thesis deals with the design and development of the server part of a web information system for private physiotherapists. At the beginning of the work is made analysis of information and requirements from physiotherapists which are collected using an Internet questionnaire. According to this analysis, a system design is performed. System design deals with the analysis of suitable technologies for the development of modern web applications. The design defines the structure and parts of the system, including the database model. The design includes an analysis of the general data protection directive. The work contains a comparison of the proposed application with competing systems. The conclusion of the thesis deals with a detailed description of the most important and interesting parts of the system implementation, including a description of how to test the application. The last chapter mentions the deployment of the system and possible improvements to the system in the future.

The result is a functional prototype of an information system that simplifies the work of physiotherapists and speeds up the treatment of clients. Both the client and the physiotherapist can use the system in Czech and English. The system implements the security requirements of the Data Protection Directive.

**Keywords** web information system, prototype, physiotherapist, API, Laravel

# Seznam zkratek

| | |
|---|---|
| API | Rozhraní pro programování aplikací |
| ČVUT | České vysoké učení technické |
| FP | Funkční požadavek |
| GDPR | Obecné nařízení o ochraně osobních údajů |
| HTTP | Hypertextový přenosový protokol |
| HTTPS | Zabezpečený hypertextový přenosový protokol |
| JSON | Zápis objektů v javascriptu |
| MVC | Model-View-Controller |
| MVP | Model-View-Presenter |
| NGINX | Webový server |
| NoSQL | Nejenom SQL |
| NP | Nefunkční požadavek |
| REST | Rozhraní pro předávání dat |
| SOAP | Jednoduchý protokol pro přístup k objektu |
| SPA | Jednostránková aplikace |
| SQL | Strukturovaný dotazovací jazyk |
| VPS | Soukromý virtuální server |

# Úvod

Fyzioterapie je zdravotní obor, který se věnuje komplexní péči o zdraví. Zabývá se především léčením pohybového ústrojí, které se každému z nás po dobu života opotřebovává. Proto si myslím, že většina z nás fyzioterapeuta za svůj život alespoň jednou navštíví. Jejich práce je pro nás všechny důležitá, nicméně není jednoduchá. Každý fyzioterapeut potřebuje ke své práci mnoho informací, převážně pak osobní údaje svých klientů, metody a způsoby léčby pohybového aparátu nebo seznam rehabilitačních cviků. Je tedy patrné, že všechny tyto informace si nelze zapamatovat a je potřeba si je někde ukládat v přístupné a jednoduché formě. Pro zpracování těchto informací budou moct fyzioterapeuti využít webový informační systém, jehož vývojem se tato bakalářská práce bude zabývat.

Téma webového informačního systému pro fyzioterapeuty zpracovávám spolu se studentem Fakulty informačních technologií ČVUT v Praze Michalem Kovářem. Jeho bratr je vystudovaný fyzioterapeut a k vývoji informačního systému nás přivedl právě on. Věděli jsme, že vytvořit takový systém nebude jednoduché. Zároveň jsme ale věděli, že se systém bude reálně používat, a že by mohl být pro fyzioterapeuty skvělým nástrojem pro urychlení a usnadnění práce. S kolegou Michalem jsme využili příležitosti a začali přemýšlet, jak takový systém realizovat.

Systém bude tvořen několika moduly. Základní částí systému bude karta klienta, která bude fyzioterapeutovi poskytovat všechny informace o klientovi a jeho léčbě. Důležitou komponentou bude kalendář, ve kterém si bude fyzioterapeut zaznamenávat schůzky s klienty. Dalším modulem bude databáze cviků, kterou bude fyzioterapeut používat při zadávání úkolů klientům. Kromě fyzioterapeuta bude mít přístup do systému i klient. Ten ve svém přehledu nalezne informace o následující schůzce, stav léčby nebo zadané cvičení na doma.

Struktura mé práce kopíruje pracovní postup, který jsem použil při vývoji systému. V první kapitole definuji cíle práce. Následně popíši sběr požadavků od fyzioterapeutů a jejich analýzu. Výstupem této kapitoly budou funkční a nefunkční požadavky. Následovat bude analýza konkurenčních řešení. V další kapitole pomocí funkčních a nefunkčních požadavků provedu návrh systému, včetně výběru vhodných technologií. Pokračovat budu popisem vlastní implementace a uvedu také způsob testování. V závěru práce se budu věnovat rozvoji systému v budoucnosti.

# Kapitola 1
# Cíl práce

Hlavním cílem práce je navrhnout a implementovat funkční prototyp serverové části webového informačního systému pro fyzioterapeuty.

Prvním z cílů je sběr a analýza požadavků od možných uživatelů aplikace pomocí dotazníku. Výstup poskytne informace o pracovní náplni fyzioterapeutů a jejich klientech. Pomocí zpracovaných informací bude možné splnit další cíl, kterým je návrh informačního systému. Tato práce bude řešit návrh převážně serverové části, která bude data zpracovávat. Návrh klientské části souběžně řeší práce studenta FIT ČVUT v Praze Michala Kováře. Podle návrhu se následně provede implementace systému. Po implementaci se vytvoří automatické testy serverové části, které ověří funkčnost prototypu.

Na závěr dojde k propojení s klientskou částí a nasazení celé aplikace do testovacího provozu. Výsledná aplikace přinese zjednodušení práce samotných fyzioterapeutů a urychlení léčby klientů.

**Sběr a analýza požadavků** V první části práce bude popsáno, jak probíhal sběr požadavků od možných budoucích uživatelů. Výstupem bude analýza nasbíraných informací a definování funkčních a nefunkčních požadavků. Ty budou zpracovány a využity v následující části věnované návrhu systému.

**Návrh systému** V návrhu budou popsány části systému včetně datového modelu. Následně se zvolí vhodné technologie pro realizaci a dojde i na analýzu existujících řešení. V závěru této části se budeme věnovat bezpečnosti a GDPR.

**Implementace serverové části** Praktická část této práce se bude věnovat samotné implementaci serverové části systému pomocí zvolených technologií. Budou popsány ty nejdůležitější a nejzajímavější části kódu. Kromě toho budou uvedeny problémy, které bylo potřeba během vývoje vyřešit. Součástí bude i ukázka zpracovávaných dat.

**Testování a nasazení systému** Závěrečná část popíše, jak probíhalo testování vytvořeného systému a propojení s klientskou částí. Budou ukázány některé typy testů a jejich výstupy. Nakonec bude obecně popsáno nasazení a zprovoznění celého systému jako celku na serveru.

**Rozvoj** Na konci této práce budou uvedeny nedokončené části prototypu, které bude potřeba před uvedením do reálného provozu dodělat. Zároveň bude diskutováno, jak prototyp v budoucnu vylepšit.

# Kapitola 2

# Sběr a analýza požadavků

*V této kapitole budou nejprve probrány zvažované metody sběru dat pro tento projekt. Následně bude popsán výsledný formulář, který byl zaslán potencionálním uživatelům systému a dojde také k analýze získaných dat. Z této analýzy budou definovány funkční a nefunkční požadavky.*

## 2.1 Sběr informací

Před začátkem celého projektu bylo potřeba získat povědomí o tom, jak takový fyzioterapeut pracuje. Proto jsme hledali způsob, jak co nejlépe získat potřebné informace. Předtím jsme ale potřebovali alespoň obecně definovat strukturu aplikace. Abychom mohli vůbec přemýšlet o struktuře systému, potřebovali jsme se zorientovat v lékařském názvosloví. S tím nám pomohl fyzioterapeut Roman Kovář — bratr mého kolegy Michala. Ten nás zasvětil do problematiky fyzioterapie, především pak do zmíněného názvosloví. Následně jsme mohli začít přemýšlet, jakým způsobem oslovit další fyzioterapeuty.

Ve svém okolí znám nejméně jednoho fyzioterapeuta, který by byl ochotný nám poskytnout informace. Dohromady jsme tak měli několik fyzioterapeutů. Nabízela se proto možnost sejít se s nimi osobně. To nám ze začátku přišlo jako dobrá možnost. Taková osobní schůzka by nám umožnila individuální otevřenou diskusi s každým fyzioterapeutem. Připravili bychom si sadu otázek, na které by nám fyzioterapeut odpověděl. Zároveň bychom s ním mohli diskutovat o změnách nebo vylepšeních. Bohužel tyto pohovory se neuskutečnily. Uvědomili jsme si, že by to bylo pro obě strany časově náročné. Navíc každý fyzioterapeut byl z jiné části republiky, takže bychom museli i cestovat, což by nám zabralo ještě více času. Kromě toho mít informace pouze od několika osob je pro obecný systém málo. Museli jsme proto najít jinou možnost, jak získat informace od více fyzioterapeutů.

Podle nás bylo nejlepším řešením vytvořit on-line formulář. Ten jsme mohli poslat jak našim známým fyzioterapeutům, tak i dalším. Navíc jej naši fyzioterapeuti dále přeposlali svým kolegům. Tohle řešení bylo jednoduché na realizaci, protože na internetu existuje několik aplikací pro vytvoření on-line formulářů. Zároveň splnilo svůj účel a poskytlo nám větší uživatelskou základnu.

### 2.1.1 Struktura formuláře

Předtím, než jsme mohli formulář zaslat respondentům, museli jsme si definovat, jaké údaje a informace potřebujeme získat. Nejvíce nás zajímalo, jaké informace si každý fyzioterapeut ukládá o každém svém klientovi. Kromě těchto informací nás ale také zajímalo, zda má fyzioterapeut svůj seznam cviků nebo encyklopedii léčby. Dále jsme se ptali na způsob zpracování

a ukládání zjištěných dat o klientovi nebo způsob zadávání a kontroly domácího cvičení. Kromě výše zmíněných údajů jsme taky zjišťovali věkovou strukturu léčených klientů.

Do formuláře jsme museli zahrnout i vizi našeho projektu. V první části jsme proto dotázanou osobu seznámili s obecným návrhem systému. Návrh byl rozdělen do pěti částí podle navržených modulů aplikace. Moduly tak byly představeny zvlášť a jejich názvy jsou: kartotéka, kalendář, databáze cviků, encyklopedie léčby a informace pro klienta. U každého modulu se mohl fyzioterapeut vyjádřit, zda by danou část aplikace při své práci využil. Následovalo několik doplňujících otázek k představované části. Tyto otázky obvykle cílily na možné rozšíření systému nad rámec základního fungování nebo detailněji specifikovaly způsob zpracování dat.

Výsledný formulář, tak jak je popsán výše jsme vytvořili v aplikaci Google Forms. Aplikace nám umožnila nejenom formulář zaslat fyzioterapeutům v on-line podobě, ale také nám umožnila formulář vyhodnotit a zobrazit některé výstupy v grafické podobě. Výsledný formulář lze nalézt v příloze A této práce. Souhrn všech odpovědí byl vyexportován a je uložen na přiloženém médiu.

## 2.2 Analýza požadavků

Následující část práce se bude věnovat vyhodnocení vyplněných dotazníků. Celkově nám přišly čtyři vyplněné dotazníky od fyzioterapeutů. Počet respondentů není vysoký, nicméně fyzioterapie je velmi úzce zaměřený obor, takže i nízký počet odpovědí je dostatečný k objektivnímu popisu dané problematiky. Protože je formulář rozdělen do pěti částí podle modulů, bude tak rozdělena i analýza. Postupně zhodnotíme odpovědi respondentů k jednotlivým modulům.

### 2.2.1 Kalendář

Podle výsledků z formuláře to bude jedna z nejdůležitějších a nejpoužívanějších částí aplikace. Většina dotázaných totiž odpověděla, že je kalendář relevantní a že jej při své práci určitě využije. Z toho je jasně patrné, že kalendář v našem systému nesmí chybět. Kromě této základní otázky jsme se ptali i na další doplňující informace, které se kalendáře týkají. Důležitým údajem pro nás je, jak dlouho obvykle trvá schůzka s klientem. Fyzioterapeuti uvedli, že délka schůzky bývá od 30 minut až po 2 hodiny. To nám určuje, jakým způsobem budeme muset ukládat časový údaj o události v kalendáři. Kromě dne budeme muset ukládat i čas, protože během dne může mít fyzioterapeut domluveno více schůzek. Dále jsme se ptali, jakým způsobem si aktuálně fyzioterapeut vede svůj kalendář. Na tuto otázku většina odpověděla, že používá papírový diář. Z toho jsme usoudili, že může být pro fyzioterapeuta občas obtížné se v takovém diáři orientovat, tak aby na žádnou schůzku nezapomněl, případně aby věděl o jakého klienta se jedná. To by se ale v jednoduchém elektronickém kalendáři nemělo stát, takže použití tohoto nástroje by mělo být pro fyzioterapeuty přínosné. Jeden dotázaný fyzioterapeut nám však navíc odpověděl, že také používá papírový diář, nicméně že se v brzké době chystá používat rezervační systém. To nám znovu potvrzuje, že je kalendář v aplikaci potřebný.

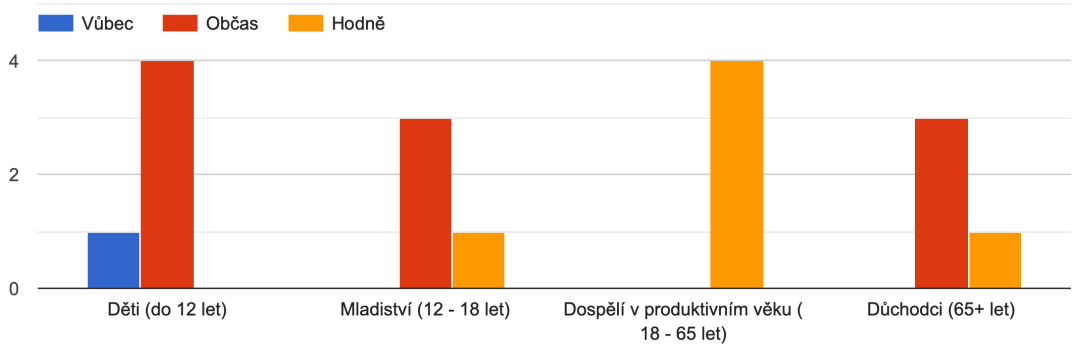
Poslední důležitou informací je, jaké informace potřebují fyzioterapeuti okamžitě vědět o plánované schůzce, když si ji v kalendáři zobrazí. Kromě data a času schůzky všichni dotázaní potřebují vidět i jméno a příjmení klienta. Několik dalších pak uvedlo, že by ocenili, kdyby se v kalendáři u plánované schůzky zobrazoval i kontakt na klienta, tedy e-mail nebo telefonní číslo, pro případ změny termínu schůzky ze strany fyzioterapeuta.

### 2.2.2 Kartotéka

Nedílnou součástí našeho systému bude i kartotéka. Stejně jako u kalendáře většina respondentů odpověděla, že kartotéku při své práci využijí. Už z této informace je jasné, že je potřeba tento modul do aplikace zakomponovat. Dále jsme potřebovali zjistit věkovou strukturu klientů, abychom věděli, jaké informace můžeme od klientů vyžadovat a ukládat. Z průzkumu vyplynulo,

že nejčastěji dotázané fyzioterapeuty navštěvují lidé v produktivním věku (18-65 let) a děti do 12 let. Celé rozdělení věkové struktury klientů znázorňuje graf na obrázku 2.1. Z toho vyplývá, že pokud je klient dítě nebo nezletilá osoba, je potřeba u osobních údajů ukládat i údaje o rodiči jakožto zákonném zástupci. U osobních údajů respondenti uvedli, že si uchovávají jméno, příjmení, datum narození klienta a kontaktní údaj, převážně telefonní číslo. E-mail, jako kontaktní údaj, nemůže být povinný, protože některé děti nebo důchodci nemusí e-mail vůbec používat. Ostatní osobní údaje jako je například pohlaví, rodné číslo, bydliště a další, může systém dle některých fyzioterapeutů ukládat, ale nesmí je nutně vyžadovat.

Kromě osobních údajů je ale dle fyzioterapeutů potřeba ukládat i lékařské zprávy, které obsahují aktuální popis zdravotního stavu klienta. Většina fyzioterapeutů by si lékařskou zprávu vyfotila na svůj telefon a z něj pak fotografii nahráli do aplikace. Další také uvedli, že by dokument naskenovali pomocí skeneru, případně by si jej nechali poslat na e-mail jako přílohu.




**Obrázek 2.1** Graf návštěvnosti fyzioterapeuta v jednotlivých věkových kategorií

### 2.2.3 Databáze cviků

Následující modul, který jsme navrhli, je databáze neboli seznam cviků. Primárním cílem tohoto modulu je zpřístupnit klientům popis a náhled cviků, které mají cvičit doma. Tento seznam cviků si bude každý fyzioterapeut vytvářet v aplikaci sám.

Podle odpovědí z formuláře by tuto část většina fyzioterapeutů při práci spíše využila. Jeden z respondentů uvedl, že by tuto část aplikace ale vůbec nepoužil. Z odpovědí plyne, že modul nebude stěžejní částí systému, ale bude pro práci fyzioterapeutů užitečný. Aby ale splnil svůj účel, potřebovali jsme vědět, jak nejlépe ukládat popis cviků, aby byl pro klienty srozumitelný, názorný a zajímavý. Všichni dotázaní se shodli, že nejvhodnější způsob pro prezentaci cviků skrze obrazovku bude kombinace obrázků a videa. Musíme tedy fyzioterapeutům umožnit ke každému cviku nahrát multimediální obsah. Dále jsme navrhli rozdělení cviků do kategorií. Tento návrh nám 100% dotázaných schválilo. Celý seznam tak bude systematicky strukturován, což přispěje k jeho přehlednosti a umožní rychlejší vyhledávání.

### 2.2.4 Encyklopedie léčby

Námi navržený modul encyklopedie léčby by podle nás mohl být také pro fyzioterapeuty užitečný. Hlavní myšlenkou je možnost zpětně nahlédnout do způsobu léčby klientů podle komplexního souboru zdravotnických informací, dále jen anamnézy. Fyzioterapeut by tak mohl zužitkovat zkušenosti z předchozích léčení a lépe tak určit způsob rehabilitace u nového klienta.

Ze zaslaných odpovědí vyplývá, že by většina respondentů modul při práci nějakým způsobem využila. Bohužel ze získaných informací už nedokážeme zjistit, co by se v modulu mělo změnit

nebo vylepšit, aby jej fyzioterapeuti chtěli více využívat. Různé pohledy na tuto část aplikace potvrzují i výsledky doplňující otázky, která se týká vyhledávání v encyklopedii. Někteří fyzioterapeuti by vyhledávali podle nežádoucího stavu neboli diagnózy, nebo způsobu léčby. Zatímco jiní by procházeli záznamy podle osobních údajů klienta nebo typu klienta (sportovec, dítě, důchodce, ...). Není tedy jasné, který ze způsobů vyhledávání upřednostnit a jak celý modul správně uchopit.

Encyklopedii léčby i přes to do systému zakomponujeme. Bude to ale část prototypu, která se bude muset v budoucnosti upravit, tak aby více odpovídala požadavkům fyzioterapeutů. Naše základní řešení této části aplikace bude jednoduché a připravíme jej na možné změny.

### 2.2.5 Informace pro klienta

Poslední modul aplikace bude část systému, do které budou přistupovat samotní klienti. Klient se bude muset při vstupu do této části přihlásit svými přihlašovacími údaji. Uvnitř následně nalezne informace od svého fyzioterapeuta potřebné ke své léčbě.

Opět jsme se ptali všech fyzioterapeutů na to, zda modul při práci využijí. Většina odpověděla, že jej bude používat, nicméně si myslí, že ne každý klient bude do systému vstupovat. Jedná se především o klienty v důchodovém věku. Někteří důchodci totiž nemusí umět pracovat na počítači, dokonce jej nemusí ani vlastnit. To ale neznamená, že bychom modul neměli do systému zapojit. Nyní si ale pojďme říct, jaké informace chtějí podle formuláře fyzioterapeuti zpřístupnit svým klientům. Nejčastější odpovědí byl termín další schůzky. S tím naprosto souhlasíme a bylo by na škodu, kdyby takový údaj v přehledu chyběl. Mezi další zveřejněné informace by měly patřit záznamy schůzek, tedy popis změn zdravotního stavu. Poslední důležitou informací bude domácí cvičení. Každý dotázaný fyzioterapeut nám potvrdil, že svým klientům zadává cviky, které mají cvičit doma. Domácí cvičení má zefektivnit rehabilitaci a zkrátit dobu potřebnou pro uzdravení. K tomu je potřeba, aby klient nejen dobrovolně cvičil doma, ale aby zadané cviky prováděl správně. A právě k tomuto účelu použijeme seznam cviků, o kterém jsme se bavili výše v oddílu 2.2.3. Díky databázi cviků umožníme fyzioterapeutovi sestavit individuální cvičební plán na doma pro každého klienta. Fyzioterapeut tak bude mít jistotu, že klient ví, jaké cviky má doma cvičit a že je bude cvičit správně. Kromě toho taky klientovi nabídneme vyplnění zpětné vazby k zadaným cvikům. Pomocí zpětné vazby si fyzioterapeut ověří, že klient rehabilituje i doma. Zpětnou vazbu si navíc může fyzioterapeut i sám vyhodnotit a připravit si tak podklady na další schůzku s klientem.

Většina fyzioterapeutů doplňující službu individuálního cvičebního plánu ocenila, zároveň také připustili, že zpětná vazba z domácího cvičení by mohla kvalitu léčby zlepšit.

## 2.3 Požadavky na systém

V závěru první kapitoly si určíme funkční a nefunkční požadavky systému, které vyplývají z provedené analýzy nasbíraných informací od fyzioterapeutů. Funkční požadavky (FP) budou detailně specifikovat základní procesy a funkcionality. Budou tak určovat kompletní chování systému. Nefunkční, neboli obecné požadavky (NP) definují očekávané vlastnosti vytvářeného systému.

### 2.3.1 Funkční požadavky

#### FP1 - Karta klienta

Systém musí fyzioterapeutovi umožnit vytváření nových klientů. Zároveň musí systém umožňovat úpravu osobních údajů již existujícího klienta. Dále systém musí umožňovat odstranění všech informací o klientovi. Fyzioterapeut může v kartě klienta ukládat dokumenty potřebné k léčbě. Tyto

dokumenty si může později ze systému stáhnout nebo je vymazat. Systém musí fyzioterapeutovi umožnit export osobních údajů klienta do souboru. Fyzioterapeut může v systému definovat mezi klienty vztah rodič-dítě. Karta dětského klienta musí zobrazovat informace o rodiči. Systém v kartě klienta bude zobrazovat průběh léčby.

## FP2 - Autentizace uživatele

Vstup do systému musí být umožněn pouze přihlášenému uživateli. Přihlášení do systému musí být jednotné jak pro fyzioterapeuta, tak pro klienta. Přihlašovací údaje se budou skládat z e-mailu a hesla. Systém musí rozlišovat roli přihlášeného uživatele tak, aby neumožnil vstup klientovi do sekcí pro fyzioterapeuta a obráceně.

## FP3 - Profil uživatele

Každý uživatel musí mít možnost provést v systému aktualizaci svých přihlašovacích údajů. Systém musí provádět kontrolu nových přihlašovacích údajů, aby nedošlo k zamezení přístupu do systému jinému uživateli. Není žádoucí, aby si uživatel mohl sám změnit uživatelskou roli.

## FP4 - Plánovač událostí

Systém musí obsahovat kalendář, který bude fyzioterapeutovi zobrazovat proběhlé a plánované události. Fyzioterapeut si může v systému definovat typy událostí. Dále může fyzioterapeut v systému vytvářet, upravovat a mazat události. Při vytváření a změně události musí systém kontrolovat obsazenost časového úseku v kalendáři. Každá událost musí být v systému přiřazena určitému klientovi. Kalendář musí fyzioterapeutovi umožňovat denní, týdenní a měsíční filtrování.

## FP5 - Záznamy léčby

Systém musí fyzioterapeutovi umožňovat průběžně zapisovat záznamy o provedené léčbě. Fyzioterapeut musí mít možnost do těchto záznamů vstupovat a provádět jejich změnu nebo smazání. Každý záznam musí být pevně spojen s určitým klientem a zároveň musí obsahovat časový údaj o zápisu léčebného záznamu. Fyzioterapeut musí mít možnost definovat si vlastní vzhled záznamu.

## FP6 - Databáze cviků

Systém umožňuje fyzioterapeutovi vytvářet, upravovat a mazat cviky. Ke každému cviku může fyzioterapeut přidat URL odkaz nebo nahrát multimediální soubory. Soubory si může fyzioterapeut i klient ze systému stáhnout, fyzioterapeut může provést i odstranění souborů. Fyzioterapeut si může v systému definovat kategorie cviků. Cviky může následně do těchto kategorií přiřadit, aby usnadnil orientaci v databázi cviků.

## FP7 - Encyklopedie léčby

Fyzioterapeut si může v systému definovat, upravovat a mazat štítky označující určitou diagnózu. Štítky může fyzioterapeut přiřazovat nebo odebírat klientům v jejich kartě. Systém následně umožňuje vyhledávat klienty a jejich historii léčby podle názvu štítku.

### FP8 - Úkoly

Systém umožňuje fyzioterapeutovi vytvářet úkoly pro klienty. Tyto úkoly může fyzioterapeut upravovat nebo mazat. Každý úkol musí být přiřazen k události, na které byl úkol zadán a klientovi, který jej má vykonat. Do úkolu lze vkládat cviky z databáze cviků. Fyzioterapeut může u každého cviku v úkolu určit dobu cvičení nebo počet opakování. Klient může úkol v systému upravit vyplněním zpětné vazby. Zpětná vazba musí obsahovat text a hodnocení složitosti cviku. Fyzioterapeutovi bude zpětná vazba zobrazena v detailu zadaného úkolu. Fyzioterapeut může klientovi vytvořit více úkolů.

### FP9 - Informace pro klienta

Systém bude obsahovat nástěnku klienta. Nástěnka bude obsahovat informace o následující schůzce, kontakt na fyzioterapeuta a přehled o léčbě. Tento přehled v sobě bude obsahovat zadané úkoly včetně cviků s multimediálním obsahem. Pokud přihlášený klient má v systému vztah k jinému klientovi, zobrazí systém seznam propojených klientů. Nástěnka klienta musí umožnit zobrazení přehledu léčby propojených klientů.

## 2.3.2 Nefunkční požadavky

### NP1 - Dostupnost

Fyzioterapeut i klienti potřebují co nejvyšší dostupnost systému. Je jasné, že systém bude využíván denně a jakýkoliv výpadek by zkomplikoval práci fyzioterapeuta. Ten potřebuje během schůzky přistupovat k datům klienta, aby mohl zapisovat změny v průběhu léčby. Zároveň také potřebuje zapisovat nové schůzky do kalendáře. Nemůže se proto stát, že by byl systém delší dobu mimo provoz. Pro případ výpadku systému je potřeba provádět pravidelnou zálohu nejdůležitějších dat, aby bylo možné systém co nejrychleji obnovit. Protože bude systém používán ve webovém prohlížeči, musí být systém dostupný ve všech moderních prohlížečích, aby nebyl fyzioterapeut ani klienti nuceni k používání určitého softwaru.

### NP2 - Výkon

Výkon je u tohoto systému definován jako doba reakce na požadavek. Serverová část musí co nejrychleji odpovídat na požadavky klientské části tak, aby byla zajištěna co nejmenší odezva systému. Zároveň musí systém produkovat co možná nejmenší zátěž na hardware, aby byl schopen provozu i na levnějším a slabším zařízení. Systém musí být schopen uložit data minimálně o 50 000 klientech a obsloužit minimálně 50 přihlášených klientů v jeden okamžik.

### NP3 - Bezpečnost

Systém bude zpracovávat a uchovávat osobní údaje a citlivé informace o léčbě. Je proto nezbytně nutné systém řádně zabezpečit a používat výhradně šifrovanou komunikaci mezi serverovou a klientskou částí systému.

# Kapitola 3
# Analýza konkurenčních řešení

Nyní když víme, jaké funkcionality bude náš systém poskytovat a z jakých částí se bude skládat, je vhodné analyzovat již existující aplikace. Na internetu jsme objevili celkem 3 konkurenční systémy, které se zaměřují na doménu fyzioterapeutů, a které obsahují některé společné prvky s námi vytvářeným systémem. Konkurenční řešení jsou převážně placené a zkušební verze nejsou veřejně dostupné, proto se při analýze budeme odkazovat na webové prezentace konkurenčních produktů.

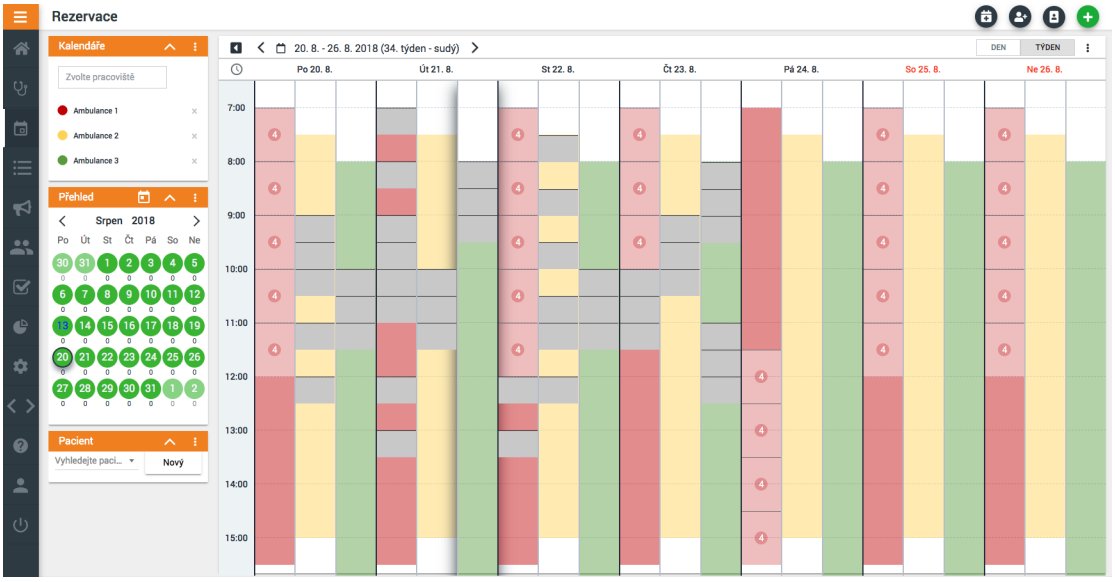
## 3.1 RehaSys

Prvním konkurenčním řešením je aplikace s názvem RehaSys, která je vyvíjena českou firmou ARTiiS GROUP, a.s. se sídlem v Brně. Tato aplikace se na své webové stránce prezentuje jako „Rehabilitační systém zítřka“. [1] Na stránkách je dále uvedeno, že systém je vhodný pro zdravotnické zařízení. Aplikace totiž není určena pouze pro fyzioterapeuty. Kromě nich může aplikaci používat i recepční, pacient nebo majitel. Na webové prezentaci systému jsou také vyjmenovány 3 agendy, které aplikace zajišťuje. Jsou to:

- plánování procedur,
- vedení zdravotnické dokumentace,
- vykazování odvedené práce pojišťovnám.

Je patrné, že náš systém se zde potkává s jednou agendou, kterou je zdravotnická dokumentace. V dalších položkách se už aplikace neshodují, takže tento systém nenabízí stejné možnosti jako námi vytvářená aplikace. Z předchozího popisu je navíc patrné, že systém je pro soukromé fyzioterapeuty příliš robustní a nabízí funkce, které by pravděpodobně soukromý fyzioterapeut ani nevyužil. Pořízení takového systému pro soukromého fyzioterapeuta by pravděpodobně bylo i velice nákladné.

Dle zmíněných skutečností si myslíme, že naše aplikace by měla být pro soukromé fyzioterapeuty, v porovnání se systémem RehaSys, lepší volbou.




**Obrázek 3.1** Rezervace v aplikaci RehaSys [2]

## 3.2 Systém LBIS/4G

Dalším konkurenčním řešením od společnosti LAURYN s.r.o. je aplikace LBIS/4G. Ta je primárně určena pro klasické léčebné lázně, lázeňská sanatoria, lázeňské hotely, rehabilitační a léčebné ústavy nebo rehabilitační zařízení. [3] Oproti předchozímu systému RehaSys nabízí více modulů, které zajišťují kompletní správu celého zdravotnického zařízení. Aplikace LBIS/4G zajišťuje následující agendy:

- ubytovací agendu,
- zdravotní agendu,
- rehabilitační agendu,
- stravovací a skladovací agendu,
- ekonomickou agendu,
- manažerský reporting.

Ze seznamu agend je na první pohled patrné, že pro soukromého fyzioterapeuta je tento systém nevhodný, i přes to že obsahuje jednu část, kterou obsahuje i náš systém. Stejně jako u předchozího konkurenčního systému i u této aplikace je mnoho funkcí, které fyzioterapeut nevyužije. Navíc je systém tak robustní, že by obsluha aplikace mohla být pro jednoho fyzioterapeuta složitá. Dále lze také odhadnout, že by tento systém byl pro fyzioterapeuta příliš nákladný.

Stejně jako v předchozím případě si myslíme, že by fyzioterapeut měl dát přednost námi vytvářené aplikaci před tímto systémem.

## 3.3 PhysiApp

Posledním konkurenčním řešením je mobilní aplikace PhysiApp pro Android a iOS od společnosti Physitrack PLC. Ta se svým obsahem a funkcemi nejvíce přibližuje námi vytvářenému systému.

Na oficiální webové stránce totiž najdeme informace o tom, že aplikace poskytuje databázi cviků. [4] Uživatel dostane od fyzioterapeuta sadu cviků, které má cvičit doma. Cvičení může uživatel ohodnotit a odeslat zpětnou vazbu fyzioterapeutovi. Tato funkce je totožná s jednou částí našeho systému. Nicméně tato aplikace je velmi úzce zaměřená pouze na domácí cvičení. Nelze v ní ukládat zdravotní informace klienta nebo plánovat osobní schůzky. Pro zpracování těchto informací by fyzioterapeut potřeboval další software, takže by při své práci musel používat více aplikací a přesouvat data mezi nimi. Kromě toho mobilní aplikace není v češtině, takže pro některé klienty by mohl být problém aplikaci vůbec používat.

Protože aplikace nabízí databázi cviků a funkci zpětné vazby, můžeme se aplikací inspirovat. Myslíme si ale, že pro nás není úplně konkurencí, protože nenabízí kompletní správu pracovních procesů fyzioterapeuta.

## 3.4 Shrnutí analýzy konkurenčních řešení

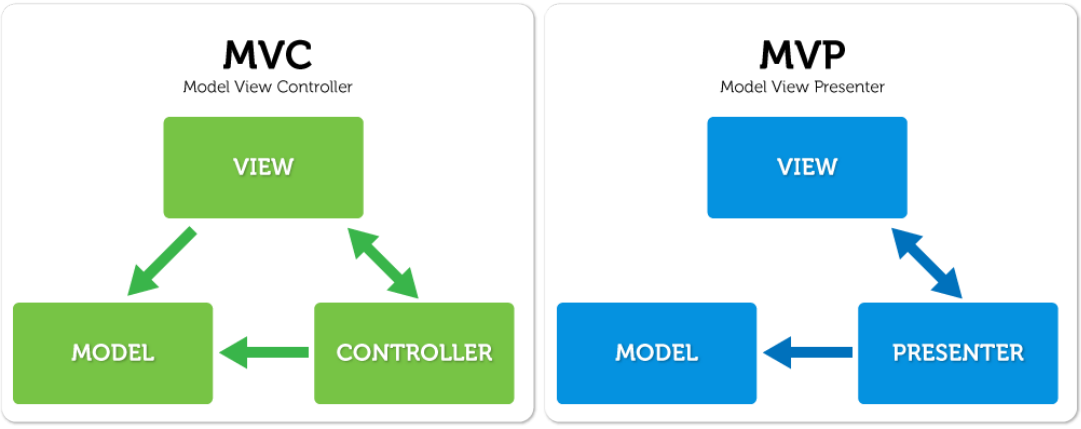
Z provedené analýzy vyplývá, že se na trhu vyskytují převážně aplikace pro zdravotní zařízení, které kromě fyzioterapie poskytují i jiné typy léčby. Tyto aplikace nabízí mnoho funkcí, které soukromý fyzioterapeut obvykle při své práci nevyužije. Kvůli robustnosti těchto systémů bude pravděpodobně vysoká pořizovací cena, což by pro soukromé fyzioterapeuty mohl být problém. Námi vytvářený systém bude nabízet pouze funkce, které fyzioterapeuti opravdu využijí. Zároveň bude systém jednoduchý na použití, takže nebude fyzioterapeuta zdržovat při práci. Navíc nabídne i možnost urychlení léčby díky zpětné vazbě. Myslím si, že naše aplikace bude konkurenceschopná, a že bude soukromými fyzioterapeuty využívaná.

# Kapitola 4

# Návrh systému

Na začátku před samotnou implementací bylo nutné rozmyslet si, jaké technologie a postupy pro vývoj použít. Protože jsem aplikaci vytvářel s kolegou Michalem, bylo potřeba systém rozdělit na dvě samostatné části, které jsme zpracovali každý zvlášť a následně spojili do funkční aplikace. Rozhodli jsme se proto rozdělit systém na serverovou a klientskou část. Serverová část zpracovává data a poskytuje je klientské části skrze API. Klientská část pak data prezentuje v samostatné SPA. Systém byl tedy postaven podle architektury Model-View-Controller, dále jen MVC.

Architektura MVC je vytvořena tak, že každý pohled je zodpovědný za vykreslení dat. Kontroler zpracovává akce uživatele a model se stará o business logiku. [5] (překlad vlastní) Proto backend (serverová část) obsahuje všechny datové modely, business logiku a kontrolery. Frontend (klientská část) pak vykresluje všechny pohledy a stará se o vzhled aplikace. Kromě architektury MVC existuje ještě architektura MVP, neboli Model-View-Presenter. Tuto architekturu jsme v tomto projektu nepoužili, protože jeho hlavním konceptem je struktura pohledu podle dat. [5] (překlad vlastní) To by v praxi znamenalo, že by serverová část určovala klientské části, jak má informace zobrazit uživateli. Pro tento projekt by ale taková architektura byla zbytečná, protože struktura dat je jednoznačná a za běhu programu neměnná.




**Obrázek 4.1** Schéma architekury MVC a MVP [6]

## 4.1 Rozbor použitých technologií

Po zvolení struktury a architektury aplikace jsem mohl začít vybírat vhodné technologie pro vývoj serverové části. Postupně jsem se zamýšlel nad programovacím jazykem, databází a frameworkem.

### 4.1.1 Programovací jazyk

Vhodných jazyků pro vývoj serverové části je celá řada. Já jsem si pro tento projekt vybral jazyk PHP. Jako každý programovací jazyk má své výhody a nevýhody oproti ostatním jazykům. Hlavní výhodou tohoto jazyka je, že na trhu existuje mnoho hostingů, které jsou pro tento jazyk přizpůsobeny. Cena takového hostingu je navíc nižší oproti ostatním hostingům pro jiné jazyky. To je dáno především tím, že PHP je multiplatformní. To znamená, že na serveru může být nainstalován libovolný operační systém. Další výhodou je fakt, že je určen právě pro programování webových stránek či aplikací. Porovnejme si ale PHP s některými jinými jazyky vhodnými pro vývoj webových aplikací.

Prvním vybraným jazykem je Python. Ten se v poslední době stal mezi programátory velmi oblíbeným jazykem, převážně kvůli své jednoduchosti. Jeho hlavní výhodou oproti jiným jazykům je zjednodušení komplexního programování. Kromě toho je také multiplatformní, stejně jako PHP. Kde ale Python zaostává jsou frameworky. PHP nabízí celou řadu frameworků, každý pro různé typy projektů a aplikací. [7] (překlad vlastní) Pokud chceme, aby byl projekt udržitelný a bylo možné jej rozvíjet, je vhodné postavit aplikaci na nějakém frameworku. Z tohoto důvodu jsem dal přednost PHP před Pythonem, aby bylo možné vybírat z většího množství frameworků.

Dalším silným konkurentem pro PHP je moderní jazyk C#, který byl vyvinut společností Microsoft. Jazyk C#, společně s frameworkem ASP.NET, je používán právě k vývoji webových aplikací. Mezi jeho největší výhody patří rychlost. Z tohoto důvodu je používán převážně v komplexních a náročných systémech. Oproti PHP má ale jednu velkou nevýhodu, a to že C# není multiplatformní. To znamená, že aplikace napsané v C# potřebují server s operačním systémem Windows. Pro provoz aplikace je tedy potřeba hosting určený právě pro C# systémy. [8] (překlad vlastní) Takový hosting může být o dost nákladnější na provoz, hlavně kvůli vyšším požadavkům na zdroje serveru. Protože naše aplikace je určená pro soukromé fyzioterapeuty, je potřeba udržet co nejpříznivější provozní cenu. Aplikace psaná v C# by tuto cenu zvýšila právě kvůli drahému hostingu. Proto jsem se rozhodl dát přednost PHP před C#.

Posledním vybraným konkurentem je Node.js, což je runtime prostředí vytvořeno pro vyhodnocování skriptů v jazyce javascript. Ten je určen pro frontend. Node.js ale běží na straně serveru, takže přináší několik funkcionalit z frontendu na backend. Mezi nejzajímavější patří asynchronní vyhodnocení kódu. Node.js je tak ideální pro aplikace, které běží v reálném čase, a které potřebují zpracovávat velké množství vstupních a výstupních dat. Příkladem můžou být různé chatovací aplikace nebo aplikace pro streamování videa. Zároveň Node.js perfektně pracuje s NoSQL databázemi jako je například MongoDB. [9] (překlad vlastní) Nicméně náš systém nepotřeboval zpracovávat data v reálném čase ani ukládat nestrukturované informace. Proto použití Node.js v tomto projektu nebylo vhodné.

### 4.1.2 Databáze

Pro uchování dat a informací bylo potřeba k systému připojit databázi. Na trhu je několik různých databází, které bylo možné v tomto projektu použít. Mezi nejznámější patří MySQL, PostgreSQL a Oracle. Oracle je velmi kvalitní a spolehlivá databáze, která je na trhu už několik desítek let. Bohužel je placená, takže zakoupení licence by zvýšilo cenu aplikace, což by mohl být pro naší cílovou skupinu fyzioterapeutů problém. Proto jsem ji ze stejného důvodu jako jazyk C# zamítl.

PostgreSQL je objektově-relační databáze, která nabízí spolehlivost, robustní funkce a výkon. Kromě toho je ale možné využít i NoSQL přístup. Databáze je open source, takže je možné ji

bezplatně používat. [10] (překlad vlastní) Nicméně pro náš systém, který bude používat soukromý fyzioterapeut a několik desítek jeho klientů, by byla PostgreSQL databáze zbytečně silná. Pro ukládání menšího nebo středního množství strukturovaných dat bylo vhodnější použít jednoduchou databázi, která obsahuje základní funkce. Na to se perfektně hodila databáze MySQL. MySQL, stejně jako PostgreSQL, je open source takže jsem ji mohl bezplatně použít. Oproti PostgreSQL je MySQL databáze čistě relační, což bylo pro naši aplikaci naprosto dostačující. Pokud by se systém v budoucnu začal zvětšovat a bylo by potřeba ukládat i nestrukturované typy dat, lze poměrně rychle databázi vyměnit. Systém byl totiž postaven na frameworku, takže při změně databáze bude stačit vyměnit databázový driver pro komunikaci s příslušnou databází. Při vývoji prototypu jsem si tedy vystačil s MySQL databází.

### 4.1.3 Distribuce dat

Zpřístupnění dat pro klientskou část vyžadovalo vytvoření API na serverové části. Na světě aktuálně existují dva nejznámější přístupy pro tvorbu API. Jsou to REST a SOAP. REST je anglická zkratka pro representational state transfer, což lze česky popsat jako rozhraní pro předávání dat. Protokol SOAP je zkratka pro simple object access protocol, v češtině tedy jednoduchý protokol pro přístup k objektu. Rozdíl mezi těmito technologiemi je ten, že SOAP je oficiální protokol spravovaný konsorciem W3C. Naproti tomu REST je soubor architektonických principů. [11] (překlad vlastní)

Protokol SOAP byl navržen tak, aby umožňoval komunikaci aplikacím psaných v různých jazycích nebo provozovaných na různých platformách. Z tohoto důvodu je nutné, aby byl protokol komplexní. Kvůli komplexnosti je ale SOAP bezpečný, konzistentní a izolovaný. Díky těmto vlastnostem je používán převážně v enterprise systémech, které vyžadují přesně definovaný způsob komunikace. Všechny vlastnosti protokolu jsou sice přínosné, bohužel ale jejich kombinace způsobuje snížení rychlosti přenosu dat. [11] (překlad vlastní) Právě kvůli tomuto problému se SOAP nehodí pro webové aplikace, které zobrazují informace zákazníkovi. Jedná se například o rezervační systémy, informační systémy nebo e-shopy. V dnešní době hraje každá vteřina načítání webové stránky svou roli. Nikdo ze zákazníků nechce čekat na načtení stránky, a proto potřebujeme rychlost přenosu co nejvíce urychlit. Nemůžeme si tedy dovolit být omezeni technologií pro distribuci dat, takže jsem se rozhodl protokol SOAP v tomto projektu nepoužít.

Pro naši aplikaci bylo lepší použít REST. Ten se v praxi používá právě pro programování webových služeb. Jeho největší výhodou je jednoduchost. [12] (překlad vlastní) Protože to není protokol, ale soubor principů, nechává programátorovi více volného prostoru pro definování komunikace s danou aplikací. REST využívá zavedený protokol HTTP, který dnes používá celý internet. Tento protokol používá 5 klíčových slov (GET, POST, PUT, PATCH, DELETE), které určují, jakou akci má backend provést. Kromě toho REST nespecifikuje formát odpovědi. Serverová část tak může vrátit třeba HTML stránku, soubor nebo JSON. [12] (překlad vlastní) Backend výše uvedené funkcionality potřeboval využívat, proto jsem se rozhodl pro vytvoření REST API.

### 4.1.4 Framework

Závěrečným úkolem byl výběr vhodného frameworku, ve kterém jsem mohl vytvořit serverovou část naší aplikace. Musel jsem vybrat takový framework, který bude v jazyce PHP, bude možné jej připojit k databázi MySQL a bude možné v něm vytvořit REST API. Na trhu existuje mnoho frameworků, které tyto požadavky splňují. Já jsem vybíral mezi Symfony, Laravelem a Lumenem. Myslím si, že tyto frameworky jsou aktuálně jedny z nejpoužívanějších pro vývoj webových aplikací. Každý, z výše zmíněných frameworků, nabízí ale rozdílné funkcionality a programovací přístupy. Bylo tedy vhodné si je navzájem porovnat před finálním výběrem. Ještě bych ale citací vysvětlil, co to vlastně framework je. „Softwarový framework je platforma pro vývoj softwarových aplikací. Poskytuje základ, na kterém mohou vývojáři softwaru vytvářet programy

pro konkrétní platformu. Framework může také zahrnovat knihovny kódů, kompilátor a další programy používané v procesu vývoje softwaru.“ [13]

Prvním zmíněným frameworkem je Symfony. Symfony je podle mě jedním z největších a nejstabilnějších frameworků. Jeho velká komunita udržuje Symfony stále aktuální. Kromě toho je framework stále vylepšován, takže přináší nové moderní funkcionality. Tyto funkcionality jsou obvykle schovány v knihovnách. Symfony pro ukládání dat do databáze používá objektově relační mapování, dále jen ORM. Komunikaci s databází pak zajišťuje knihovna Doctrine. [14] (překlad vlastní) Doctrine pro práci s databází využívá architektonický vzor data mapper. Data mapper je objekt, který provádí zápis, úpravu nebo smazání dat z databáze. Doctrine umí komunikovat s většinou známých databází, tedy i s MySQL. Symfony tak splňuje požadavek našeho systému pro komunikaci s databází. Symfony ale neobsahuje pouze tuto knihovnu, jejich daleko víc. Množství použitých knihoven v Symfony zvětšuje robustnost, což znamená, že se tento framework nehodí pro každý projekt. Na webové stránce CoolClub [13] je zmíněno, že Symfony je ideální pro použití u projektů, které mají vyšší složitost. Další zdroj [15] (překlad vlastní) také zmiňuje, že se Symfony používá převážně pro enterprise systémy. Myslím si, že naše aplikace je velmi úzce zaměřená na jedno odvětví, takže nemusím předpokládat, že se systém bude v budoucnu enormně rozšiřovat. Rozhodl jsem se tedy framework Symfony nepoužít, protože jeho možnosti zbytečně přesahují potřeby našeho systému.

Následující framework byl Lumen. Lumen je bratr Laravelu, což byl další nabízený framework. I když oba tyto frameworky pochází ze stejné rodiny, mají mezi sebou jeden velký rozdíl. Každý je vhodný na úplně jiný druh aplikací. Interně oba frameworky využívají stejné knihovny, jako například Eloquent, který se používá pro komunikaci s databází. Ten, stejně jako Doctrine, využívá ORM, ale jiný architektonický vzor. Eloquent používá vzor active record. „Active Record je objekt, který nese data i chování.“ [16] I přes stejné knihovny, Laravel nabízí více možností oproti Lumenu. Lumen byl totiž vytvořen pouze pro podporu bezstavových API. Je tedy vhodný pro programování malých webových služeb, které mají na starost specifické úkoly. Laravel, který také obsahuje podporu pro bezstavové API, nabízí další funkce. Mezi ty nejdůležitější patří šablonovací systém, relace uživatele (session) nebo zpracování formulářových požadavků. [17] (překlad vlastní) Backend naší aplikace poskytuje bezstavové REST API, takže by se mohlo zdát, že Lumen by byl dobrou volbou. Protože bylo ale nezbytné zpracovávat data z frontendu, potřeboval jsem využít zpracování formulářových požadavků. Kromě toho si myslím, že šablonovací systém se v budoucnu při rozšiřování systému může hodit. Například pro stylizaci emailových zpráv. Bylo tedy vhodnější dát přednost Laravelu před Lumenem, aby bylo možné systém v budoucnosti jednoduše rozšiřovat.

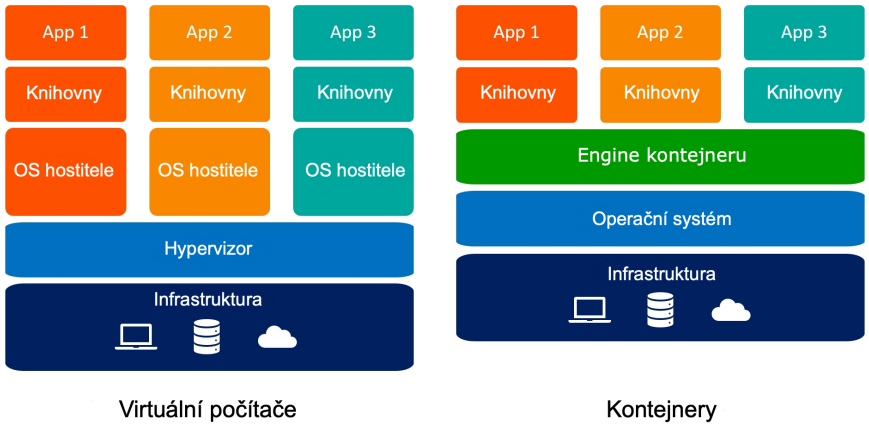
### 4.1.5 Docker

Při implementaci serverové části jsem použil několik programů a nástrojů usnadňujících vývoj. Nejdůležitějším pomocníkem byl jednoznačně Docker. Ten se v dnešní době podle mě používá při vývoji téměř každé moderní aplikace. Docker je open source platforma pro vytváření, nasazování a správu kontejnerizovaných aplikací. [18] (překlad vlastní) Co to ale je kontejnerizovaná aplikace? Na webu IBM [18] (překlad vlastní), které zveřejnilo článek o Dockeru, se dočteme, že Docker umožňuje vývojářům zabalit své aplikace do tzv. kontejnerů, což jsou standardizované spustitelné komponenty, které kombinují zdrojový kód aplikace s knihovnami operačního systému a závislostmi potřebnými ke spuštění tohoto kódu v jakémkoli prostředí. Tato možnost je pro vývojáře velmi užitečná. Už při raném vývoji si může programátor pomocí kontejnerů definovat prostředí, ve kterém bude aplikace spuštěna při ostrém nasazení. Programátor tak má naprostou jistotu, že jeho kód bude na skutečném serveru spustitelný, a že při nasazení nedojde ke komplikacím, například s kompatibilitou. Tohle ovšem není jediný přínos.

Díky kontejnerům si programátor nemusí do svého počítače instalovat všechny programy potřebné pro spuštění své aplikace. Jde například o webový server, databázový server nebo jiný další software. Všechny tyto programy si vývojář nakonfiguruje pro každou svou aplikaci

zvlášť v kontejnerech, které pak spustí se svou aplikací v Dockeru pomocí orchestračního nástroje Docker Compose. Ten spustí všechny kontejnery najednou a postará se o jejich komunikaci, takže celý systém ani nepozná, že je rozdělen do více částí. Pokud už programátor nepotřebuje svou aplikaci dále vyvíjet, může kontejnery jednoduše smazat. Při obnovení vývoje lze kontejnery znovu sestavit pomocí konfiguračního souboru DockerFile. [18] (překlad vlastní) Takový postup by byl u běžně nainstalovaných programů problém, protože reinstalace a konfigurace by zabrala spoustu času. Navíc pokud je systém vyvíjen v týmu, je velmi jednoduché sdílet konfigurační soubor. Každý vývojář je pak schopen sestavit projekt z totožných kontejnerů bez nutné další konfigurace.

Kontejnerizace ale není jediný způsob, jak si při vývoji vytvořit prostředí pro spuštění aplikace. Existují totiž virtuální počítače. Virtuální počítač je úplně stejný jako normální počítač. Obsahuje procesor, paměť, úložiště a další komponenty. Ve virtuálním počítači nejsou tyto komponenty fyzickým hardwarem, jsou reprezentovány kódem. Tohoto principu je možné docílit virtualizací. „Virtualizace je proces vytvoření softwarové neboli virtuální verze počítače s vyhrazenými objemy CPU, paměti a úložiště, které jsou vypůjčené z fyzického hostitelského počítače.“ [19] Tím se dostáváme k hlavnímu rozdílu mezi virtuálním počítačem a kontejnerizací. Z definice virtuálního počítače je patrné, že virtuální počítač běží v operačním systému fyzického počítače. Virtuální počítač pak obsahuje svůj vlastní operační systém, ve kterém teprve dochází ke spuštění aplikace. Zatímco kontejnerizace využívá přímo operační systém fyzického počítače. Z tohoto popisu by mělo být patrné, že kontejnerizace je méně náročnější na zdroje fyzického počítače, takže je pro vývoj aplikací vhodnější.




**Obrázek 4.2** Architektura virtuálních počítačů a kontejnerů [20]

Použití Dockeru při implementaci bylo velmi jednoduché. Laravel totiž obsahuje balíček Sail, který přináší jednoduché rozhraní příkazového řádku pro ovládání základního Docker vývojového prostředí pro Laravel. [21] (překlad vlastní) Balíček Sail již obsahuje připravený soubor docker-compose ve formátu YAML, který definuje kontejnery pro webový a databázový server. Celé spuštění aplikace jsem pak prováděl pouze pomocí jednoho příkazu. Můj kolega Michal tuto funkci také ocenil, protože jako vývojář frontendu se nemusel starat o konfiguraci backendu a instalaci potřebného software pro spuštění serverové části. Abych si implementaci ještě více zpříjemnil, modifikoval jsem část konfiguračního souboru týkající se webového serveru tak, aby bylo možné provádět ladění backendu. Úprava nebyla složitá, protože postup pro debugování je popsán v dokumentaci.

Na závěr bych chtěl všem vývojářům Docker doporučit, protože se jedná o velmi užitečný nástroj, který dokáže vývoj urychlit a zpříjemnit.

## 4.2 Databázový model

Po výběru všech vhodných technologií jsem navrhl databázový model. Vytvoření modelu ještě před začátkem implementace bylo velmi důležité, protože jsem si ujasnil, jaké entity budou používány v jednotlivých modulech našeho systému. Návrh modelu jsem průběžně konzultoval s kolegou Michalem, aby i on byl obeznámen se strukturou dat v systému. Navržené entity nyní popíšu a uvedu jejich vztah k danému modulu. Schéma databázového modelu jsem vytvořil v aplikaci dbdiagram.io [22] a lze jej nalézt v příloze B.

### Klient

Začnu největší entitou, kterou je klient (client). Entita klienta v sobě obsahuje převážně osobní informace o klientovi jako je například rodné číslo, pohlaví, výška a váha nebo adresa bydliště. Většina z nich je nepovinná. Povinné je pouze jméno, příjmení, datum narození a kontaktní telefon. Mezi povinné údaje jsem nemohl zařadit e-mail, protože mezi klienty jsou i děti nebo důchodci, kteří nemusí e-mailovou adresu vlastnit. Dále pak entita klienta umožňuje fyzioterapeutovi uložit informace o celkové anamnéze, sportovních aktivitách nebo prodělaných nemocech a úrazech. Tato entita je tedy základním stavebním kamenem pro kartu klienta v systému.

### Personál

Od klienta se o moc neliší entita personál (staff), která v systému reprezentuje fyzioterapeuta. Při návrhu jsem přemýšlel nad tím, že by do systému mohlo přistupovat více fyzioterapeutů nebo jiní pomocní pracovníci. Z tohoto důvodu se entita nejmenuje fyzioterapeut. Entita je tedy už od návrhu nachystána na možný rozvoj systému. Uvnitř obsahuje základní informace o fyzioterapeutovi jako je jméno, příjmení a kontaktní telefon. Tyto údaje se zobrazují převážně v modulu informace pro klienta.

### Uživatel

Entita uživatel (user) v systému zajišťuje přihlášení. Obsahuje proto dvojici přihlašovacích údajů e-mail a heslo. Zároveň taky obsahuje roli, aby systém při přihlášení rozpoznal, jestli se přihlašuje fyzioterapeut nebo klient. Uživatel pak obsahuje identifikátor buď klienta nebo personálu, aby bylo možné v systému po přihlášení zobrazit data napojené na danou entitu.

### Událost a typ události

Další větší entitou je událost (event). Z názvu je patrné, že tato entita je využívána v kalendáři. Fyzioterapeut si pomocí této entity v kalendáři vytvoří schůzku s klientem. Entita proto obsahuje název schůzky, datum a čas začátku a konce, poznámku a identifikátor klienta a fyzioterapeuta. Kromě toho může fyzioterapeut přiřadit události její typ. Typ události je malá přidružená entita. Díky ní si lze v aplikaci zavést různé typy událostí, jako třeba úvodní schůzka nebo školení.

### Záznam

Důležitou entitou pro uchování stavu léčby je záznam (record). Ten se váže k určité události. Lze tedy záznamy časově seřadit, aby bylo možné si projít celý průběh léčby. Do záznamu si fyzioterapeut může zapsat postup léčby, poznámku ke cvičení nebo libovolný strukturovaný text. Postup léčby je číselná hodnota, která vyjadřuje zlepšení nebo zhoršení stavu. Seznam těchto hodnot ze všech záznamů může být vhodný pro grafické zobrazení průběhu léčby v kartě klienta.

### Úkol

Další entita, která se váže k určité události je úkol (task). Fyzioterapeut může na každé schůzce vytvořit klientovi nový úkol pro domácí cvičení. K úkolu může fyzioterapeut zapsat jakýkoliv text, například s vysvětlením, jak doma cvičit. Úkol je pak propojen s cviky pomocí propojovací entity. V této entitě může fyzioterapeut specifikovat jak dlouho nebo kolikrát daný cvik opakovat. Klient pak může skrz tuto entitu poskytnout zpětnou vazbu ke každému cviku a ohodnotit složitost cviku.

### Cvik

Databáze cviků se bez entity cvik (exercise) neobejde. Entita umožňuje uložit název cviku, jeho popis nebo externí odkaz. Externí odkaz může fyzioterapeut použít libovolně. Může se tak odkázat například na video, kde je cvik prováděn, a nebo na on-line článek o cvičení. Ke cviku je taky možné nahrát své vlastní videa nebo fotky. Informace o souboru se uloží do přidružené entity soubor cviku (exercise file). Tato entita si ukládá pouze data nutné pro uložení a načtení souboru z interního úložiště aplikace. Fyzioterapeut může cvik přiřadit do kategorie, aby mohl využít filtrování v databázi cviků.

### Kategorie

Malá entita s velkým významem je kategorie (category). Její hlavní použití je v databázi cviků. Fyzioterapeut si může definovat své kategorie, do kterých si pak bude ukládat cviky. Lze odhadnout, že cviků bude celá řada, takže filtrování podle kategorií bude určitě užitečné. Kategorie má svůj název a barvu, pro rozlišení jednotlivých kategorií u cviků.

### Příloha

Příloha (attachment) je entita, která se využívá v kartě klienta. Její primární úkol je ukládat zdravotní dokumentaci, kterou klient poskytne fyzioterapeutovi od jiného lékaře. Převážně se bude jednat o soubory typu PDF nebo vyfocené lékařské zprávy.

### Štítek

Poslední navrženou entitou je štítek (tag). Ten se využívá ve vyhledávání v encyklopedii léčby. Fyzioterapeut si může v aplikaci zavést libovolné štítky, které reprezentují určitou diagnózu. Štítku lze přiřadit barvu, stejně jako u kategorie, aby bylo možné je barevně rozlišit. Fyzioterapeut může štítky přiřadit klientům s danou diagnózou. O propojení štítku s klientem se interně stará propojovací entita.

## 4.3 Obecné nařízení o ochraně osobních údajů

Při návrhu systému bylo potřeba myslet i na ochranu osobních údajů. Naše aplikace totiž uchovává osobní data a informace o klientech. Musel jsem se proto seznámit s nařízením Evropského parlamentu a Rady Evropské unie o ochraně fyzických osob v souvislosti se zpracováním osobních údajů, dále jen GDPR. [23] (překlad vlastní) Toto nařízení je samozřejmě dále upraveno českými zákony, které detailněji specifikují způsob ochrany. Po prostudování nařízení jsem zjistil, že systém musí obsahovat určité funkcionality, aby nařízení splňoval.

Každý z nás se již někdy setkal při registraci na webové stránce se zaškrtávací možností „souhlasím se zpracováním osobních údajů.“ Tento souhlas je požadován v rámci GDPR. V naší

aplikaci ale toto zaškrtávací políčko nenajdete. Důvodem je skutečnost, že klient se do systému nemůže sám registrovat. Informace o klientovi do systému vždy vyplňuje fyzioterapeut, který je povinen klienta obeznámit se zpracováním osobních údajů. Většina zdravotních středisek má pro tento účel vytvořen papírový formulář. Tento formulář obvykle obsahuje informace o tom, jak budou osobní údaje zpracovány a kdo k nim bude mít přístup. Pokud tedy bude chtít fyzioterapeut využívat naši aplikaci, bude si muset takový formulář vytvořit.

Další požadavky v nařízení se už ale naší aplikace týkají. Důležitou vyžadovanou funkcionalitou v systému je logování. Údaje o zdravotním stavu totiž patří do speciální skupiny ochrany osobních údajů [24], takže systém musí obsahovat log (soubor se záznamy) o tom, kdo, kdy a k jakým datům přistupuje. Směrnice nevyžaduje ovšem zaznamenávání každé události v systému. Pro splnění této podmínky tedy stačí, aby se zaznamenával pouze přístup k osobním údajům. V systému tedy zapisuji, kdo a kdy se do systému přihlásil a pod jakou rolí. Dále pak eviduji přístup a změnu v kartě klienta a přístup a změnu v záznamu léčby. Další data v systému se přímo netýkají osobních údajů, takže není potřeba uchovávat informace o přístupu.

Směrnice také určuje, jaké práva má klient vůči subjektu, který zpracovává jeho osobní údaje. Každý klient má právo na poskytnutí kopie svých osobních údajů. [23] (překlad vlastní) Tento požadavek naše aplikace splňuje tím, že poskytuje export osobních údajů do souboru. Soubor je následně dostupný fyzioterapeutovi, který jej může poskytnout klientovi na vyžádání. Kromě poskytnutí kopie osobních údajů má klient právo i na výmaz údajů ze systému, ale pouze pokud není jiný důvod pro další uchování. [23] (překlad vlastní) Systém samozřejmě umožňuje vymazat klienta, nicméně o nutnosti uchování osobních informací musí rozhodnout sám fyzioterapeut. K době uchování dat ještě doplním, že fyzioterapeut může uchovávat údaje o léčbě 5 let od posledního poskytnutí zdravotních služeb. [25] Po této době by měl data sám odstranit.

Poslední, co po našem systému GDPR vyžaduje, je technické zajištění integrity, dostupnosti a odolnosti systému. [23] (překlad vlastní) Důležitá je tedy zabezpečená komunikace mezi serverovou částí a klientskou. Protože se jedná o webový informační systém, vím, že komunikace bude probíhat skrze internet. Je tedy nezbytně nutné, aby tato komunikace byla šifrovaná. Šifrovanou komunikaci neřeší aplikace přímo uvnitř kódu, ale nastavení serveru. Při nasazení systému do reálného provozu bude nutné pořídit certifikát pro doménu, na které bude aplikace dostupná. Certifikát zajistí aplikaci možnost komunikovat pomocí zabezpečeného protokolu HTTPS. Kromě zabezpečené komunikace je potřeba omezit přístup do systému heslem. S tím už ale počítá návrh databázového modelu, takže tato podmínka je splněna. Dostupnost a integrita systému je důležitá v případě, že dojde k technické poruše na serveru. Při výpadku serveru může dojít ke ztrátě dat, takže je potřeba mít zálohu. Protože provádíme logování, je možné zálohu získat z logu. V dnešní době existuje několik nástrojů, které dokážou data z logu přečíst a vytvořit tak plnohodnotnou zálohu. Jsem si vědom, že log neobsahuje všechny data z celého systému, nicméně GDPR požaduje zálohu pouze osobních údajů, které jsou v logu systému obsaženy.

Serverová část splňuje veškeré požadavky GDPR, takže se fyzioterapeut nemusí obávat o ochranu osobních údajů svých klientů.

# Kapitola 5

# Implementace serverové části

Po dokončení návrhu systému jsem začal se samotnou implementací backendu. Nejprve jsem musel vytvořit API endpointy, abych mohl kolegovi Michalovi upřesnit, jak bude jeho klientská část získávat data z té serverové. Implementaci API jsem s Michalem průběžně konzultoval tak, aby předávaná data mohly být vhodně zpracována na frontendu.

## 5.1 API

Pro komunikaci serverové a klientské části bylo naprosto nezbytné vytvořit API. Implementace API se odvíjela od návrhu databázového modelu, takže musela umožňovat CRUD operace (create, read, update, delete) nad navrženými entitami. Jak už jsem zmínil v kapitole o návrhu systému, API jsem vytvořil pomocí REST architektonických principů. Každá adresa v systému má definovanou metodu, která specifikuje akci, kterou systém při zavolání provede. V systému používám převážně základní metody GET, POST, PUT, DELETE a PATCH. Metoda GET slouží pouze k získání dat ze serverové části. Pro vytvoření nového objektu je určena metoda POST. Pro upravení již existujícího objektu používám metodu PUT. Metoda PATCH také provádí změny v již existujícím objektu, nicméně upravuje pouze určité parametry, nepřepisuje celý objekt. Poslední používanou metodou je DELETE, kterou používám pro mazání objektů, jak je patrné z názvu. Kromě definování metod endpointů, bylo potřeba určit formát dat. V systému data zpracovávám a poskytuji ve formátu JSON. Tento formát je čitelný jak pro lidi, tak pro stroje. Zároveň je dnes velmi používaný ve webových aplikacích a frameworky s ním dokážou jednoduše pracovat. Ve výpisu kódu 5.1 je ukázáno, jak vypadá zápis dat entity záznam ve formátu JSON. Serverová část v odpovědi na požadavek nevrací pouze data v určitém formátu. Součástí odpovědi je i číselný HTTP status kód, který popisuje stav zpracování. V systému jsem použil kódy 200, 204, 400, 401, 404, 409 a 500. Kódy začínající číslem 2 oznamují, že se požadavek podařilo zpracovat. Kódy s počátečním číslem 4 informují, že požadavek nebyl zpracován a že je chyba na straně klienta. Poslední kód 500 informuje, že se požadavek nepodařilo zpracovat a že chyba je na straně serveru.

Endpointy jsem implementoval tak, že jsem ve frameworku definoval adresy a jejich metody. K těmto adresám jsem přiřadil funkce, kterým se má předat požadavek, který na danou adresu přijde. Funkce, které provádějí zpracování požadavků, se nacházejí v kontrolerech. Každý kontroler obsahuje funkce, které obsluhují jednu entitu databázového modelu. Funkce po zpracování požadavku zašle odpověď, která buď obsahuje data anebo chybovou hlášku. Data každého požadavku jsou kontrolována, zda jsou validní a správného typu.

Celkem jsem vytvořil 66 endpointů, pomocí kterých může klientská část komunikovat se serverovou částí. Tyto endpointy umožňují provádět CRUD nad navrženými entitami. Seznam všech

adres vytvořeného API je uveden v příloze C. V následujícím textu popíšu nejdůležitější a nejzajímavější části vytvořeného API.

**Výpis kódu 5.1** Data záznamu ve formátu JSON

```
{
    "Record": {
        "id": 1,
        "progress": 1,
        "progress_note": "Zlepšena hybnost kotníku.",
        "exercise_note": "Cvičení již není bolestivé.",
        "text": null,
        "event_id": 1,
        "created_at": "2022-04-29T11:34:44.000000Z",
        "updated_at": "2022-04-29T11:34:44.000000Z"
    }
}
```

## 5.2 Přihlášení do systému

Už před zahájením implementace bylo jasné, že v systému bude potřeba vytvořit přihlášení. Protože se celá aplikace skládá ze dvou částí, bylo potřeba najít způsob, jak na backendu ověřit, že komunikace probíhá právě s frontendem. Kdybych toto ověření nezajistil, mohl by kdokoliv zasílat požadavky na backend a získat tak citlivé data. Pro zajištění autentizace jsem využil připravený balíček Laravel Sanctum.

Sanctum je balíček, který byl vytvořen přímo pro poskytování autentizace pro SPA, mobilní aplikace a jednoduché API. [26] (překlad vlastní) Já jsem využil autentizaci API pomocí tokenů. Celé přihlášení do systému funguje následovně. Klientská část zašle na serverovou část požadavek na přihlášení na příslušnou adresu s přihlašovacími údaji uživatele. Funkce na backendu provede kontrolu těchto údajů tak, že se pokusí nalézt uživatele v databázi. Pokud jej najde, vytvoří pro něj token. Vygenerovaný token je pak vrácen v odpovědi zpět frontendu. Pokud uživatel zadá špatné přihlašovací údaje, systém vrátí odpověď se status kódem 401 a chybovou hlášku. Token, který má klientská část po přihlášení k dispozici, je potřeba vložit do hlavičky každého požadavku, který je posílán na backend. Všechny adresy, které jsem v systému definoval, a které nemají být veřejné, jsou totiž chráněny pomocí autentizačního middleware. Autentizační middleware je již nachystán v balíčku Sanctum, takže není potřeba jej konfigurovat. Každá adresa, která je tímto middlewarem chráněna, nejprve předá požadavek této službě, která zkontroluje token. Pokud je token platný, middleware vrátí požadavek zpět k následnému zpracování. Pokud token není vyplněn nebo není správný, middleware automaticky tento požadavek zablokuje. Sanctum samozřejmě umožňuje nastavit různé vlastnosti tokenu, které můžou zvýšit bezpečnost. Já jsem pro každý token určil platnost na jednu hodinu. V základním nastavení má token platnost vypnutou, takže každý vygenerovaný token je možno použít kdykoliv. Myslím si, že nekonečná platnost určitě není bezpečná. Nastavená hodinová platnost je podle mě vyhovující. Fyzioterapeut se sice bude muset během dne vícekrát přihlašovat, nicméně pracuje s citlivými údaji a zveřejnění nekonečně platného tokenu by byl bezpečnostní problém. Navíc se do systému budou přihlašovat i klienti, kteří se systémem nebudou pracovat celý den. Kromě toho někteří klienti se často zapomínají odhlašovat ze systémů, takže omezenou platností tokenu zajistím automatické odhlášení.

Sanctum kromě změny vlastností nabízí taky přiřazení schopností tokenu při vytváření. Tuto funkcionalitu jsem v systému využil k rozpoznání role přihlášeného uživatele. Definoval jsem pro uživatele role admin a client. Role admin je určena pro fyzioterapeuta. Ten to totiž potřebuje přístup do celého systému, takže může prakticky provádět veškeré operace. Zatímco role client

může v systému využívat pouze některé funkce. Při přihlášení nejprve rozpoznám uživatele a jeho roli podle přihlašovacích údajů. Následně generuji token, buď se schopností admin nebo client dle role. Autentizační middleware pak může kromě kontroly samotného tokenu kontrolovat i jeho schopnosti. Mohl jsem proto u každé adresy definovat, jaká role může na danou adresu zasílat požadavek. Sanctum se tak sám postará o omezení přístupu, které bych jinak musel kontrolovat v každém kontroleru ručně. Ve výpisu kódu 5.2 je ukázka, jak se vytváří odpověď s tokenem.

**Výpis kódu 5.2** Vytvoření odpovědi s tokenem pro roli admin

```
if(Auth::attempt($credentials)) {
    if(Auth::user()->role === 1) {
        return $this->sendData(
            [
                'Token' => Auth::user()->
                    createToken('auth_token', ['admin'])->
                    plainTextToken
            ]
        );
    }
}
```

## 5.3 Nezletilý klient

Z analýzy informací od fyzioterapeutů jsem věděl, že jejich služby často využívají děti. Při zpracování informací o dětech bylo obtížné rozhodnout, jaké údaje mohou být povinné. U dětí totiž není nikdy jisté, jestli už mají své telefonní číslo nebo e-mail. Z tohoto důvodu jsem nejprve uvažoval nad vytvořením oddělené entity, která by byla pouze pro děti. Nakonec jsem ale došel k názoru, že by entita dítě byla téměř ve všech parametrech stejná jako klient a lišila by se pouze v detailech. Rozhodl jsem se proto udělat entitu klient univerzální. Většina údajů je proto v entitě klient nepovinná, aby bylo možné do systému uložit i údaje o dítěti.

Bohužel nepovinnost údajů, konkrétně e-mailu, způsobila další problém. V případě, že u nezletilého klienta nebude vyplněn e-mail, nebude možné se přihlásit do systému a vidět zadané úkoly se cviky. To je pro dítě problém, protože zapamatovat si správnou techniku cvičení může být obtížné. Aby dítě správně rehabilitovalo, musí mít přístup do systému, aby si mohlo připomenout, jak zadané cviky provádět. Každé dítě má své zákonné zástupce, kteří budou pravděpodobně s fyzioterapeutem komunikovat, aby byli obeznámeni s průběhem léčby svého dítěte. Nabízela se proto možnost vyplnit pro přihlášení e-mail rodiče. Toto řešení je samozřejmě možné použít. Nicméně opět může nastat problém. Co když k fyzioterapeutovi dochází rodič i dítě současně? Fyzioterapeut potřebuje dvě karty klienta pro každého zvlášť. To znamená, že i přihlašovací účty musí být dva. Oba uživatelé ale nemůžou mít stejný e-mail, protože ten musí být unikátní, aby se mohl použít jako přihlašovací údaj. Tento problém se mi podařilo vyřešit vazbou entity klienta sama na sebe. Entita má v sobě nepovinný parametr client_id. Ten když je vyplněn, tak obsahuje číselný identifikátor rodiče jako klienta. Fyzioterapeut si nejdříve v systému uloží údaje o rodiči, včetně e-mailu. Následně si může vytvořit nového klienta s údaji o dítěti bez e-mailu, u kterého vybere zákonného zástupce ze seznamu již existujících klientů. Tím dojde k propojení klientů do jednoho uživatelského účtu. Rodič se tak může přihlásit do systému pomocí svého e-mailu a kromě své léčby uvidí i léčbu svého dítěte pod jedním účtem. Zároveň toto propojení uvidí i fyzioterapeut, takže když bude potřebovat kontaktovat rodiče, najde informace o rodiči v kartě klienta dítěte.

Výsledné řešení je podle mého názoru ideální z hlediska datové struktury a jednoduchosti pro rodiče. Pro fyzioterapeuta může toto řešení být komplikovanější, protože musí založit kartu klienta rodiči, který ale nemusí fyzioterapeuta vůbec navštěvovat.

## 5.4 Kontrola rodného čísla

Někteří soukromí fyzioterapeuti můžou spolupracovat s pojišťovnami stejně jako fyzioterapeuti ve zdravotnických zařízeních. Tato spolupráce obvykle zajišťuje proplacení léčebného procesu klientovi jeho pojišťovnou. V praxi to funguje tak, že klient dostane žádanku k fyzioterapeutovi od svého lékaře. Fyzioterapeut klienta s žádankou přijme a nepožaduje po klientovi uhrazení léčby. Klient ale musí fyzioterapeutovi poskytnout rodné číslo, aby mohl fyzioterapeut požadovat úhradu léčby po pojišťovně. Z tohoto důvodu umožňuji fyzioterapeutovi u každého klienta uložit rodné číslo a číslo pojišťovny.

**Výpis kódu 5.3** Funkce pro validaci rodného čísla

```
public static function verifyPIN(String $pin): bool
{
    if(!preg_match('#^\s*(\d\d)(\d\d)(\d\d)(\d\d\d)(\d?)\s*$#',
        $pin, $matches)) {
        return false;
    }
    list(, $year, $month, $day, $ext, $c) = $matches;
    if($c === '') {
        return $year < 54;
    }
    $mod = ($year.$month.$day.$ext) % 11;
    if($mod === 10) {
        $mod = 0;
    }
    if($mod !== (int)$c) {
        return false;
    }
    $year += $year < 54 ? 2000 : 1900;

    if($month > 70 && $year > 2003) {
        $month -= 70;
    }
    else if($month > 50) {
        $month -= 50;
    }
    else if($month > 20 && $year > 2003) {
        $month -= 20;
    }
    if(!checkdate($month, $day, $year)) {
        return false;
    }
    return true;
}
```

Rodné číslo je unikátní identifikátor každého člověka, takže má přesně definovanou strukturu. Aby si fyzioterapeut byl jistý, že mu klient sdělí správně své rodné číslo, bylo potřeba v systému vytvořit validaci. Každý stát má ale strukturu rodného čísla jinou, takže jsem implementoval validaci pouze pro rodné čísla z České republiky. Struktura rodného čísla je definována tak, že prvních šest číslic je datum narození v pořadí rok, měsíc, den. Další čtyři číslice se používají k ověření. Pokud jsou následující číslice po datu narození pouze tři, znamená to, že rodné číslo bylo přiděleno před rokem 1954. Takové rodné číslo nelze validovat. Klasické rodné čísla, jaké se používají dnes, se kontrolují následovně. Musí se spočítat zbytek po dělení prvních devíti číslic číslem 11. Pokud je zbytek 10, musí být poslední číslice rodného čísla rovna 0. Pokud je zbytek jiný, musí být poslední číslice rovna tomuto zbytku po dělení. Jestliže rodné číslo splňuje kontrolu,

pak může být validní. Rodné číslo má ještě jednu specifikaci, která se projevuje pouze u žen. K měsíci v datu narození se totiž přičítá číslo 50. Od roku 2003 se navíc může k měsíci přičítat i číslo 20 nebo 70. Je proto potřeba kontrolovat i číslo měsíce v datu narození po odečtení jedné z konstant, jestli je v intervalu 1-12. Teprve po ověření všech podmínek je rodné číslo validní. Výpis kódu 5.3 ukazuje mou implementaci ověření rodného čísla.

## 5.5 Filtrování událostí

Jedním z nejdůležitějších modulů v systému je kalendář. V něm bude fyzioterapeut pracovat velmi často a tak je nutné, aby práce s kalendářem byla co možná nejjednodušší. Klientská část bude fyzioterapeutovi nabízet různé způsoby zobrazení kalendáře. Základní zobrazení kalendáře bude týdenní. Fyzioterapeut si bude ale moct přepnout na měsíční nebo denní zobrazení. Pro tento způsob přepínání není vhodné, aby klientská část obdržela od backendu seznam všech událostí, který by pak musela filtrovat podle zvoleného zobrazení. Filtrování na straně frontendu by způsobilo zpomalení odezvy aplikace. Zároveň by se systém navíc ještě zpomalil i přenosem zbytečně velkého množství dat, které v tu chvíli frontend ani nepotřebuje.

Na straně backendu jsem vytvořil endpoint, který přijímá dva query parametry. Prvním je parametr datetime, který určuje jaké datum chce fyzioterapeut zobrazit. Druhým parametrem je period, který může nabývat třech hodnot: day, week, month. Perioda určuje, jak velký úsek chce vzhledem k zadanému datu fyzioterapeut zobrazit. V případě, že by klientská část nepoužila ani jeden parametr, použije se aktuální datum a denní zobrazení pro periodu. Filtrovací funkce pracuje tak, že obdrží oba query parametry. Následně podle periody zvolí způsob získání dat z databáze. Pro denní periodu dojde k porovnání začátku a konce události vzhledem k celému dni, který je určen parametrem datetime. U týdenní periody se nejprve získá datum pro pondělí a neděli z data, které přijde z frontendu. Pak dojde opět k porovnání času události s daty mezi pondělím a nedělí. Měsíční perioda provádí stejné porovnání jako týdenní, akorát místo pondělí a neděle se použije první a poslední den v měsíci. Všechny události, které spadají do meze filtru jsou odeslány v kolekci klientské části. Výpis kódu 5.4 obsahuje část funkce pro filtrování událostí.

**Výpis kódu 5.4** Část funkce pro týdenní filtrování událostí

```
$events = Collection::empty();
$monday = date("Y-m-d",
    strtotime('monday this week',
    strtotime($dateTime)));
$sunday = date("Y-m-d",
    strtotime('sunday this week',
    strtotime($dateTime)));
$events = self::all()->
whereBetween('start',
    [$monday.' 00:00:00',
        $sunday.' 23:59:59'])->
whereBetween('end',
    [$monday.' 00:00:00',
        $sunday.' 23:59:59']);
return $events;
```

## 5.6 Protokoly událostí

V návrhu systému jsem analyzoval obecné nařízení o ochraně osobních údajů, které požaduje sbírat informace o přístupu k citlivým údajům v systému. Zároveň nařízení vyžaduje zálohování osobních údajů, aby bylo možné systém obnovit v případě výpadku. Z těchto důvodů jsem do systému implementoval logování.

Laravel má již v základním nastavení po instalaci logování zapnuto. Toto logování ale ukládá pouze informace o chybách, které se v systému staly. Pro logování přístupu a změny dat bylo potřeba definovat vlastní informační kanály. Laravel umožňuje v konfiguračním souboru provést kompletní nastavení. Já jsem v systému vytvořil tři logovací kanály. Všechny mnou vytvořené logy mají nastavenou důležitost info, protože budou zapisovat pouze informace o tom, jaké akce v systému proběhly. Laravel nabízí sedm dalších úrovní důležitosti, takže je možné logovat různé typy událostí a akcí v systému. První log, který jsem nastavil, je recepce (reception). Tento log bude ukládat informace o přihlášení do systému, odhlášení a změně přihlašovacích údajů. Další log je záznam (record), který bude sbírat informace o tom, jak byly zdravotní záznamy upravovány. Poslední log klient (client) bude zaznamenávat změny v osobních údajích v kartě klienta. Všechny záznamy v každém logu obsahují časový údaj a identifikátor autora, který akci v systému provedl. Zápis do logu se provádí tak, že v kontroleru na místě, kde se provádí určitá akce, která má být zalogována, provedu volání logovací komponenty v Laravelu, které určím, kam a co má zalogovat. Mnou vytvořené logy splňují požadavek směrnice na logování změn a přístupu k osobním údajům. Zároveň logy vytváří zálohu pro obnovu systému. V dnešní době totiž extuje mnoho nástrojů, které dokážou s logy pracovat tak, že jsou schopny vyjmout data a připravit je pro databázový import. Následující výpis 5.5 zobrazuje konfiguraci logovacích kanálů v systému. Výpis 5.6 obsahuje část výpisu souboru reception.log.

**Výpis kódu 5.5** Konfigurace logovacích kanálů

```
'reception' => [
    'driver' => 'single',
    'path' => storage_path('logs/reception.log'),
    'level' => 'info',
],
'record' => [
    'driver' => 'single',
    'path' => storage_path('logs/record.log'),
    'level' => 'info',
],
'client' => [
    'driver' => 'single',
    'path' => storage_path('logs/client.log'),
    'level' => 'info',
]
```

**Výpis kódu 5.6** Výpis souboru reception.log

```
[2022-04-27 20:49:42] local.INFO: Logging admin user {"user_id":1}
[2022-04-27 20:50:19] local.INFO: Updated user login credentials.
    {"Params":{"email":"admin@test.com"}}
[2022-04-27 20:51:48] local.INFO: Logging out user.
    {"User":{"App\\Models\\User":
        {"id":1,"email":"admin@test.com","role":1,
        "email_verified_at":"2022-04-23T19:16:16.000000Z",
        "staff_id":1,"client_id":null,
        "created_at":null,"updated_at":"2022-04-27T20:50:19.000000Z"}}}
```

## 5.7 Vícejazyčnost

Na začátku implementace jsem se rozhodl, že systém bude v anglickém jazyce. Jednalo se především o chybové a kontrolní zprávy systému pro frontend. Tyto zprávy se zobrazují fyzioterapeutovi nebo klientovi v případě, že se nepovedlo vykonat požadovanou akci. Po konzultaci s vedoucím práce jsem došel k závěru, že systém je určen pro české klienty a fyzioterapeuty, takže by chybové hlášky měly být i v češtině, aby jim uživatel rozuměl. Protože je angličtina světovým jazykem, bylo vhodné zachovat hlášky i v angličtině. Nebylo tedy jiné řešení, než zavést v systému vícejazyčný překlad. Tato funkcionalita nebyla původně v návrhu a je tedy implementována nad rámec zadání práce.

Laravel obsahuje obecný konfigurační soubor, ve kterém lze určit jazykovou lokalizaci aplikace. Podle lokalizace je následně zvolen soubor s definovanými překlady. Problém nastal, když bylo potřeba změnit lokalizaci za běhu systému. V konfiguračním souboru se totiž lokalizace nastavuje přímo a nelze ji tak měnit v závislosti na frontendu. Musel jsem proto implementovat vlastní middleware pro lokalizaci. Tento middleware jsem následně nastavil všem adresám v systému. Lokalizační middleware funguje stejně jako autentizační middleware ze sekce 5.2. Každý příchozí požadavek nejprve vstoupí do lokalizační služby, která se pokusí v hlavičce najít parametr X-localization. Pokud je tento parametr vyplněn, middleware nastaví lokalizaci aplikace na hodnotu parametru. Hodnota může nabývat dvou zkratek en (anglický jazyk) a cs (český jazyk). V případě, že parametr není vyplněn nebo je hodnota jiná než výše uvedené, použije se angličtina. Po nastavení lokalizace služba vrátí požadavek zpět ke zpracování. V případě, že se požadavek neprovede, vrací se frontendu odpověď s chybovou zprávou. Aby byla chybová zpráva správně přeložena, musel jsem použít překládací funkci Laravelu. Ta zajistí, že podle lokalizace použije správný soubor s přeloženými hláškami. Každá hláška je v souborech uložena pod stejnou proměnnou. Překládací funkci stačí předat název proměnné, podle které se překlad vyhledá. Soubory pro anglický a český překlad jsem musel vytvořit a všechny hlášky v nich definovat. Protože systém nyní využívá překládací funkci, bude v budoucnosti jednoduché do systému přidat další jazykové překlady. Bude pouze potřeba vytvořit kopii existujícího souboru s hláškami a přeložit je do požadovaného jazyka. Níže ve výpisu 5.7 uvádím funkci middlewaru zajišťující nastavení lokalizace. Výpis 5.8 pak demonstruje použití překládací funkce v odpovědi.

**Výpis kódu 5.7** Funkce lokalizačního middleware

```
public function handle(Request $request, Closure $next): mixed
{
    $local = ($request->hasHeader('X-localization')) ?
        $request->header('X-localization') : 'en';
    app()->setLocale($local);
    return $next($request);
}
```

**Výpis kódu 5.8** Překlad chybové hlášky v odpovědi

```
public function sendNotFound(string $message): Response
{
    return response(['message' => trans($message)], 404);
}
```

## 5.8 Testování

Zadání práce požadovalo otestovat systém vhodnými automatizovanými testy. I kdyby ale zadání testování systému nepožadovalo, stejně bych musel systém před uvedením do reálného provozu otestovat. V dnešní době se klade velký důraz na testování aplikací, protože každý uživatel je

jiný a může se systémem pracovat nestandardně. V takovém případě by měl systém upozornit uživatele na špatné používání a ukázat chybu, které se uživatel v systému dopustil. Někteří vývojáři testování podceňují a vystavují se tak riziku, že aplikace během používání selže. Abych se tomuto problému vyvaroval, vytvořil jsem sadu testů, které ověřují správné chování systému.

Během vývoje jsem používal pro simulaci frontendu aplikaci Postman. Tato aplikace umožňuje zasílat požadavky na zadanou adresu a zobrazovat odpovědi ze serveru. V aplikaci lze také definovat strukturu backendového API. Podle struktury pak lze vytvořit automatické testy, které ověří správnost odpovědi. Možnost vytvořit testy v aplikaci Postman jsem nakonec nevyužil. Problém totiž nastal při sdílení testů. Abych mohl testy provést i na jiném počítači, musel bych testy z aplikace vyexportovat a pomocí repozitáře je nasdílet. Na druhém počítači bych pak musel aplikaci Postman nainstalovat a testy importovat zpět. Toto řešení se mi zdálo komplikované a pro možný budoucí týmový vývoj nevhodné. Z tohoto důvodu jsem vytvořil testy přímo v systému. Testy se tak sdílí současně s celou aplikací a pro jejich spuštění není potřeba další software. Laravel obsahuje testovací knihovnu PHPUnit, která umožňuje testovat PHP aplikace. Pomocí PHPUnit jsem vytvořil jednotkové (unit) testy, které mají za úkol otestovat jednotlivé funkce v dané třídě. V mém případě se jedná o funkce, které se nacházejí uvnitř kontrolerů a provádí zpracování dat. Testy jsem rozdělil do souborů podle kontrolerů. Každý test v souboru pak testuje určitou funkci v daném kontroleru. Testování probíhá tak, že se provede volání příslušné adresy s testovacími daty. Následně se přijme odpověď, která je kontrolována na správnost a úplnost dat. V případě, že kontrola odpovědi selže, je test vyhodnocen jako neúspěšný. Většina adres v systému používá autentizační middleware pro kontrolu tokenu v hlavičce požadavku. Abych v každém testu nemusel provádět přihlášení pro získání tokenu, provedl jsem u některých testů vypnutí autentizační služby. Odpojení middleware pro některé testy jsem udělal i kvůli fylosofii jednotkových testů, které mají testovat pouze požadovanou funkcionalitu zpracování a nemají se zabývat testováním celkového chování systému, které samozřejmě obsahuje zablokování požadavku nepřihlášenému uživateli. Výpis 5.9 demonstruje způsob testování systému.

**Výpis kódu 5.9** Část funkce pro testování vytvoření události

```
$response = $this->postJson('/api/event', [
    'name' => 'Testovací␣událost',
    'start' => '2022-04-30␣15:00:00',
    'end' => '2022-04-30␣16:00:00',
    'note' => null,
    'event_type_id' => 1,
    'client_id' => 2,
    'staff_id' => 1
]);
$response->assertStatus(200)->
    assertJsonPath('Event.name', 'Testovací␣událost')->
    assertJsonPath('Event.start', [
        'date' => '2022-04-30␣15:00:00.000000',
        'timezone_type' => 3,
        'timezone' => 'UTC'
    ])->
    assertJsonPath('Event.end',  [
        'date' => '2022-04-30␣16:00:00.000000',
        'timezone_type' => 3,
        'timezone' => 'UTC'
    ])->
    assertJsonPath('Event.note', null)->
    assertJsonPath('Event.event_type_id', 1)->
    assertJsonPath('Event.client_id', 2)->
    assertJsonPath('Event.staff_id', 1);
```

# Kapitola 6

# Nasazení a rozvoj systému

V poslední kapitole této práce popíšu, jakým způsobem jsem provedl nasazení systému na server a jak jsem propojil backend s frontendem. V závěru uvedu nedokončené části systému a návrhy na vylepšení a rozšíření systému v budoucnu.

## 6.1 Nasazení systému

Po dokončení implementace prototypu jsem se rozhodl nasadit systém na server, abych jej mohl následně propojit s klientskou části kolegy Michala. Pro nasazení jsem si pronajal virtuální server (VPS) u společnosti WEDOS Internet, a. s. Kromě serveru jsem musel také zakoupit doménu, přes kterou se bude do systému přistupovat. Protože jsem doménu a server zařizoval ještě před tím, než jsem začal pracovat na bakalářské práci, má doména jiné jméno než má aplikace. Z tohoto důvodu je backend pro testovací účely dostupný na adrese https://api.jakubvolak.tech/api.

Na serveru jsem musel pro spuštění systému nainstalovat několik knihoven a aplikací. Pro běh systému bylo potřeba nainstalovat jazyk PHP v nejnovější verzi 8.1. Dále jsem na server nainstaloval databázový server MySQL, webový server NGINX a orchestrační nástroj Composer, který používá Laravel pro instalaci dalších knihoven a závislostí. Po instalaci potřebného softwaru jsem provedl přesun systému na server. Následně jsem musel nakonfigurovat webový server, aby byl schopen přijímat požadavky a spustit mou aplikaci. V konfiguraci jsem musel nastavit název domény a umístění souboru index.php na serveru, který je vstupním bodem aplikace. Serveru NGINX jsem také musel nastavit používanou verzi jazyka PHP. Po úspěšném spuštění jsem ještě musel zabezpečit doménu, abych vyhověl požadavkům GDPR na šifrovanou komunikaci. Pro zabezpečení své domény jsem použil aplikaci certbot [27], která je zdarma. Tato aplikace provede ověření serveru a následně vygeneruje bezpečnostní certifikát, který zaručí zabezpečení domény. Díky certifikátu je komunikace zajištěna šifrovaným protokolem HTTPS. Nasazení na server proběhlo rychle a bez komplikací tak, jak jsem očekával.

Propojení s klientskou částí bylo jednoduché. Kolegovi Michalovi jsem pouze sdělil adresu, na které je dostupná serverová část. Michal následně při nasazení své klientské části změnil základní adresu backendu v konfiguraci frontendu. Tím došlo k propojení obou částí. Nasazení a spuštění klientské části se věnuje Michal ve své práci.

## 6.2 Rozvoj systému

Aplikaci, kterou jsem s kolegou Michalem vytvořil, bych nyní považoval za prototyp. Ten bude potřeba v blízké budoucnosti upravit a vylepšit, aby aplikace mohla být použita v reálném

provozu. Abychom věděli, jak systém rozšířit, budeme muset aplikaci nechat otestovat fyzioterapeuty. Ti nám sice na začátku poskytli informace o své pracovní náplni, nicméně jenom oni poznají, jestli jim námi vytvořená aplikace vyhovuje. Budeme od nich sbírat zpětnou vazbu a systém podle nových poznatků přizpůsobovat tak, aby se fyzioterapeutům co možná nejlépe používal. Zároveň budeme chtít sbírat podněty na vylepšení i od klientů, pro které musí být aplikace také jednoduchá a srozumitelná. Následující text popíše nedostatky vytvořeného prototypu.

Systém aktuálně při vytváření karty klienta také vytváří uživatelský účet pro přístup do systému. Před uložením přihlašovacích údajů do databáze je systémem vygenerováno náhodné heslo. Toto heslo je následně odesláno na e-mail klienta, aby se mohl do systému přihlásit. Informační e-mail pro klienta vytváří a odesílá backend. Odeslání e-mailu sice funguje, ale šablona e-mailu neobsahuje styl. Obsah e-mailu je tedy čistý černý text na bílém pozadí. Vzhled e-mailu nijak neosloví klienta, aby začal aplikaci používat. To samozřejmě není dobrá vizitka systému, který by měl léčbu klientům zpříjemnit a urychlit. Před ostrým nasazením systému bude určitě potřeba vytvořit styl pro e-mailové šablony tak, aby e-maily působily věrohodně a poutavě. Vzhled e-mailu by měl být určitě stejný jako vzhled frontendu aplikace.

Aplikace obsahuje data o událostech, které si fyzioterapeut naplánuje. Události mají už nyní v systému svůj typ. Tyto informace by se mohly v budoucnu využít pro vytváření statistik. Fyzioterapeut v systému aktuálně nemá nástěnku, která by poskytovala informace například o odpracovaných hodinách nebo počtu odbavených klientů za určité časové období. Všechny potřebné data pro vytvoření statistik v systému již sbíráme, nicméně nemáme vytvořené algoritmy, které by počítaly statistické údaje. Myslím si, že informace vypočítané systém na základě vyplněných dat, by mohly být pro fyzioterapeuta užitečné. Volbu, jaké statistiky počítat, bychom samozřejmě nechali na fyzioterapeutech. Ti totiž sami nejlépe vědí, jaké informace by pro ně mohly být relevantní a přínosné.

Každý klient může v aplikaci vyplnit zpětnou vazbu k zadanému cvičení. Hodnocení se nyní skládá z textové odpovědi a číselné hodnoty složitosti cviku. Protože se zatím jedná o prototyp, myslím si, že taková forma zpětné vazby je dostačující. Nicméně si myslím, že až bude aplikace v provozu, bude nutné zpětnou vazbu vylepšit. Bude potřeba sledovat, jak moc klienti zpětnou vazbu vyplňují a jak moc je relevantní pro fyzioterapeuta. Podle zjištěných poznatků by se hodnocení zadaných cviků mohlo například rozložit do více specifických možností. Zároveň by aplikace mohla v budoucnu klienta upomínat, aby vyplnil zpětnou vazbu. Tím by aplikace mohla klienta donutit k provedení cviků. Upomínky by aplikace nemusela posílat jenom k vyplnění zpětné vazby, ale i k blížícímu se termínu schůzky s fyzioterapeutem.

Kromě vylepšení dílčích funkcionalit systému bude potřeba se zaměřit i na další vývoj modulu encyklopedie léčby. Ten je v systému aktuálně vyřešen pomocí štítků, které se přiřazují jednotlivým klientům. Štítky si v systému může fyzioterapeut definovat sám pro každý typ diagnózy. Následně fyzioterapeut může v systému vyhledávat klienty podle štítku, aby si mohl zobrazit historii léčby. Zobrazení předchozího způsobu léčby by fyzioterapeutovi mělo pomoct při výběru vhodného léčebného procesu u nových klientů. Tohle řešení jsme navrhli s kolegou Michalem, bohužel každý fyzioterapeut jej podle získaných informací pochopil trochu jinak. Myslím si proto, že aktuální jednoduché řešení by mělo být lépe představeno fyzioterapeutům. Ti by pak měli určit, jak encyklopedii léčby lépe zpracovat, aby poskytovala lepší a odbornější informace pro stanovení diagnózy a její léčby.

Prototyp systému jsme s kolegou Michalem vytvořili co nejlépe podle analýzy získaných informací od fyzioterapeutů. Aplikace je funkční a lze ji používat. Obsahuje ale určité části, které by se měly před ostrým nasazením odladit a vylepšit podle zpětné vazby fyzioterapeutů a klientů. Aplikace by se proto měla v blízké budoucnosti nasadit do testovacího provozu několika fyzioterapeutům, aby bylo možné pokračovat ve vylepšování funkcionalit.

# Závěr

Cílem této bakalářské práce bylo navrhnout a vytvořit prototyp serverové části webového informačního systému pro fyzioterapeuty. Systém byl vytvořen na základě požadavků získaných od fyzioterapeutů. Serverová část byla řádně otestována a propojena s klientskou částí kolegy Michala Kováře. Kromě samotného vývoje byly také zpracovány návrhy pro rozvoj systému. Všechny cíle jsem splnil a dodržel tak zadání práce.

Implementace serverové části je postavena na moderních technologiích a principech, tím je mimo jiné zajištěna snadná udržitelnost. Zároveň je možné díky využití frameworku systém jednoduše rozšiřovat o nové funkcionality. V návrhu systému jsem se zabýval bezpečností tak, aby serverová část splňovala směrnici o ochraně osobních údajů. Během vývoje jsem se potýkal s několika implementačními problémy, nicméně všechny se mi podařilo vyřešit. Oproti zadání jsem navíc v systému implementoval podporu vícejazyčnosti. Aktuálně systém lze používat v českém a anglickém jazyce a je připraven na doplnění dalších jazyků.

Aplikace nyní umožňuje fyzioterapeutovi kompletně spravovat kartu klienta, včetně souborů obsahující lékařské zprávy. Fyzioterapeut si v aplikaci může plánovat schůzky se svými klienty a zapisovat si průběh léčby. V aplikaci si může fyzioterapeut vytvořit vlastní databázi cviků. Cviky pak může přiřazovat klientům, aby je mohli cvičit doma a urychlit tak léčbu. Ke každému cviku může fyzioterapeut nahrát multimediální obsah. Systém umožňuje fyzioterapeutovi definovat štítky reprezentující určité diagnózy. Podle štítků může fyzioterapeut v systému vyhledávat klienty, aby mohl nahlédnout do historie léčby. Klient v systému nalezne přehled o léčbě, zadané úkoly se cviky a informace o následujících schůzkách. Klientovi je také umožněno vyplnit zpětnou vazbu k zadaným cvikům.

Výsledný prototyp systému je funkční a je možné jej nasadit do testovacího provozu, který přinese nové poznatky pro budoucí rozšíření systému.

Příloha A

# Dotazník pro fyzioterapeuty

# Informační systém pro fyzioterapii

Dobrý den,
jmenuji se Michal Kovář, jsem studentem Fakulty informačních technologií na ČVUT v Praze a společně se spolužákem Jakubem Volákem pracujeme na bakalářské práci, jejíž cílem je navrhnout a realizovat Informační systém pro fyzioterapii.

V první části tohoto dotazníku Vám představíme naší vizi - jak bychom si takový systém představovali a žádáme po Vás zpětnou vazbu, která nám naši vizi potvrdí, případně nás nasměruje tím správným směrem.

Ve druhé části bychom rádi zmapovali Vaše procesy a získali přehled o tom, jak vypadá Váš pracovní den. To nám pomůže analyzovat proces fyzioterapeutické činnosti a následně v našem systému tyto činnosti vhodně optimalizovat.

Nebojte se prosím chválit a hlavně kritizovat, jedině to nám pomůže určit co je a co není vhodné do systému zavést. Věřím, že se nám společně podaří vybudovat ideální řešení na míru, které úspěšně uvedeme do provozu.

---

* Required

1. Jméno a příjmení

_____

2. E-mail

_____

3. Co nejlépe vystihuje Vaši činnost? *

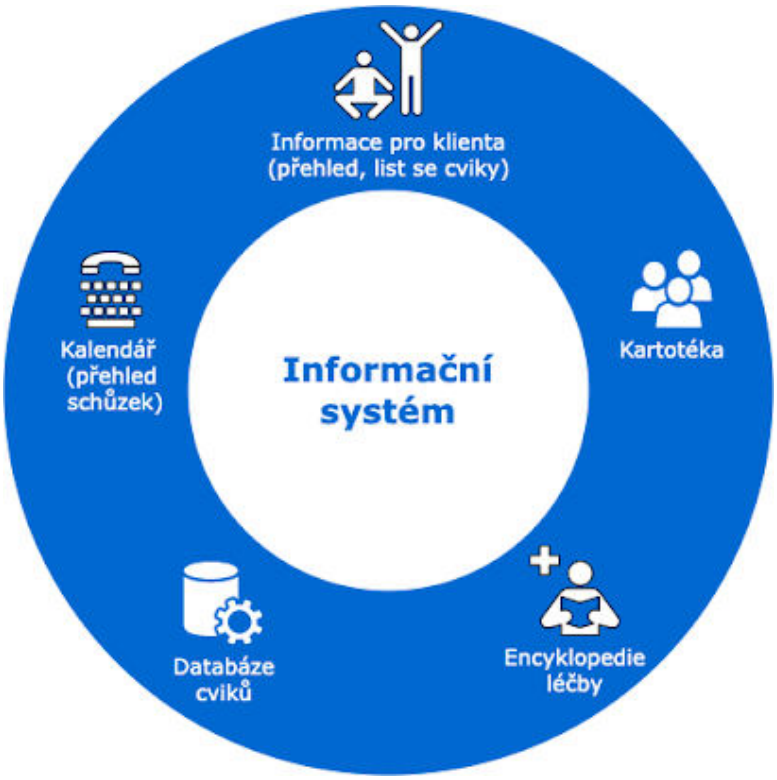
*Mark only one oval.*

- ◯ Fyzioterapeut
- ◯ Sportovní trenér/trenérka
- ◯ Other: _____

## Představení naší vize

Nyní Vám popíšeme jednotlivé části informačního systému. U každé z částí zaškrtněte relevantnost našeho návrhu, případně připojte doplňující komentář.

Grafické znázornění klíčových částí informačního systému.




### Kalendář

Modul, který spravuje fyzioterapeutův časový harmonogram. Umožňuje vytvářet, upravovat a mazat schůzky s klientem, případně jiné aktivity, kterou si fyzioterapeut definuje.

4. Kalendář považuji za relevantní a při práci ho využiji. *

*Mark only one oval.*

| | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| Rozhodně ano | ◯ | ◯ | ◯ | ◯ | ◯ | Rozhodně ne |

5. Na jak dlouhý časový blok obvykle plánujete schůzky? *

(např. 30 minut, 1 hodina, ...)

6. Necháváte si časovou rezervu mezi schůzkami? Pokud ano, jak velkou? *

7. Máte stanovenou fixní pracovní dobu? *

*Mark only one oval.*

- Ano
- Ne

8. Doplňující komentář ke kalendáři

## Databáze cviků

Modul, umožňující ukládat vlastní cviky ve formě videí, obrázků či textového popisu. Cviky se následně posílají klientovi formou listu úkolů, jež slouží jako průvodce cvičením a zároveň jako kontrolní mechanismus fyzioterapeuta, zda klient aktiv pracuje na zlepšení svého zdravotního stavu.

9. Databázi cviků považuji za relevantní a při práci ji využiji. *

*Mark only one oval.*

| | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| Rozhodně ano | ◯ | ◯ | ◯ | ◯ | ◯ | Rozhodně ne |

10. Jakou formou byste nejčastěji ukládal/a cviky do databáze?

*Check all that apply.*

- [ ] Textový popis
- [ ] Obrázky
- [ ] Videa
- [ ] Kombinace výše uvedených

11. Využil/a byste možnost rozdělení cviků do kategorií? *

*Mark only one oval.*

- ◯ Ano
- ◯ Ne

12. Máte nějaký zdroj odkud cviky vybíráte? Pokud ano, jaký? *

Můžete vložit i odkazy na webové stránky.

_____

13. Doplňující komentář k databázi cviků

## Encyklopedie léčby

Fyzioterapeut si může klienty označovat pomocí štítků*, které mohou symbolizovat diagnózu, způsob léčby či jiná relevant data. Modul "encyklopedie léčby" následně umožňuje mezi těmito štítky vyhledávat a filtrovat, a pomoci tak fyzioterapeut efektivně určovat diagnózy a léčby na základě historických dat.

*Pojmem "štítek" se označuje slovo nebo slovní spojení, přiřazené například určité diagnóze, které co nejpřesněji takovou diagnózu vystihuje. Lze pomocí tohoto štítku pak vyhledávat obsah se stejnou diagnózou.

14. Encyklopedii léčby považuji za relevantní a při práci ji využiji. *

*Mark only one oval.*

| | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| Rozhodně ano | ◯ | ◯ | ◯ | ◯ | ◯ | Rozhodně ne |

15. Podle jakých kritérií byste nejčastěji vyhledával/a v encyklopedii?

*Check all that apply.*

- [ ] Podle diagnózy
- [ ] Podle způsobu léčby
- [ ] Podle údajů klienta (např. jméno, věk, ...)
- [ ] Podle typu klienta (např. sportovec, důchodce, dítě, ...)
- [ ] Other: _____

16. Doplňující komentář k encyklopedii léčby

## Kartotéka

Modul umožňuje spravovat karty klientů, včetně ukládání citlivých informací potřebných k léčbě jako např. zdravotní zpráv zdravotní stav apod. Zároveň si zde fyzioterapeut vede záznamy o proběhlých schůzkách, průběhů léčby a aktuálním zdravotním stavu klienta.

17. Kartotéku považuji za relevantní a při práci ji využiji. *

*Mark only one oval.*

| | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| Rozhodně ano | ◯ | ◯ | ◯ | ◯ | ◯ | Rozhodně ne |

18. Jak byste evidoval/a lékařské zprávy klientů?

*Check all that apply.*

- [ ] Přepsal/a bych zprávu do systému
- [ ] Naskenoval/a bych zprávu
- [ ] Vyfotil/a bych ji přes telefon
- [ ] Other:

19. Doplňující komentář ke kartotéce

## Informace pro klienta

Tento modul je přístupný klientovi, který zde má přehled o léčbě, záznamy schůzek uplynulých, připomínky schůzek budoucích a list cviků, které má cvičit do příští návštěvy fyzioterapeuta. Modul zároveň klientovi umožňuje posílat zpětnou vazbu při průběhu cvičení. To fyzioterapeutovi umožní lépe posoudit aktuální zdravotní stav klienta a přizpůsobit léčbu.

20. Modul "Informace pro klienta" považuji za relevantní a při práci ho využiji. *

*Mark only one oval.*

| | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| Rozhodně ano | ◯ | ◯ | ◯ | ◯ | ◯ | Rozhodně ne |

21. Jaké informace byste chtěl/a zpřístupnit klientovi? *

*Check all that apply.*

- [ ] Záznam schůzek
- [ ] Stav léčby
- [ ] Cviky, ale jen ty, které má na listu úkolů
- [ ] Celou databázi cviků
- [ ] Termín další schůzky
- [ ] Other: _____

22. Jaké informace byste NEchtěl/a zpřístupnit klientovi?

23. Jakým způsobem nejčastěji komunikujete s klientem? *

*Check all that apply.*

- [ ] Osobně
- [ ] Telefonicky
- [ ] SMS
- [ ] E-mailem
- [ ] Other: _____

24. Doplňující komentář k modulu "Informace pro klienta"

## Mapování klíčových procesů

Nyní Vám položíme sadu otázek týkající se procesů, které definují průběh Vašeho pracovního dne. U uzavřených otázek zaškrtněte vyhovující políčka a pokud Vám nevyhovuje žádná z odpovědí, uveďte podrobnosti v položce "Jiné..."

25. Jak často Vás navštěvují tyto věkové kategorie? *

*Check all that apply.*

| | Vůbec | Občas | Hodně |
|---|---|---|---|
| Děti (do 12 let) | ☐ | ☐ | ☐ |
| Mladiství (12 - 18 let) | ☐ | ☐ | ☐ |
| Dospělí v produktivním věku (18 - 65 let) | ☐ | ☐ | ☐ |
| Důchodci (65+ let) | ☐ | ☐ | ☐ |

26. Jak často Vás navštěvují tyto typy lidí? *

*Check all that apply.*

| | Vůbec | Občas | Hodně |
|---|---|---|---|
| Sportovci (vrcholoví či velmi aktivní) | ☐ | ☐ | ☐ |
| Lidé s omezenou schopností pohybu | ☐ | ☐ | ☐ |
| Lidé s obezitou | ☐ | ☐ | ☐ |
| Lidé po zranění či operaci | ☐ | ☐ | ☐ |
| Lidé se "sedavým zaměstnáním" | ☐ | ☐ | ☐ |

27. Jakým způsobem si vedete kalendář schůzek? *

*Check all that apply.*

- ☐ V papírovém diáři či stolním kalendáři
- ☐ V mobilní/webové aplikaci (Google Calendar, Apple Calendar, ...)
- ☐ V Excelu
- ☐ Other: _____

28. Jaké informace o klientovi potřebujete mít v kalendáři u plánované schůzky? *

*Check all that apply.*

- [ ] Jméno a příjmení
- [ ] Datum a čas schůzky
- [ ] Zdravotní stav
- [ ] Other: _____

29. Jak si uchováváte informace o klientech? *

(např. osobní údaje, postup léčby, ...)

*Check all that apply.*

- [ ] V papírové formě
- [ ] V mobilní/webové aplikaci
- [ ] V excelu
- [ ] Other: _____

30. Jaké informace o klientech si ukládáte při první návštěvě? *

*Check all that apply.*

- [ ] Jméno a příjmení
- [ ] Pohlaví
- [ ] Datum narození či věk
- [ ] Rodné číslo
- [ ] Číslo pojišťovny
- [ ] Váha
- [ ] Výška
- [ ] E-mail
- [ ] Telefonní číslo
- [ ] Adresu bydliště
- [ ] Lékařské zprávy
- [ ] Prodělané nemoci
- [ ] Prodělané úrazy
- [ ] Sportovní aktivita
- [ ] Other: _____

31. Jaké informace o klientech si zapisujete při běžné další návštěvě?

32. Jaké statistiky o klientech si vytváříte v rámci libovolného časového období (např. týden, měsíc, rok)? *

*Check all that apply.*

- [ ] Počet nových klientů
- [ ] Celkový počet stávajících klientů
- [ ] Počet vyléčených/propuštěných klientů
- [ ] Počet odpracovaných hodin
- [ ] Other: _____

33. Popište, jak probíhá úvodní schůzka s klientem.

34. Popište, jak probíhá běžná schůzka s klientem, případně v čem se liší od schůzky úvodní.

35. Jak probíhá zadávání a kontrola domácího cvičení klientů? *

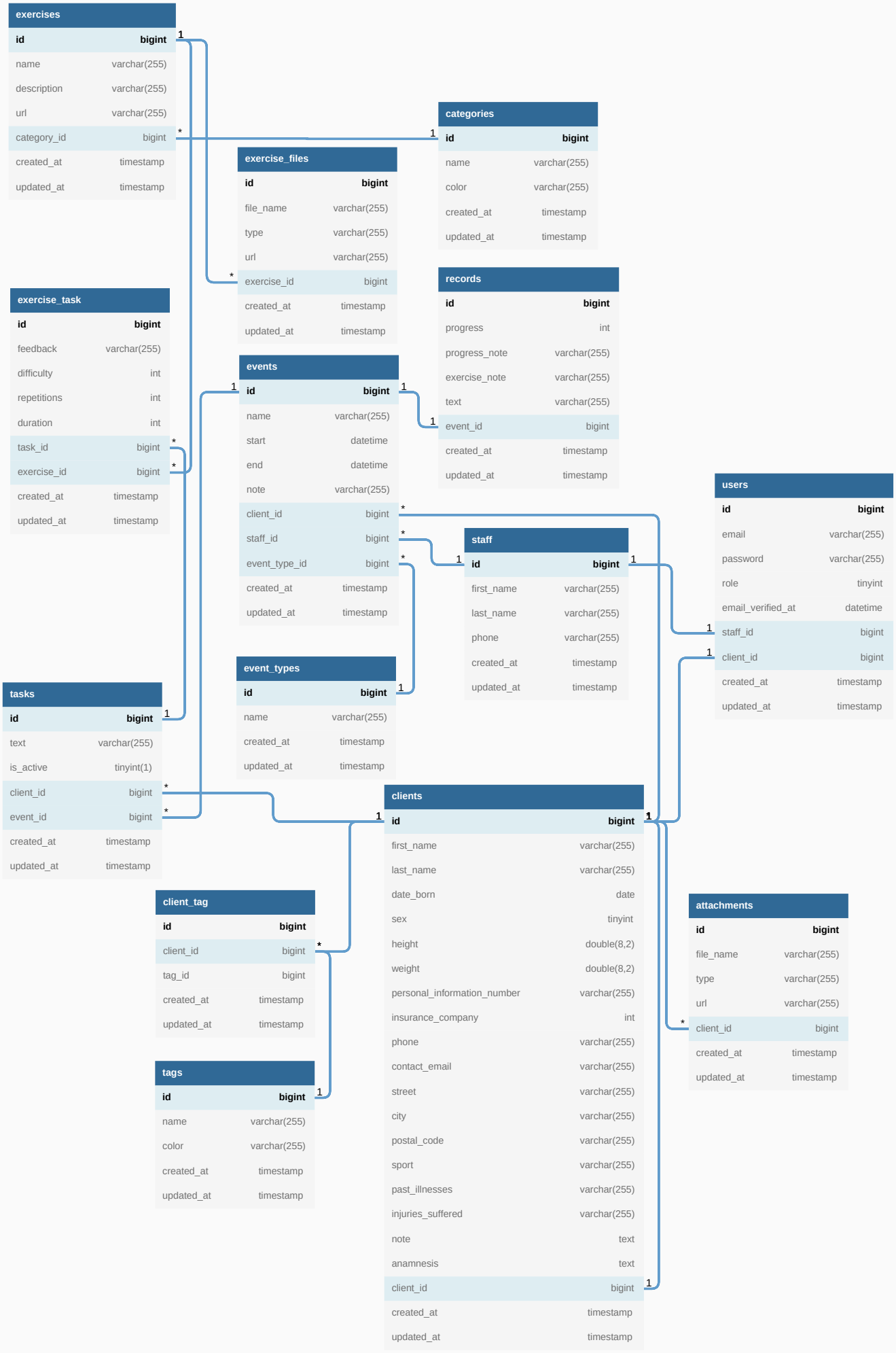
36. Doplňte prosím další informace a připomínky. Budeme rádi za jakékoliv nápady, které v dotazníku nebyly zmíněny.

Závěr

Děkujeme za Váš čas, který jste věnoval/a vyplnění tohoto dotazníku. Pokud k systému máte jakékoli dodatečné připomínky a návrhy, tak nás prosím kontaktujte na kovarm36@fit.cvut.cz nebo volakjak@fit.cvut.cz.

Příloha B

# Schéma databázového modelu


exercises
id bigint
name varchar(255)
description varchar(255)
url varchar(255)
category_id bigint
created_at timestamp
updated_at timestamp
categories
id bigint
name varchar(255)
color varchar(255)
created_at timestamp
updated_at timestamp
exercise_files
id bigint
file_name varchar(255)
type varchar(255)
url varchar(255)
exercise_id bigint
created_at timestamp
updated_at timestamp
exercise_task
id bigint
feedback varchar(255)
difficulty int
repetitions int
duration int
task_id bigint
exercise_id bigint
created_at timestamp
updated_at timestamp
records
id bigint
progress int
progress_note varchar(255)
exercise_note varchar(255)
text varchar(255)
event_id bigint
created_at timestamp
updated_at timestamp
events
id bigint
name varchar(255)
start datetime
end datetime
note varchar(255)
client_id bigint
staff_id bigint
event_type_id bigint
created_at timestamp
updated_at timestamp
users
id bigint
email varchar(255)
password varchar(255)
role tinyint
email_verified_at datetime
staff_id bigint
client_id bigint
created_at timestamp
updated_at timestamp
staff
id bigint
first_name varchar(255)
last_name varchar(255)
phone varchar(255)
created_at timestamp
updated_at timestamp
event_types
id bigint
name varchar(255)
created_at timestamp
updated_at timestamp
tasks
id bigint
text varchar(255)
is_active tinyint(1)
client_id bigint
event_id bigint
created_at timestamp
updated_at timestamp
clients
id bigint
first_name varchar(255)
last_name varchar(255)
date_born date
sex tinyint
height double(8,2)
weight double(8,2)
personal_information_number varchar(255)
insurance_company int
phone varchar(255)
contact_email varchar(255)
street varchar(255)
city varchar(255)
postal_code varchar(255)
sport varchar(255)
past_illnesses varchar(255)
injuries_suffered varchar(255)
note text
anamnesis text
client_id bigint
created_at timestamp
updated_at timestamp
client_tag
id bigint
client_id bigint
tag_id bigint
created_at timestamp
updated_at timestamp
attachments
id bigint
file_name varchar(255)
type varchar(255)
url varchar(255)
client_id bigint
created_at timestamp
updated_at timestamp
tags
id bigint
name varchar(255)
color varchar(255)
created_at timestamp
updated_at timestamp

# Příloha C

# Seznam API endpointů

## C.1 Entita uživatel

| metoda | adresa | role |
|---|---|---|
| POST | /login | všichni |
| GET | /info | admin, klient |
| GET | /logout | admin, klient |
| PUT | /user/{id} | admin,klient |

## C.2 Entita klient

| metoda | adresa | role |
|---|---|---|
| GET | /client | admin |
| POST | /client | admin |
| GET | /client/search/{tag} | admin |
| GET | /client/{id} | admin |
| PUT | /client/{id} | admin |
| DELETE | /client/{id} | admin |
| GET | /client/{id}/graph | admin |
| POST | /client/{id}/user | admin |
| POST | /client/{id}/attachment | admin |
| POST | /client/{id}/attach | admin |
| DELETE | /client/{id}/detach/{tag_id} | admin |
| GET | /client/{id}/export | admin |

## C.3 Entita typ události

| metoda | adresa | role |
|---|---|---|
| GET | /event-type | admin |
| POST | /event-type | admin |
| GET | /event-type/{id} | admin |
| PUT | /event-type/{id} | admin |
| DELETE | /event-type/{id} | admin |

## C.4 Entita událost

| metoda | adresa | role |
|---|---|---|
| GET | /event | admin |
| GET | /event/filter | admin |
| POST | /event | admin |
| GET | /event/{id} | admin |
| PUT | /event/{id} | admin |
| DELETE | /event/{id} | admin |
| POST | /event/{id}/record | admin |
| GET | /record/{id} | admin |
| PUT | /record/{id} | admin |
| DELETE | /record/{id} | admin |

## C.5 Entita záznam

| metoda | adresa | role |
|---|---|---|
| GET | /record/{id} | admin |
| PUT | /record/{id} | admin |
| DELETE | /record/{id} | admin |

## C.6 Entita kategorie

| metoda | adresa | role |
|---|---|---|
| GET | /category | admin |
| POST | /category | admin |
| GET | /category/{id} | admin |
| PUT | /category/{id} | admin |
| DELETE | /category/{id} | admin |

## C.7 Entita cvik

| metoda | adresa | role |
|---|---|---|
| GET | /exercise | admin |
| POST | /exercise | admin |
| GET | /exercise/{id} | admin |
| PUT | /exercise/{id} | admin |
| DELETE | /exercise/{id} | admin |
| POST | /exercise/{id}/upload | admin |

## C.8 Entita úkol

| metoda | adresa | role |
|---|---|---|
| GET | /task | admin |
| POST | /task | admin |
| GET | /task/{id} | admin |
| PUT | /task/{id} | admin |
| DELETE | /task/{id} | admin |
| PATCH | /task/{id}/status | admin |
| POST | /task/{id}/exercises | admin |
| PUT | /exercise-task/{id} | admin, klient |
| DELETE | /exercise-task/{id} | admin |

## C.9 Entita štítek

| metoda | adresa | role |
|---|---|---|
| GET | /tag | admin |
| POST | /tag | admin |
| GET | /tag/{id} | admin |
| PUT | /tag/{id} | admin |
| DELETE | /tag/{id} | admin |

## C.10 Informace pro klienta

| metoda | adresa | role |
|---|---|---|
| GET | /dashboard | klient |
| GET | /dashboard/{id} | klient |

## C.11 Soubory

| metoda | adresa | role |
|---|---|---|
| GET | /attachment/{id}/{filename} | všichni |
| DELETE | /attachment/{id} | admin |
| GET | /exercise-file/{id}/{filename} | všichni |
| DELETE | /exercise-file/{id} | admin |
| GET | /export/{id}/{filename} | všichni |

# Bibliografie


1. *Zdravotnický software pro rehabilitace a FYZIOTERAPIE.* [online]. ARTiiS GROUP, a.s. [cit. 2022-04-14]. Dostupné z: `http://www.rehasys.cz/`.
2. Rezervace v aplikaci RehaSys. In: *RehaSys* [online]. ARTiiS GROUP, a.s., © 2022 [cit. 2022-04-14]. Dostupné z: `http://www.rehasys.cz/pro-terapeuty`.
3. *Informační systém LBIS/4G pro lázně a OLÚ.* [online]. LAURYN s.r.o. [cit. 2022-04-14]. Dostupné z: `https://www.lauryn.cz/produkty/system-lbis4g/`.
4. *PhysiApp® - Healthcare engagement at your fingertips.* [online]. Physitrack PLC [cit. 2022-04-14]. Dostupné z: `https://www.physiapp.com`.
5. BAELDUNG. Difference Between MVC and MVP Patterns. In: *Baeldung* [online]. Baeldung, © 2021 [cit. 2022-04-15]. Dostupné z: `https://www.baeldung.com/mvc-vs-mvp-pattern`.
6. MEGALI, Tin. Model View Presenter (MVP) in Android,Part 1. In: *Medium* [online]. Medium, © 2016 [cit. 2022-04-16]. Dostupné z: `https://medium.com/@tinmegali/model-view-presenter-mvp-in-android-part-1-441bfd7998fe`.
7. CAMPBELL, Steve. Python Vs PHP: What's the Difference Between Python and PHP? In: *Guru99* [online]. Guru99, © 2022 [cit. 2022-04-15]. Dostupné z: `https://www.guru99.com/python-vs-php.html`.
8. BHAGAT, Varun. PHP Vs ASP.NET: How to Choose the Right One? In: *PixelCrayons* [online]. PixelCrayons, © 2021 [cit. 2022-04-15]. Dostupné z: `https://www.pixelcrayons.com/blog/php-vs-asp-net-how-to-choose-the-right-one/`.
9. HOODA, Parikshit. PHP vs. Node.js. In: *GeeksforGeeks* [online]. GeeksforGeeks, © 2022 [cit. 2022-04-15]. Dostupné z: `https://www.geeksforgeeks.org/php-vs-node-js/`.
10. *PostgreSQL: The World's Most Advanced Open Source Relational Database.* [online]. The PostgreSQL Global Development Group [cit. 2022-04-15]. Dostupné z: `https://www.postgresql.org`.
11. REST vs. SOAP. In: *Red Hat* [online]. Red Hat, Inc., © 2019 [cit. 2022-04-16]. Dostupné z: `https://www.redhat.com/en/topics/integration/whats-the-difference-between-soap-rest`.
12. SOAP vs. REST: A Look at Two Different API Styles. In: *Upwork* [online]. Upwork® Global Inc., © 2021 [cit. 2022-04-16]. Dostupné z: `https://www.upwork.com/resources/soap-vs-rest-a-look-at-two-different-api-styles`.

13. SVOBODA, Radek. Nejpoužívanější softwarové frameworky pro vývoj webových aplikací. In: *CoolClub* [online]. CoolPeople a.s., © 2020 [cit. 2022-04-16]. Dostupné z: https://club.coolpeople.cz/nejpouzivanejsi-softwarove-frameworky-pro-vyvoj-webovych-aplikaci/1379.html.

14. *Symfony, High Performance PHP Framework for Web Development.* [online]. Symfony SAS [cit. 2022-04-16]. Dostupné z: https://www.symfony.com.

15. Laravel vs Symfony in 2022 – Which PHP Framework Choose For Your Project? In: *ASPER BROTHERS* [online]. ASPER Spzoo Spk, © 2019 [cit. 2022-04-15]. Dostupné z: https://asperbrothers.com/blog/laravel-vs-symfony/.

16. ŠÍPEK, Robert. Active Record. In: *Zápisník vývojáře webových stránek* [online]. zooom.cz, © 2017 [cit. 2022-04-16]. Dostupné z: https://zooom.cz/active-record/.

17. ALMEIDA, Jeff. Laravel vs Lumen- What should I use? In: *Medium* [online]. Medium, © 2019 [cit. 2022-04-15]. Dostupné z: https://medium.com/@jeffalmeida_27473/laravel-vs-lumen-what-should-i-use-63c196822b2d.

18. IBM CLOUD EDUCATION. Docker. In: *IBM Cloud Learn Hub* [online]. IBM, © 2021 [cit. 2022-04-17]. Dostupné z: https://www.ibm.com/cz-en/cloud/learn/docker.

19. Co je virtuální počítač? In: *Microsoft Azure* [online]. MICROSOFT s.r.o., © 2022 [cit. 2022-04-17]. Dostupné z: https://azure.microsoft.com/cs-cz/overview/what-is-a-virtual-machine/#overview.

20. Docker vs Virtual Machines (VMs) : A Practical Guide to Docker Containers and VMs. In: *WEAVEWORKS* [online]. WEAVEWORKS, © 2020 [cit. 2022-05-04]. Dostupné z: https://www.weave.works/blog/a-practical-guide-to-choosing-between-docker-containers-and-vms.

21. *Laravel Sail.* [online]. Laravel LLC [cit. 2022-04-17]. Dostupné z: https://laravel.com/docs/9.x/sail.

22. *Draw Entity-Relationship Diagrams, Painlessly.* [online]. Holistics Software [cit. 2022-04-17]. Dostupné z: https://dbdiagram.io/home.

23. EUROPEAN PARLIAMENT AND THE COUNCIL. *Regulation (EU) 2018/1725 on the protection of natural persons with regard to the processing of personal data by the Union institutions, bodies, offices and agencies and on the free movement of such data, and repealing Regulation (EC) No 45/2001 and Decision No 1247/2002/EC.* © 2018. Dostupné také z: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32018R1725.

24. *Často kladené otázky k rodným číslům podle oblasti zpracování údajů.* [online]. Úřad pro ochranu osobních údajů [cit. 2022-02-18]. Dostupné z: https://www.uoou.cz/k%5C%2Drodnym%5C%2Dcislum/ds-5091/p1=5091.

25. Vyhláška č. 98/2012 Sb. In: *Zákony pro lidi* [online]. AION CS, s.r.o., © 2012 [cit. 2022-02-18]. Dostupné z: https://www.zakonyprolidi.cz/cs/2012-98.

26. *Laravel Sanctum.* [online]. Laravel LLC [cit. 2022-04-29]. Dostupné z: https://laravel.com/docs/9.x/sanctum.

27. *Certbot.* [online]. Electronic Frontier Foundation [cit. 2022-05-04]. Dostupné z: https://certbot.eff.org.

# Obsah přiloženého média

- `README.txt` ..... stručný popis obsahu média
- `text` ..... adresář s textem práce
  - `Bakalarska-prace-Jakub-Volak-2022.pdf` ..... text práce ve formátu PDF
  - `BP` ..... adresář obsahující zdrojové kódy textu
- `prilohy` ..... adresář obsahující přílohy práce
  - `Dotaznik.pdf` ..... export dotazníku pro fyzioterapeuty ve formátu PDF
  - `Odpovedi.pdf` ..... export odpovědí z dotazníku ve formátu PDF
  - `Databaze.pdf` ..... schéma databázového modelu ve formátu PDF
- `zdroj` ..... adresář obsahující odkazy na zdroje
  - `Zdroje.txt` ..... soubor obsahující odkaz na repozitář se zdrojovými kódy a testy