VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

INFORMAČNÍ SYSTÉM PRO PEČOVATELSKOU ORGANIZACI

INFORMATION SYSTEM FOR A CARE SERVICE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — KATEŘINA KARÁSKOVÁ
AUTHOR

VEDOUCÍ PRÁCE — Ing. VLADIMÍR BARTÍK, Ph.D.
SUPERVISOR

BRNO 2018

Ústav informačních systémů (UIFS) Akademický rok 2018/2019

Zadání bakalářské práce

21408

Studentka: **Karásková Kateřina**
Program: Informační technologie
Název: **Informační systém pro pečovatelskou organizaci**
Information System for a Care Service
Kategorie: Informační systémy
Zadání:

1. Seznamte se s principy tvorby webových aplikací, dostupnými vývojovými prostředími a frameworky.
2. Analyzujte požadavky na informační systém, který bude evidovat provedenou péči a platby za tuto péči. Bude také možné automaticky vytvářet rozvrhy péče pro jednotlivé pečovatele tak, aby byla efektivní a nedocházelo ke zbytečným přejezdům apod. Vytvořený rozvrh bude možné v přehledné formě zobrazit a vytisknout.
3. Navrhněte informační systém dle požadavků, návrh konzultujte s vedoucím.
4. Informační systém implementujte a otestujte na vhodném vzorku dat.
5. Zhodnoťte dosažené výsledky a navrhněte další možná rozšíření tohoto projektu.

Literatura:

- Naramore, E., Gerner, J. et al: PHP 6, MySQL, Apache: Vytváříme webové aplikace. Computer Press, 2009. ISBN: 978-8-0251-2767-4.
- Žára, O.: JavaScript - Programátorské techniky a webové technologie, Computer Press, 2015. ISBN: 978-80-251-4573-9

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Bartík Vladimír, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 15. května 2019
Datum schválení: 16. října 2018

Abstrakt

Informační systém pro pečovatelskou organizaci byl inspirován existujícím systémem. V rámci této práce byl vytvořen nový informační systém, který odstranil nedostatky existujícího systému. Informační systém umožňuje evidovat údaje o zaměstnancích, klientech, vykonané péči a úhradách za provedenou péči. Součástí systému je také přehled o uplynulé a také nadcházející péči u klientů. Informační systém byl implementován jako webová aplikace. To sebou přináší vyhodu přístupu do aplikace z jakéhokoliv elektronického zařízení, jenž má přístup k internetu. Vytvořený informační systém je možné rozšířit o různé funkce, dle potřeb pečovatelské organizace. Cílem bylo vytvořit informační systém, který bude plnit potřeby zaměstnanců pečovatelské organizace a zároveň práci zaměstnancům ulehčí.

Abstract

Information system for a care service was inspired by the existing system. In there was created a new information system that eliminated shortcomings of the existing system. The information system enables to record data on employees, clients, care and payments for care. The system also includes an overview of past and future client care. The information system was implemented as a web application. This brings the benefit of accessing the application from any electronic device that has Internet access. The created information system can be extended by various functions, as needed by the care organization. The goal was to create an information system that would meet the needs of the care organization's employees and at the same time make it easier for employees to work.

Klíčová slova

framework, Laravel, návrhové vzory, model-view-controller, MVC, diagram případů užití, databáze, frontend, backend, informační systém, PDF, PHP, HTML

Keywords

framework, Laravel, design patterns, model-view-controller, MVC, use case diagram, database, frontend, backend, information system, PDF, PHP, HTML

Citace

KARÁSKOVÁ, Kateřina. *Informační systém pro pečovatelskou organizaci.* Brno, 2018. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vladimír Bartík, Ph.D.

Informační systém pro pečovatelskou organizaci

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně pod vedením pana Ing. Vladimíra Bartíka, Ph.D. Uvedla jsem všechny literární prameny a publikace, ze kterých jsem čerpala.

.....
Kateřina Karásková
7. května 2019

Poděkování

Ráda bych poděkovala vedoucímu mé bakalářské práce panu Ing. Vladimíra Bartíka, Ph.D za ochotný přístup a cenné rady, které mi usnadnili sepsaní této práce.

Obsah

Kapitola 1

Úvod

Pečovatelská organizace je sociální terénní služba, která je vhodným řešením pro lidi, kteří se dostali do těžké životní situace, která způsobyla, že nejsou schopni se o sebe sami postarat. Tuto situaci mohl způsobyt špatný zdravotní stav způsobený nějakou vážnou nehodou nebo vrozenou vadou (postižením) a také tato situace mohla vzniknout z důvodu dovršení vysokého věku člověka. Pečovatelská organizace napomáhá takovým lidem setrvat ve svých domech a zachovat tak vazby na své přirozené prostředí, na svou rodinu, přátele a blízké. Jedná se individuálně poskytovanou péči. Každý z těchto lidí potřebuje péči specifickou dle jeho schopností umožňujících se o sebe postarat. Pečovatelská organizace Charita Veselí nad Moravou, kterou je inspirována tato práce, nabízí péči, kterou lze vykonávat třikrát denně v rámci návštěv u klienta. Jedná se o pomoc při zvládání běžných úkonů (např. asistence při doprovodu k lékaři), pomoc při osobní hygieně, pomoc při zajištění chodu domácnosti (úklid, nákupy, ...), zajištění a podávání stravy. Dle sociální pracovnice je s pečovatelskou organizací dohodnuta individuální péče pro každého klienta. K tomuto klientovi následně dochází pečovatelé v dohodnutých intervalech a provádí potřebnou péči. Každou péči je třeba evidovat k čemuž pečovatelská organizace Charita Veselí nad Moravou používá informační systém, který eviduje provedené úkony u klienta. Tento informační systém bohužel nemá úplnou sadu funkcí, které by jej činily efektivním a praktickým. Například v tomto informačním systému chybí funkce pro tisk výkazů, plánování rozvrhů a přehled plateb za provedenou péči.

Informační systém popsaný v této práci představuje nový informační systém, který obsahuje mimo jiné i výše zmíněné chybějící funkce spolu s přívětivým uživatelským rozhraním a dokáže tak zefektivnit a usnadnit práci pečovatelům. Popisovaný informační systém obsahuje čtyři role – pečovatel, koordinátor, účetní a administrátor. Pečovatelé zadávají provedenou péči u klientů, práci vykonanou mimo péči u klientů a mohou převzít peněžní částku od klienta za prováděnou péči. Koordinátor vytváří rozvrhy pro pečovatele, spravuje individuální péči klientů a má také stejné pravomoce a povinnosti jako pečovatelé. Účetní má přístup k platbám za péči a odpracovaným hodinám zaměstnanců. Administrátor spravuje údaje o klientech, zaměstnancích a věci související s jejich evidováním.

Cílem vytvořeného informačního systému je poskytnout pečovatelské organizaci vhodný nástroj k evidování všech potřebných údajů a provedené péče.

Informační systém pro pečovatelskou organizaci je webová aplikace implementována pomocí webových technologií. Díky tomu je to platformě nezávislá aplikace, kterou lze používat na libovolném elektrickém zařízení, jenž obsahuje webový prohlížeč a má přístup k internetovému připojení. K implementaci byl využit jazyk PHP s podporou databází MySQL a framework Laravel. K tvorbě uživatelského rozhraní byla použita šablona Ad-

minLTE využívající aplikační rámec Bootstrap jazyka CSS a aplikační rámec jQuery pro jazyk Javascript.

Kapitola 2

Technologie pro tvorbu webových aplikací

Tato kapitola popisuje technologie pro tvorbu webových aplikací. Jsou zde popsány programovací jazyky (pro backend i frontend), které se používají k tvorbě webových aplikací. Dalším pojmem je Dashboard což je šablona, která poskytuje programátorovi již vytvořený frontend, čímž ušetří programátorovi čas. Dále je zde popsán pojem framework, který usnadňuje vytváření a udržování složitého webové projektu. K tomuto pojmu se váže pojem architektonický vzor, který člení webovou aplikaci na logické části, které spolu komunikují. Poslední věcí, kterou se tato kapitola zabývá je převod HTML stránky do PDF dokumentu, což je v dnešní době poměrně nepostradatelná součást webové stránky.

2.1 Základní technologie

Základními technologiemi jsou myšleny jazyky pro frontend (vzhled webové aplikace) a backend (logika webové aplikace), které se používají pospolu pro tvorbu webových aplikací.

Frontend jazyky

Zde se nachází jazyky, jejichž kód provádí nebo interpretuje prohlížeč bez použití serveru. Lze je dále rozdělit na jazyky sestavující stránku a jazyky interagující s ní.

- HTML (Hypertext Markup Language) – značkovací jazyk, popisuje obsah webu pomocí značek, které se zapisují do spičatých závorek, značky můžou být párové (např. <\div>) a nepárové (např.
)
- CSS (Cascading Style Sheets) – stylovací jazyk, mění vzhled webu přidáním stylu HTML značkám, styl se může zapsat ke konktrétní značce, avšak pro větší přehlednost kódu se všechny styly umísťují do jednoho samostatného souboru
- JavaScript – scriptovací jazyk, nekompiluje se, přímo se interpretuje, pomocí javascriptu můžeme měnit obsah webu (přidávat či odebírat elemety)
- jQuery – javascriptová knihovna, klade důraz na interakci mezi JavaScriptem a HTML

Backend jazyky

Jsou to jazyky, jejichž kód spouští nebo iterpretuje webový server. Provádění kódu není viditelné pro žádného návštěvníka webu.

- PHP – nejběžnější jazyk pro vývoj webových aplikací, má jednoduchou syntaxi a velkou komunitu vývojářů, každá proměnná začíná značkou $ a každý příkaz je ukončen středníkem
- Java – objektově orientovaný jazyk, Java má široké spektrum použití a u webových aplikací se používá pro zabezpečení při prohlížení webu [10]

2.2 Dashboard

Dashboard slouží jako šablona pro frontend webových aplikací. Programátor se díky dashboardu nemusí zabývat vzhledem webové aplikace. Na programátorovi je naprogramovat funkci jednotlivých prvků webu tak, aby web správně fungoval. Dashboard zahrnuje řídící panel, který je hlavní součástí webové aplikace. Slouží k přechodu mezi jednotlivými stránkami naší aplikace a jako orientační bod na stránce. Většinou je viditelný na každé stránce naší aplikace. Dále dashboard zahrnuje různé UI elementy (tlačítka, ikony, ...), formuláře, tabulky a různé volitelné současti jako je například kalendář a diagramy. Každý element na dashbordu může programátor upravit dle sebe.

Existuje mnoho dashboardů a každý je jiný. Programátor si nevybírá dashboard pouze podle vzhledu, ale také podle dostupných funkčních elementů, které ve své webové aplikaci potřebuje. Existují dashboardy volně dostupné i komerční. Komerční dashboardy obvykle poskytují více funkčních elementů než ty volně dostupné. K volně dostupným populárním dashboardům patří například Now UI Dashboard, AdminLTE Control Panel Template a Material Dashboard React.[8]

2.3 Framework

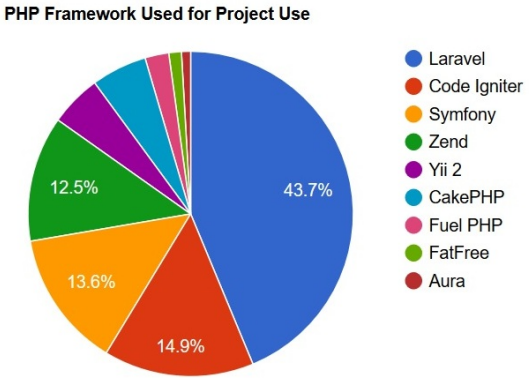
Obecně je framework nadstavba či program nad nějakým jazykem sloužící jako podpora nebo průvodce při vývoji či programování různých aplikací. Tato struktura označuje komponenty, které by měly spolu komunikovat, a způsob jakým by měly být spolu propojeny. [21] Framework pro tvorbu webových aplikací je obvykle sada knihoven programu, komponent a nástrojů umožňující vytvářet a udržovat složité projekty webových aplikací. Za předpokladu dodržení určitých pravidel, zajišťuje framework větší bezpečnost a tvárnost aplikace než jazyk samotný a tím je vývoj webových aplikací rychlejší a efektivnější.

Frameworky jsou navrženy tak, aby zjednodušily práci programátorovi při programování aplikace a to tím, že většinou mají zaimplementované nejvíce používané komponenty (např. návrhové vzory) a mnoho komponent, které potřebujeme použít, se píšou snáze. Validace některých komponent může proběhnout ještě před odesláním dotazu na server napsáním jednoho řádku kódu (např. formulář). Dalšími výhodami jsou zajištění redukce opakujícího se kódu a oddělení logiky od rozhraní. Těchto výhod docílíme použitím návrhových vzorů. [10]

Při výběru frameworku je dobré se nejprve zaměřit na verzi PHP, kterou framework používá. Ne všechny webové hostingy při nasazení nové verze PHP ji okamžitě instalují na svůj webový server a proto někdy může být lepší zvolit starší verzi PHP, aby byla kompatibilní s tou na webovém serveru. V mnoha případech se volí novějši verze, neboť vůči

té starší obsahuje vylepšení, z čehož plynou další funkce pro usnadnění práce při programování. Všechny frameworky však nepoužívají nejnovější verzi PHP a tak je třeba zjistit, jakou verzi PHP používá náš vybraný framework a dle toho si rozmyslet, zda ho použít nebo ne.

V současné době se používají frameworky, které vytvářejí jednu konktrétní webovou aplikaci. Takové frameworky aplikují určitý návrhový vzor nebo architektonický vzor – nejčastěji MVC (model-view-controller). [18] Nejpoužívanější frameworky v současné době jsou vidět na obrazku 2.1.



Obrázek 2.1: Nejpoužívanější frameworky současnosti (zdroj: https://coderseye.com/best-php-frameworks-for-web-developers/)

Laravel

Laravel je nejpoužívanější framework současnosti a to skrz jeho výhody – rychlý vývoj aplikací pomocí architektury MVC, vysoká míra abstrakce, organizovanost kódu a souborů, velmi silné šifrovací balíčky, rozsáhlá a dobře propracovaná dokumentace včetně naučných videí a stále rozrůstající se komunita vývojářů. [24]

Pro vývoj webových aplikací ve frameworku Laravel je doporučeno používát lokální vývojové prostředí Homestead. Laravel Homestead je virtuální stroj, který pokrývá veškeré systémové požadavky na vývoj aplikace. Pokud se vývojovému prostředí Homestead vyhneme, je třeba se ujistit, že používaný server splňuje potřebné systémové požadavky. [3]

CodeIgniter

CodeIgniter framework si dává za cíl vytvářet aplikace rychleji, než kdyby byly psány od nuly a to díky poskytnutí bohaté sadě knihoven, jednoduchému rozhraní a logické struktuře pro přístup k těmto knihovnám. Díky tomu je velmi přívětivý k vývojářům. [1] Další vyhody jsou: nepotřebnost žádných zvláštních závislostí a podpory, používání standartních databází, funguje dobře téměř na všech sdílených a specializovaných hostingových platformách, má kvalitní dokumentaci a dlouhodobou podporu. Framework také poskytuje vývojářům velkou svobodu. To znamená, že modely a pohledy si vývojář volí volitelně a může použít vlastní konvence pro kódování a pojmenování. CodeIgniter framework není výhradně založen na MVC modelu. [24][9]

Symfony

Symfony framework umožňuje vytvářet webové aplikace. Sestává z nejpoužívanějších komponent, které jsou oddělené a mohou se opakovaně používat. Ze Symfony komponent je například sestaven systém pro správu obsahu Drupal nebo systém pro vytvoření interaktivního fóra phpBB. [7] Symfony framework je velmi stabilní, vysoce výkonný díky ukládání do mezipaměti a má rozsáhlou dokumentaci a též komunitu vývojářů. [24]

2.4 Převod HTML do PDF

V současné době je převod HTML stránky do PDF dokumentu velmi žádaný a také praktický. V případě, že uživatel potřebuje vytisknout obsah webové stránky, je převod této stránky do formátu PDF dokumentu nezbytný. Pokud si uživatel chce obsah webové stránky pouze uložit k sobě do počítače a kdykoliv si tento obsah zobrazit offline, což znamená bez přístupu k internetu, musí si tento obsah také uložit jako PDF dokument.

V současné době lze do některých webových prohlížečů přidat nástroj, který převod HTML stránky do PDF dokumentu provede. Tento nástroj je vhodný v případě, že samotná webová stránka tuto možnost převodu neumožňuje. Jeden z těchto nástrojů poskytuje společnost Adobe Systems ve své skupině progamů Adobe Acrobat. Nástroj s názvem Adobe PDF, který si nainstaluje aplikace Acrobat, je přístupný ve webových prohlížečích Internet Explorer (verze 7.0 nebo novější), Google Chrome a Firefox (verze 3.5 nebo novější). Pomocí tohoto nástroje lze snadno převést HTML stránku do PDF včetně obrázků JPEG, CSS stylů, tabulek a formulářů. Po převodu se výsledný PDF dokument chová podobně jako HTML stránka – funkční hypertextový odkaz, multimediální soubory a další. Nástroj Adobe PDF umožňuje převést celou webovou stránku, ale také umožňuje převést pouze vybrané části této stránky. [6]

Jinou variantou pro převod HTML stránky do formátu PDF je použití knihoven, které jsou přímo vytvořeny pro převod do PDF formátu. Tahle varianta je nezávislá na tom, zda webový prohlížeč podporuje nějaký nástroj pro převod HTML do PDF. Knihoven pro převod HTML stránky do PDF dokumentu existuje mnoho a záleží jen na programátorovi, kterou z nich si vybere a použije ve svém projektu. Každá knihovna je jiná a liší se v různých parametrech. Například obtížnost implementace požadované funkčnosti, v jakých případech lze knihovnu použít a co knihovna zvládne udělat. Nejpoužívanější open source knihovny pro převod HTML stránky do PDF dokumentu jsou popsány dále.

2.4.1 FPDF

FPDF je PHP třída, která umožňuje vytvářet PDF dokumenty s použitím jazyka PHP. Poslední verze FPDF potřebuje k funkčnosti PHP alespoň verze 5.1. Funkce FPDF jsou na vysoké úrovni. Knihovna umožňuje nastavení jednotky míry (cm, inches), formátu stránky, okrajů, záhlaví a zápatí, poskytuje automatické zalomení stránky a řádků. Podporuje obrázky typu JPEG, PNG a GIF, změnu barev, odkazy a kompresy stránky.

2.4.2 DOMPDF

Knihovna DOMPDF slouží k rozvržení stránky (layout) pomocí HTML značek a CSS stylů. Je to vykreslovací prostředek naprogramovaný v jazyku PHP a řízený stylem. DOMPDF dokaže zpracovat většinu CSS stylů verze 2.1 a několik vlastností CSS3 včetně pravidel @import, @media a @page. Podporuje prezentaci HTML atributů verze 4.0, komplexní

tabulky, externí styly, základní podporu formátu SVG a obrázky typu GIF, PNG, BMP a JPEG. Pro funkčnost DOMPDF potřebuje PHP verze 5.3.0 nebo vyšší, DOM, GD a MBString rozšíření a knihovny php-font-lib a php-svg-lib.

2.4.3 Snappy (wkhtmltopdf)

Tato knihovna umožňuje generování miniatur, snímků a PDF dokumentů z HTML stránek nebo z url. Používá webkit založený na wkhtmltopdf a wkhtmltoimage k dispozici na operačním systému OSX, linux a windows. wkhtmltopdf a wkhtmltoimage jsou nástroje příkazové řádky s open source kódem a nevyžadují grafické zobrazení.

2.4.4 TCPDF

TCPDF je knihovna pro snadné generování PDF dokumentů z HTML stránek. Knihovna má mnoho funkcí a snadno se přizpůsobuje programátorovým potřebám. Knihovna podporuje písma Unicode. Na oficiální domovské stránce TCPDF lze nalézt rozsáhlou sbírku příkladů kódu a z nich vygenerovaných PDF dokumentů, která slouží jako dobrý příklad toho, co knihovna dokáže a také jako předloha pro úpravu generovaného PDF dokumentu. Pro základní funkce nejsou zapotřebí žádné externí knihovny. Umožňuje nastavení standartních formátů stránky i možnost nastavení vlastního formátu stránky, nastavení velikosti okrajů a nastavení jednotky míry. Další funkce, které knihovna umožňuje jsou: psaní zprava doleva, publikování některých kódů (XHTML + CSS, Javascript, Forms), zobrazení obrázků (JPEG, PNG, SVG), geometrických útvarů, 1D a 2D čárových kódů, umožňuje použití barevného profilu ICC, RGB, CMYK, stupně šedi a průhledné fólie, automatické nastavení záhlaví a zápatí, automatické zalomení stránky a řádků, možnost nastavení více sloupců, automatické číslování stránek, viditelnost vrstev a objektů, ...[5]

2.4.5 Srovnání knihoven

Všechny knihovny obsahují základní funkce pro převod HTML stránky do PDF dokumentu. Ne však se s každou z nich dobře a snadně pracuje. Z pohledu potřeby externích knihoven a požadavků na funkčnost je zřejmě nejméně praktická knihovna DOMPDF, která ke svému používání potřeuje různá rozšíření a knihovny navíc. Co se týka pokročilejších funkcí na nastavení výsledného vzhledu PDF dokumentu v popředí jsou knihovny TCPDF, DOMPDF a FPDF. Tyto knihovny obsahují mnoho funkcí pro vytvoření našeho výsledného vzhledu, ale nejvíce funkcí obsahuje knivna TCPDF, navíc její použití je velmi snadné.

Kapitola 3

Návrhové vzory a MVC

Tato kapitola se zabývá popisem chování nejznámějších návrhových vzorů a popisem chování MVC architektury, která z některých návrhových vzorů vychází.

3.1 Návrhové vzory

Návrhové vzory jsou doporučené postupy k řešení opakujících se problémů při objektově orientovaném programování. Používají se k návrhu architektury softwarové aplikace. Jejich použitím by se měla snížit pravděpodobnost výskytu chyb v aplikaci neboť návrhové vzory jsou šablony nebo předlohy, do kterých dosazujeme vlastní objekty, rozhraní a třídy. Nemusíme tedy vymýšlet vlastní architekturu, ale můžeme použít architekturu, která je již odzkoušená a ověřená jak po funkční stránce, tak po stránce praktické. Co se týká praktické stránky, návrhové vzory počítají s budoucím rozšířením aplikace, následná rozšíření a úpravy zaberou výrazně méně práce a aplikace bude ve výsledku fungovat efektivněji a bude spolehlivější. Návrhové vzory bývají nejčastěji popisovaný pomocí jazyku UML, přesněji jeho diagramů, které jsou doprovázeny slovním popisem. [23] Obecně mají návrhové vzory čtyři základní prvky:

1. Jméno – jednoslovný nebo dvouslovný popis problému, který tento návrhový vzor řeší. Zvolení výstižných názvů pro vzory bylo nejtežší částí pro vývoj katologu vzorů.

2. Popis problému – vysvětluje problém, který návrhový vzor řeší, a jeho souvislosti. Z něj by měl programátor pochopit, kdy daný návrhový vzor může účelně použít. Někdy popis problému obsahuje seznam podmínek, které musí být splněny před samotným použitím, aby použití vzoru mělo smysl.

3. Popis řešení – popisuje prvky tvořící návrh neboli jejich vztahy, spolupráci a povinnosti. Řešení nepopisuje konkrétní způsob implementace, je to pouze šablona, která může být použita v různých situacích.

4. Důsledky – výsledky uplatnění vzoru, jsou rozhodující při vyhodnocování jiných návrhových řešení a pochopení důsledků a přínosů tohoto vzoru. Důsledky vzoru zahrnují vliv na jeho flexibilitu, rozšířitelnost přenosytelnost systému.

V návrhových vzorech se často vyskytují dva pojmy – třída a abstraktní třída. Třída je základní konstrukční prvek pro objektově orientované programování. Slouží jako šablona pro vytváření instancí třídy (objektů) a obsahuje definice atributů (vlastností) a metod (funkcí

objektů). Na každou instanci třídy vzniklou z jedné třídy můžeme zavolat shodnou sadu metod, ale hodnoty atributů u jednotlivých instancí třídy se mohou lišit.

Abstraktní třída slouží jako předpis metod a atributů pro další (konkrétní) třídy odvozené pomocí dědičnosti. Metody abstraktní třídy mohou být neimplementované (pouze deklarace funkce, implmentace vyžadována ve třídě potomka) nebo implementované (metoda bude rovnou použitelná ve třídě potomka díky dědičnosti).[16]

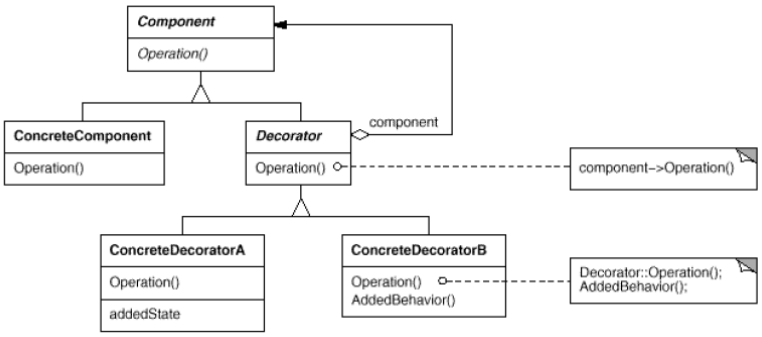
Návrhové vzory se dělí do čtyř skupin. Jsou to návrhové vzory strukturální (structural), tvořivé (creational), behaviorální a architekturní. [15]

3.2 Strukturální návrhové vzory

Existuje sedm strukturálních návrhových vzorů, tři nejpoužívanější budou popsány níže a mimo ně existuje Bridge, Proxy, Flyweight a Facade. Každý z vzorů má vliv na flexibilitu, bezpečnost a dlouhodobou použitelnost námi vytvořené aplikace. Mohou být tedy použity již při návrhu aplikace nebo později při její údržbě. Srukturální vzorce se zabívají skladbou tříd a objektů tak, aby z nich mohli být vytvářeny větší struktury. Účel strukturálních vzorů je například dynamické přidávání či odebírání funkcí k existujícím objektům (Dekorátor), řízení přístupu k objektu(Proxy), umožnit nezávyslý vývoj rozhraní a implementace komponet (Bridge), porovnávání jinak nekompatibilních rozhraní (Adaptér) a další.

3.2.1 Decorator (Dekorátor)

Úlohou Dekorátoru je dynamicky přidávat nové odpovědnosti objektu, což je alternativa pro rozšiřivání objektů o novou funkcionalitu bez použití třídní dědičnosti. Dekórovný objekt je v podstatě původní třída s nějakými přidanými vlastnostmi. Výhodou vzoru Dekorátor je, že původní objekt neví o jakékoli dekoraci, neexistuje žádná třída se všemi možnostmi dekórovaných objektů a jednotlivé dekorace jsou na sobě nezávislé.



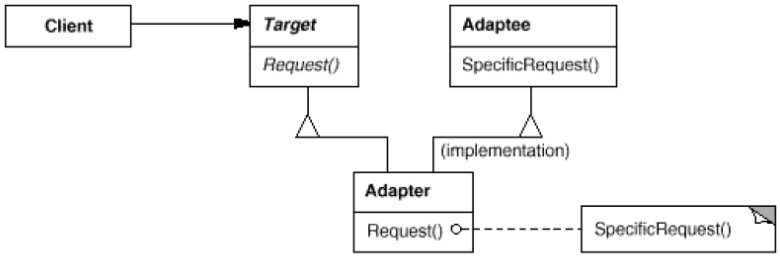
Obrázek 3.1: UML diagram Dekorátor [15]

- `Component` – definice rozhrani objektů
- `ConcreteComponent` – definice objektů, ke kterým lze přidat nové zodpovědnosti
- `Decorator` – odkaz na objekt `Component`, definice rozhraní odpovídajícímu rozhraní `Component`
- `ConcreteDecoratorA,B` – přidává zodpovědnosti komponentě

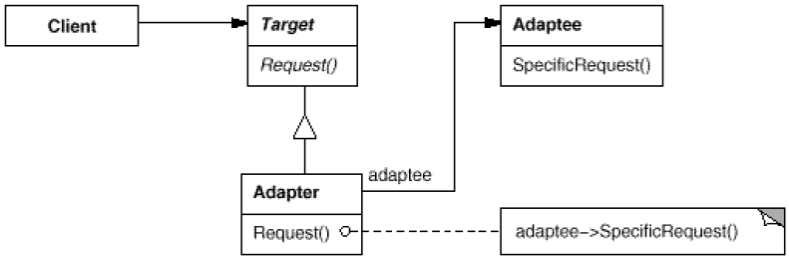
Vzor Dekorátor se používá ve světě grafiky, hudby i videí. Například obrázek může být libolně upraven a video může být komprimováno různou rychlostí.

3.2.2 Adapter (Adaptér)

Vzor Adaptér, též známý jako Wrapper, umožňuje spolupraci třídám, které mají zcela odlišné rozhraní a bez Adaptéru by spolu nemohli spolupracovat. Jinými slovy převádí rozhraní jedné třídy na jiné rozhraní očekávané klientem. Adaptér se používá v případě, že chceme, aby komunikovaly dvě třídy z různým rozhraním, a v případě, že chceme vytvořit opakovatelně použitelnou třídu spolupracující s třídami, které nemusí mít nutně kompatibilní rozhraní.



Obrázek 3.2: UML diagram třídní Adaptér [15]



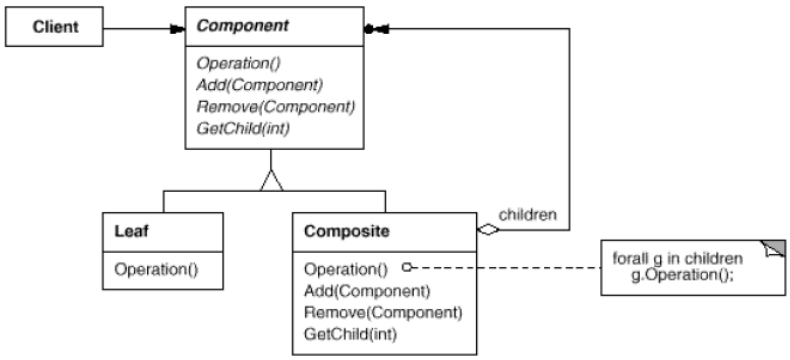
Obrázek 3.3: UML diagram objektový Adaptér [15]

- `Client` – spolupracuje s objekty, které mají stejné rozhrané jako `Target`
- `Target` – definuje rozhraní, které používá klient
- `Adaptee` – definuje existující rozhraní, které se má přizpůsobit jinému
- `Adapter` – přizpůsobuje rozhraní adaptéru k rozhraní `Target`

Někdy je potřeba mít objekty typu Target nebo Adaptee trasnsparentní, čehož se dá docílit tak, že adaptér zdědí obě rozhraní. Takového zdědění však ve všech programovacích jazycích nelze docílit a místo způsobu zdědění se používá obousměrný adaptér (Two-way Adapter). Obousměrný adaptér dokáže řešit problémy, kde v jednom systému je třeba používat vlastnosti druhého systému a naopak. Třída adaptéru je nastavena tak, aby absorbovala důležité společné vlastnosti obou systémů a následně vzniklé objekty adaptéru budou přijatelné pro oba systémy. Tato metoda se dá použít na více než dva systémy současně.

3.2.3 Composite (Skladba)

Skladba uspořádává objekty do stromových struktur. Každá stromová struktura reprezentuje dílčí hiearchii, ve které jsou jednotlivé i skupinové komponenty zpracovávány stejným způsobem a klient tak může pracovat se skupinou komponent stejně jako s jednou komponentou.



Obrázek 3.4: UML diagram Skladba [15]

- `Component` – definuje operace a výchozí chování použitelné u obou objektů
- `Leaf` – implementuje operace, které provádí dále nerozložitelné objekty
- `Composite` – implementuje operace pro dále rozložitelné objekty

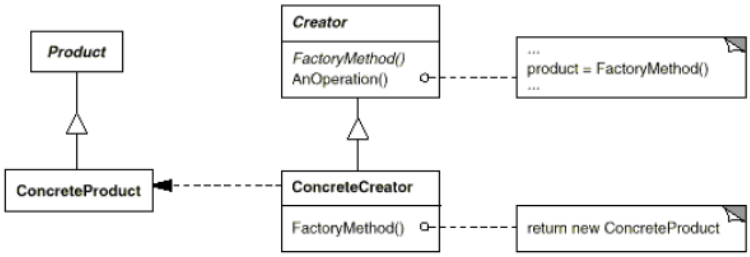
3.3 Tvořivé návrhové vzory

Tvořivé návrhové vzory pomáhají vytvářet systém, který je nezávilý na tom, jak jsou objekty vytvářeny, složeny a reprezentovány. Třídy vytvořené instanciováním jsou měněny pomocí třídní dědičnosti. Tvořivé návrhové vzory poskytují informace o tom, které třídy systém používá a naopak skrývá podrobnosti o vytváření instancí tříd a o jejich sestavení.

Nejpoužívanější tvořivé návrhové vzory jsou uvedeny níže, mimo uvedené sem ještě patří Builder a Prototyp.

3.3.1 Factory method (Tovární metoda)

Tovární metoda vytváří objekty s tím, že rozhodnutí o tom, která třída se má konkretizovat nechává na podtřídách. Rozhraní vytvořeného objektu tedy může implmentovat některá z podtříd, která z nich však bude instanciována, závisí na informacích dodaných klientem nebo vyextrahovaných z aktuálního stavu.



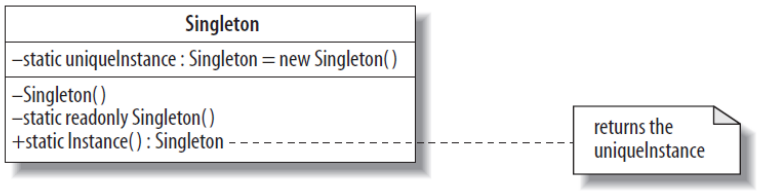
Obrázek 3.5: UML diagram Tovární metoda [15]

- `Product` – rozhraní produktu
- `ConcreteProduct` – třídy implementující konkrétní produkt
- `Creator` – poskytuje funkci `FactoryMethod`
- `ConcreteCreator` – funkce pro vytváření konkrétního produktu

Tento vzor dosahuje nezávislosti na třídách specifických pro danou aplikaci. Programátor naprogramuje rozhraní a zbytek nechá na návrhovém vzoru.

3.3.2 Singleton (Jedináček)

Účelem vzoru Jedináček je zajištění, aby třída měla pouze jednu instanci a všechny požadavky jsou směrovány právě na ni. Přístup k této instanci (objektu) je přes globální přístupový bod. Objekt by neměl být vytvořen, dokud ho není potřeba.



Obrázek 3.6: UML diagram Jedináček [19]

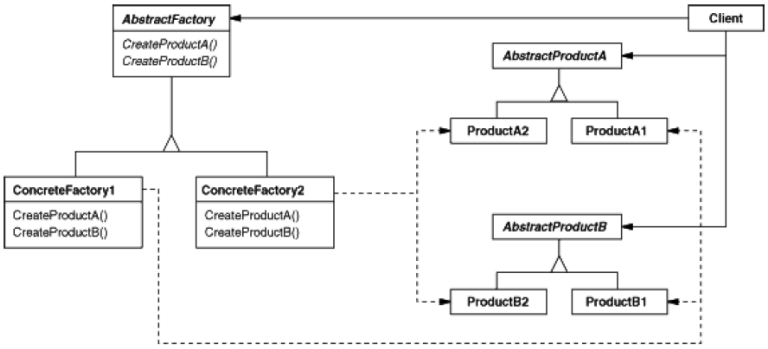
- `Singleton` – třída obsahuje mechanismus pro instanciaci sama sebe
- `Instance` – vlastnost pro vytváření instance a přístupového bodu k ní

Mnoho dalších návrhových vzorů může použít návrhový vzor Jedináček k zajištění vytvoření pouze jedné kopie třídy. Účinnost vzoru závisí pouze na vývojařích. Tento návrhový vzor se používá v případě, že potřebujeme zajistit existenci právě jedné instance třídy a když je kontrolovaný přístup k instanci nezbytný.

3.3.3 Abstract factory (Abstraktní továrna)

Abstraktní továrna zajišťuje rozhraní pro vytváření rodin příbuzných nebo závislých objektů bez specifikování jejich konkrétních tříd a udržuje jejich detaily nezávislé na zákaznících.

Abstraktní továrna může být rozdělena na konktrétní továrny. Každá z nich může vytvářet produkty různých typů a kombinací. Jediný způsob, jak ziskat definice produktu a názvy tříd od klienta, je přes továrnu. Díky tomu mohou být produktové rodiny snadno vyměněny či aktualizovány bez narušení struktury klienta.



Obrázek 3.7: UML diagram Abstraktní továrna [15]

- `AbstractFactory` – deklaruje rozhraní pro operace vytvářející abstraktní objekty produktu
- `ConcreteFactory` – implementuje operace vytvářející konkrétní objekty produktu
- `AbstractProduct` – deklaruje rozhraní pro typ objektu produktu
- `ConcreteProduct` – implementuje rozhraní AbstractFactory; definuje objekt produktu jenž má být vytvořen odpovídající továrnou
- `Client` – používá pouze rozhraní AbstractFactory a AbstractProduct
- `ProductA1,A2,B1,B2` – třídy implementjící rozhraní AbstractProduct a definující objekty produktu jenž mají být vytvořeny odpovídajícími továrnami

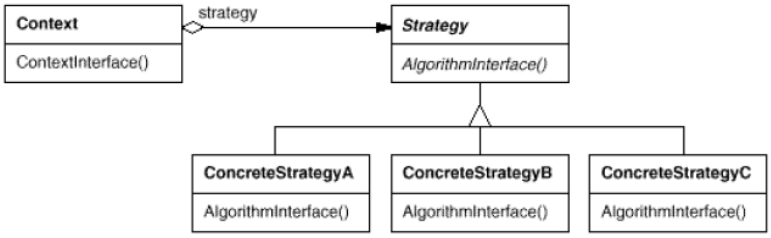
Nejčastěší omezení abstraktní továrny je nesnadné přidávání dalších druhů produktů. Pro přidání nového produktu je třeba aktualizovat tovární rozhraní, všechny jeho konkrétní podtřídy a klient se též musí změnit, aby mohl využívat nově přidané produkty.

3.4 Behaviorální návrhové vzory

Behaviorální navrhové vzory se zabývají algoritmy a komunikací mezi nimi, kterou je obtížné sledovat z důvodu komplexního řízení toku dat. Operace tvořící jeden algoritmus mohou být rozděleny mezi několik tříd. Toto uspořádání je těžce udržovatelné a spravovatelné. Behaviorální vzory udávají jak rozdělit operace mezi jednotlivé třídy a jak vhodně optimalizovat komunikaci mezi nimi. Mimo níže popsané behaviorální návrhové vzory existuje dále vzor State, Visitor, Interpreter, Memento, Chain of Responsibility, Command, Iterator a Mediator. [15][19]

3.4.1 Strategie (Strategy)

Vzor Strategie osamostaňuje algoritmus, odstraní jej z hostitelské třídy a dá jej do samostatné. Tím se vyhneme kódu s hromadou podmíněných příkazů. Může existovat více algoritmů řešící jeden problém a použití každého z nich se může odvíjet od situace, která má být právě řešena. [19] Vzor Strategie v průběhu programu vyměňuje implementace algoritmů na žádost klienta nebo na základě nastavení bez nutnosti změny kódu. [4]



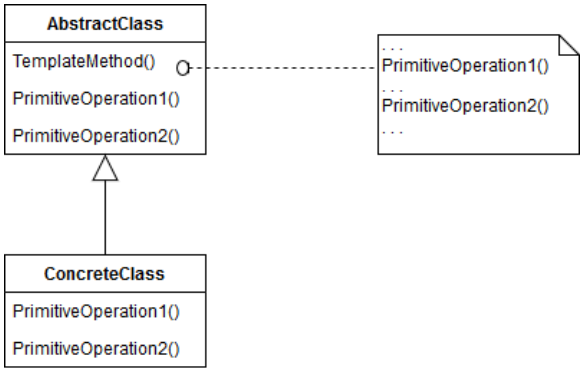
Obrázek 3.8: UML diagram Strategie [15]

- `Strategy` – deklaruje rozhraní všem podporovaným algoritmům, toto rozhraní používý `Context` k vyvolání algoritmu definoveného v `ConcreteStrategy`
- `ConcreteStrategyA,B,C` – implementuje konkrétní algoritmus pomocí rozhraní `Strategz`
- `Context` – udržuje odkaz na objekt `Strategy`, je nakonfigurován objektem `ConcreteStrategy`

3.4.2 Šablonová metoda (Template Method)

Šablonová metoda definuje kostru algoritmu a umožňuje mu odložit určité kroky tohoto algoritmu do podtříd. Struktura algoritmu se nemění, ale některé jeho části jsou zpracovávány jinde. Podtřídy mohou předefinovat určité kroky algoritmu aniž by změnily jeho strukturu.

Šablonová metoda je užitečná pro kterýkoliv systém, jenž odloží své primitivní operace na jiné místo než je základní třída. Metoda má schopnost zpracovávat sekvenci více volání metod. Některé implementace těchto metod jsou odloženy do podtříd. Rozhodnutí, které kroky algoritmu budou invariantní (neměnné) a které variantní (lze je přizpůsobit), je na programátorovi. Neměnné kroky algoritmu jsou implementovány v základní třídě a ty, co se mají přizpůsobit, jsou dány výchozí implementací nebo vůbec žádnou. Šablonová metoda je velmi užitečná ve spojení se vzorem Strategie.

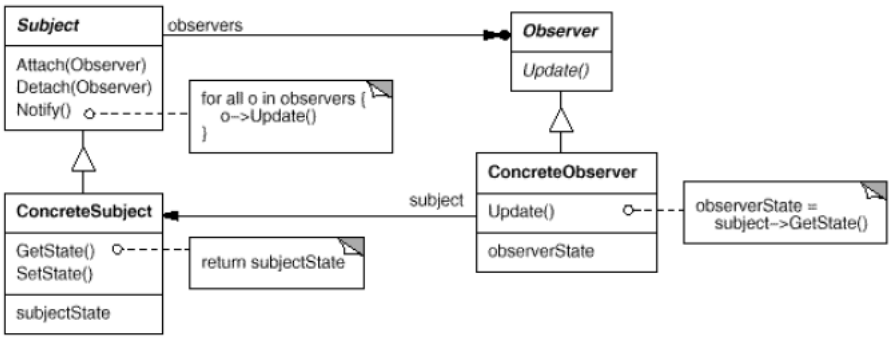


Obrázek 3.9: UML diagram Šablonové metody

- `AbstractClass` – definuje primitivní operace (implementace kroků algoritmu), které konkretizují podtřídy
- `ConcreteClass` – implementuje metodu šablony definující kostru algoritmu; volá primitivní operace stejně jako operace definovaní v `AbstractClass` nebo v jiných objektech

3.4.3 Pozorovatel (Observer)

Pozorovatel definuje vztahy mezi objekty tak, že při změně stavu jednoho objektu budou informovány a aktualizovány ostatní závislé objekty. Obvykle existuje jeden hlavní objekt, dle kterého se mění ostatní závislé objekty. Vzor Pozorovatel je široce používán u webovýh stránek, především u blogů a softwarových systémů jako je Flickr. Vzor Pozorovatel se může tvářit jako model "publish/subscribe"a architektonický vzor Model-View-Controller.



Obrázek 3.10: UML diagram Pozorovatel [15]

- `Subject` – poskytuje rozhraní pro připojení a odpojení objektů Pozorovatele; zná své pozorovatele
- `Observer` – definuje rozhraní pro aktualizaci objektů, které by měly být informovány o změnách v hlavním objektu
- `ConcreteSubject` – uchovává stav zájmu objektů `ConcreteObserver` a informuje pozorovatele o změnách jeho stavu

- `ConcreteObserver` – udržuje odkaz na object `ConcreteSubject`; implementuje rozhraní pro aktualizaci `Observer` pro udržení svého stavu v souladu s předmětem [15][19]

3.5 MVC architektura

Model-View-Controller (dále jen MVC) je architektonický vzor, na kterém bývá postavena většina populárních webových frameworků. Používá se především u náročnějších projektů jako pomyslná struktura aplikace. Celá architektura se skládá ze tří logických částí:

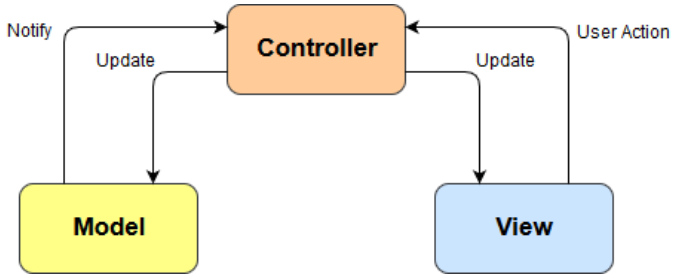
- model
- view (pohledy)
- controller (kontrolery)

Před vznikem MVC byla tendence tyto logické části slučovat. Základní myšlenkou MVC je oddělení logiky aplikace od jejího výstupu, přesněji řečeno logické operace (PHP, databázové) a vykreslování HTML výstupu se nikdy nebudou nacházet v jednom souboru. Tato myšlenka se zdá srozumitelná a jednoduchá, avšak její realizace je poměrně obtížná. Každou část MVC architektury lze samostatně upravovat aniž by dopad těchto úprav na zbylé části byl pozorovatelný. To znamená, že vývojáři logiky a vývojáři uživatelských rozhraní mohou pracovat současně a téměř nezávisle na sobě, aniž by vzájemně museli řešit, co a jak mají naprogramovat.

Za logickou část aplikace se považuje model a kontroler, část pohled je uživatelským rozhraním. Rozdíl mezi částmi model a kontroler se může překrývat, což může vést k nesprávnému použití tohoto architektonického vzoru. Není vždy jednoduché rozhodnout, zda danou metodu umístit do modelu nebo kontroleru. [12][11]

Díky oddělení části modelu od části pohled je umožněno implemetovat několik uživatelských rozhraní, které využívají společnou logiku aplikace. Tyto dvě části spolu komunikují či spolupracují skrz kontroler, nikdy ne navzájem. Vykreslení libovolné webové stránky tedy probíhá v těchto krocích: Uživatel zadá URL adresu webové stránky, kontroler zjistí tuto adresu, zpracuje ji a rozloží ji na jednotlivé částí. Následně zjistí, zda je URL adresa validní a pokud ano, zavolá příslušný model, který vykoná potřebné funkce a předá data kontroleru. Ten zavolá příslušný pohled, kterému předá potřebná data a prohlížeč tento pohled vykreslí uživateli. [20]

V MVC lze nalézt známky návrhových vzorů. Hlavní vztahy v MVC vychází z návrhových vzorů Pozorovatel, Skladba a Strategie. Například návrh, který odděluje modely od pohledů, lze aplikovat na obecnější problém: Oddělit objekty takovým způsobem, aby změna jednoho objektu se projevila na různém počtu dalších objektů a to aniž by měněný objekt znal jejich podrobnosti. Návrhový vzor, který má takové chování se nazývá Pozorovatel. Zacházení se složeným pohledem, stejně jako s jednoduchým, což je chování třídy `CompositeView` (podtřída třídy `View`), lze zobecnit na problém, kdy se skupinou objektů chceme pracovat jako s jedním objektem. Takový obecnější návrh popisuje návrhový vzor Skladba. Návrhový vzor Strategie ztvárňuje vztah pohled - kontroler. Například odezva na uživatelský vstup v pohledu se nastavuje v kontroleru. Abychom změnily odezvu na uživatelský vstup, můžeme vyměnit naši instanci kotroleru za jinou, která bude odezvu zpracovávat jinak. Dále například můžeme v MVC nalézt Tovární metodu ke specifikaci výchozí třídy kontroleru pro náhled. [14]



Obrázek 3.11: Komunikace v MVC modelu

Každá část MVC je prezentována třídou, která dědí z abstraktních tříd Model, View a Controller. [12]

Model

Model obsahuje data aplikace. Tato data získává zvenčí, ukládá je, mění je a poskytuje je k dalšímu zpracování. Neobsahuje žádnou logiku, která by prezentovala data uživateli, to zařizuje kontroler. Model například používá dotazy na databázi, aby získal potřebná data, která poskytuje kontroleru a ten je poskytne pohledu. Stejný model by měl být opakovaně použitelný v různých rozhraních dané aplikace.

View

Stará se o zobrazení výstupu – zobrazení dat modelu v příslušném rozhraní. Nejčastěji se jedná o HTML šablonu, ve které lze používat určité PHP cykly, podmínky a proměnné, které jsou definovány v kontrolerech a získávají data z modelu. PHP komponenty jsou v HTML šabloně dostupné přes určitý tag.

Controller

Kontroler komunikuje s pohledem i modelem, tím propojuje celou architekturu a řídí tok událostí v programu. Nachází se zde hlavní logika aplikace – kontroler získává data od modelu a předává je do pohledu k zobrazení a také získává data z pohledu a dává je ke zpracování modelu. [12]

Kapitola 4

Databáze

Databáze je organizovaná sbírka dat či souborů. Obecně databáze bývá uložená a přístupná elektronicky v počítačovém systému. Databáze jsou zdrojem dat pro aplikace, které pro svou funkčnost a použitelnost potřebují spoustu dat, které je třeba mít uložené a zároveň snadno přístupné. S pojmem databáze souvisí pojem perzistence dat. Data uložená v databázi jsou perzistentní, což znamená, že data jsou dostupná i po ukončení aplikace, která je vytvořila, případně po restartu počítače, na kterém jsou data uložena. Data jsou tedy ve skutečnosti uložena na energeticky nezávislém médiu, na kterém data zůstanou uložena i při přerušení napájení. Tímto médiem v dnešní době může být magnetický disk. Dříve existovaly pouze tři databázové modely – hiearchický, síťový a relační. Později vznikly postrelační databáze což jsou relační databáze rozšířeny o určitou specializaci na databázové úrovni, neboť aplikační řešení by bylo nedostačující. Mezi postrelační databáze patří prostorová databáze, objektově orientovaná databáze, deduktivní databáze, temporární databáze, multimediální databáze a aktivní databáze. Databázové modely zahrnují:

- Definici logické struktury – způsob, jakým jsou data v databázi strukturována na logické úrovni bez ohledu na implementaci
- Definici obecních integritních pravidel – omezení kladená na data, která musí být dodržena
- Formální dotazovací jazyk – slouží k formulování dotazů nad daty uloženými v databázi

Systém řízení báze dat (dále SŘBD) je software sloužící k přístupu a správě uložených dat v databázi. Umí efektivně pracovat s velkým množstvím dat a také umí vhodně definovat jejich strukturu pro uložení. SŘBD umožňuje uživatelům komunikovat s jednou nebo více databázemi a přistupovat k datům, které obsahují. Funkce, které SŘBD poskytuje, lze rozdělit do čtyř skupin:

- Definice dat – vytváření, modifikace a mazání definic definujících organizaci dat.
- Aktualizace dat – vkládání, modifikace a mazaní samotných dat.
- Vyhledávání – poskytnutí dat z databáze ve formě přímo použitelné nebo ve formě k dalšímu zpracování. Získaná data mohou mít stejnou strukturu jako data uložená v databázi nebo mohou být pozměněna v závislosti na kombinaci těchto dat.
- Správa – registrace a monitorování uživatelů, zabezpečení, integrita dat, sledování výkonu, obnovení poškozených informací.

V souvislosti s databází existují důležité pojmy, které je třeba znát. Datové objekty jsou objekty databáze (tabulky, indexy, pohledy, ...) a záznam je jeden řádek tabulky.

4.1 Relační model dat

Relační model je nejrozšířenější model databáze pro ukládání dat. Data v relační databázi jsou strukturována do tabulek (relací). Strukturu tabulky můžeme rozdělit na záhlaví (schéma relace) a tělo. Vlastnosti, které chceme o objektu uložit do databáze, se u relačního modelu nazývají atributy – jedním atributem se míní jeden sloupec tabulky. Každý atribut může nabývat hodnot z množiny přípustných hodnot definovaných pro daný sloupec. Takové množiny atributů se nazývají domény a mohou obsahovat pouze atomické (skalární) hodnoty. Jeden řádek tabulky je n-tice tvořena atributy nabývající hodnoty. Počet řádků tabulky i jejich obsah se v čase mění. Výhodou relačního modelu je přirozená reprezentace zpracovávaných dat a zpracování vazeb.

4.1.1 Integritní omezení

Integrita databáze zajišťuje, aby do databáze mohly být uloženy pouze data, která odpovídají určitým pravidlům. K zajištění integrity se používají integritní omezení, která jsou součástí definice databáze. Jsou to pravidla, která zajišťují, že databáze bude konzistentní – zábrání vložení nesprávných dat, poškození či ztrátě záznamů a další. V rámci integrity dat existují důležité pojmy k pochopení integritních omezení.

Kandidátní klíč

Kandidátní klíč označuje sloupec či sloupce tabulky, jejichž hodnoty jsou v rámci tabulky unikátní. Musí splňovat dvě vlastnosti – hodnota kandidátního klíče v relaci (tabulce) je unikátní (neexistují žádné dvě n-tice relace se stejnou hodnotou tohoto atributu) a atribut kandidátní klíč musí být minimální (je-li kandidátní klíč složeným atributem, nelze z něj vypustit žádnou složku, aniž by přestala platit unikátnost hodnot). Podle kandidátních klíčů je možné jednoznačně idetifikovat každý řádek tabulky. Jeden z kandidátních klíčů slouží jako primární klíč, ostatní kandidátní klíče se označují jako alternativní klíče.

Primární klíč

Primární klíč je jeden z kandidátních klíčů. Je to atribut jenž má unikátní hodnotu pro každý záznam v relaci (tabulce). Pokud existuje pouze jeden kandidátní klíč, je také primárním klíčem. Hodnota primárního klíče musí být vždy zadaná.

Cizí klíč

Cizí klíč slouží pro vytváření vztahů mezi tabulkami databáze. Ukazuje, které záznamy v databázi spolu souvisí. Hodnota cizího klíče může být buď nezadaná nebo zadaná. Cizí klíč může být složený a v takovém případě pro jeho platnost musí být plně zadaný nebo plně nezadaný. Pokud je hodnota cizího klíče zadaná musí se shodovat s hodnotou některého sloupce, kterým je kandidátním klíčem v odkazované tabulce.

Existují čtyři druhy integritních omezení. První pravidlo je pravidlo integrity entit. Toto pravidlo říká, že u žádné složky primárního klíče nesmí chybět hodnota. Referenční integrita se zaměřuje na vztahy relačně propojených tabulek v databázi, kde jejich vztah je určen cizím a primárním klíčem. V případě, že tabulka obsahuje sloupec s cizím klíčem odkazujícím se na záznam v jiné tabulce, musí odkazovaný záznam existovat. Doménová integrita zajišťuje dodržení datových typů definovaných u sloupců tabulek v databázi. Aktivní referenční integrita definuje činnosti, které se provedou v případě, že budou porušena některá integritní omezení.

Pokud data v databázi splňují všechna integritní omezení, je databáze konzistentní.

4.1.2 Vztahy mezi databázovými tabulkami

Vztahy mezi tabulkami jsou vazby mezi daty, které spolu souvisí. Vztah bývá definován třemi vlastnostmi.

Stupeň vztahu:

- unární – relace je spojena sama se sebou (např. vztah zaměstnance a nadřízeného, nadřízený je také zaměstnanec)
- binární – vztah mezi dvěmi relacemi
- ternární – vztah mezi třemi relacemi zároveň
- N-ární – vztah mezi N relacemi zároveň

Kardinalita vztahu:

- Žádná vazba – tabulka nemá žádné vazby k jiným tabulkám
- 1:1 – tato vazba se používá v případě, že jednomu záznamu v první tabulce odpovída pouze jeden záznam v druhé tabulce, tato vazba se moc často nevyskytuje a v případě výskytu je vhodné zvážit, zda tyto záznamy nesloučit do jedné databázové tabulky
- 1:N – tato vazba se používá v případě, kdy jednomu záznamu z jedné tabulky odpovídá více záznamů z druhé tabulky, jedná se o nejvíce používanou vazbu
- M:N – tato vazba se používá v případě, kdy lze více záznamům z jedné tabulky přiřadit více záznamů z druhé tabulky; kvůli praktičnosti se tento vztah realizuje pomocí dvou vztahů – 1:N a 1:M, kdy tyto vztahy ukazují do pomocné tabulky, která obsahuje dva cizí klíče odkazující se na výchozí tabulky (M,N)

Volitelná účast učuje povinnost či nepovinnost účasti relace ve vztahu.

4.1.3 Normální formy

Normalizace je proces, při kterém zjednodušujeme strukturu databázových tabulek za účelem snazší manipulace s daty, zábránění redundanci a lepší konzistenci dat. Správnost navržených struktur lze ohodnotit pomocí normálních forem, což jsou pravidla, která by měla data v tabulce splňovat. Čím vyšší normální forma, tím by práce s našimi daty měla být snazší.

Nultá normální forma (0NF)

Když je tabulka v nulté normální formě obsahuje alespoň jeden sloupec (atribut), který obsahuje více druhů hodnot. Tento sloupec se dá dělit (neobsahuje atomická data). Tato forma se kříží s pravidlem relačního modelu – data v jednom sloupci nesmí být dále dělitelná.

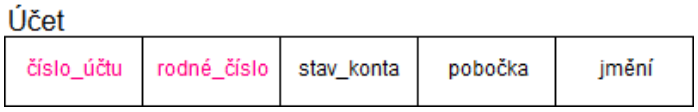
Definice 1 *Relace je v nulté normální formě, existuje-li takové její pole, které obsahuje více než jednu hodnotu. Není-li relace v nulté normální formě, je alespoň v první normální formě.*

Příkladem může být jméno a přijmení zadané v jednom sloupci, kdy nemůžeme vyhledávat podle přijmení, neboť sloupec obsahuje i jméno.

První normální forma (1NF)

Tabulka je v první normální formě pokud všechny sloupce tabulky obsahují pouze atomická data, jeden sloupec neobsahuje složené hodnoty.

Definice 2 *Relace je v první normální formě, právě když všechny její jednoduché domény obsahují pouze atomická data.*

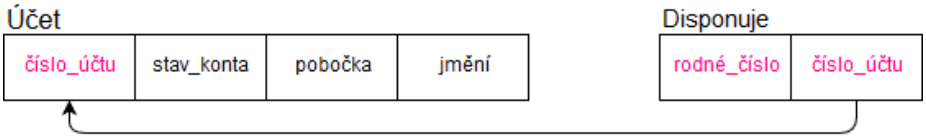


Obrázek 4.1: Tabulka v první normální formě

Druhá normální forma (2NF)

Tabulka je v druhé normální formě, když obsahuje pouze data plně funkčně závislá na kandidátním klíči.

Definice 3 *Schéma relace je ve druhé normální formě, právě když je v 1NF a každý její neklíčový atribut, je plně funkčně závislý na každém kandidátním klíči.*

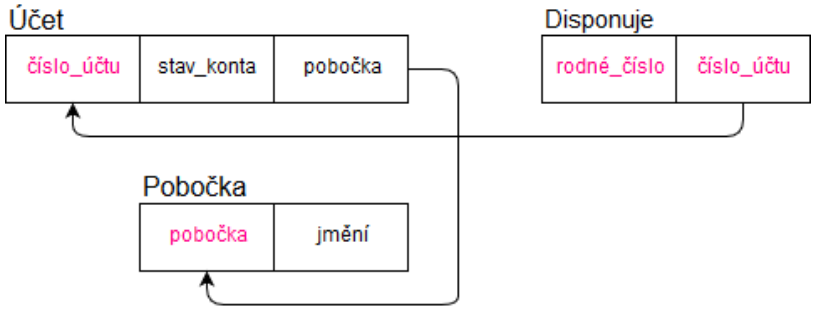


Obrázek 4.2: Tabulka v druhé normální formě

Třetí normální forma (3NF)

Tabulka je ve třetí normální formě, pokud neexistují žádné funkční závisloti mezi neklíčovými atributy.

Definice 4 *Schéma relace je ve třetí normální formě, právě když je v 2NF a neexistuje žádný neklíčový atribut, který je tranzitivně závislý na některém kandidátním klíči.*



Obrázek 4.3: Tabulka v třetí normální formě

Boyce-Coddova normální forma (BCNF)

Každá relace v Boyce-Coddově normální formě, je také ve třetí normální formě. Opačně to však nemusí platit – pokud relace obsahuje více kandidátních klíčů, které mohou být složené nebo které se mohou překrývat. Tato normální forma řeší některé speciální situace, které mohou nastat, a také zajišťuje úplné odstranění redundance. Dosažení BCNF není vždy možné kvůli splnění požadavku na zachování funkčních závislostí. V případě, kdy tahle situace nastane, je třeba se rozhdout, zda upřednostníme snadnější kontrolu integritních omezení plynoucích z funkčních závislostí nebo úplné odstranění redundance.

Definice 5 *Schéma relace R je v Boyce-Codově normální formě, právě když pro každou netriviální funkční závislost X -> Y je X superklíčem. Superklíčem rozumíme každou nadmnožinu kandidátního klíče relace R.*

Čtvrtá normální forma (4NF)

Čtvrtá normální forma udává, že atributy v tabulce popisují pouze jeden fakt či souvislost.

Pátá normální forma (5NF)

Při páté normální formě se jedná o vlastnost takovou, že přidáním jednoho atributu do tabulky, by se tabulka rozpadla na více tabulek. [2][17][13] [22]

Kapitola 5

Popis a návrh systému

Informační systém pro pečovatelskou organizaci je inspirován existujícím informačním systémem pečovatelské organizace Charita Veselí nad Moravou, která eviduje údaje o provedené péči, údaje o klientech, zaměstnancích a také údaje o odpracovaných hodinách zaměstnanců pro účetní, která měsíčně zpracovává platy. Pečovatelská organizace provadí péči až třikrát denně na domech pro seniory, které spadají pod tuto oragnanizaci, a také přímo v domovech klientů. Ke klientům se dochází v intervalech navolených samotným klientem či jeho rodinou. Konkrétní potřebnou péči pro klienta doporučuje sociální pracovnice dle klientova zdravotního stavu a jeho schopností se o sebe postarat. V systému jsou evidování i klienti, kteří od pečovatelské organizace odebírají pouze službu rozvoz obědů do domu.

Existujícímu informačnímu systému pečovatelské organizace chybí některé podstatné funkce pro efektivní práci – zobrazení a správa rozvrhů, tisk výkazů, rozvrhů a přehledu odpracovaných hodin, zobrazení přehledu úhrad za péči a odpracovaných hodin pečovatelů. Vyvíjený informační systém pro pečovatelskou organizaci obsahuje funkce existujícího informačního systému a také výše zmíněné chybějící funkce pro zefektivnění správy pečovatelské organizace.

5.1 Analýza funkčních požadavků na systém

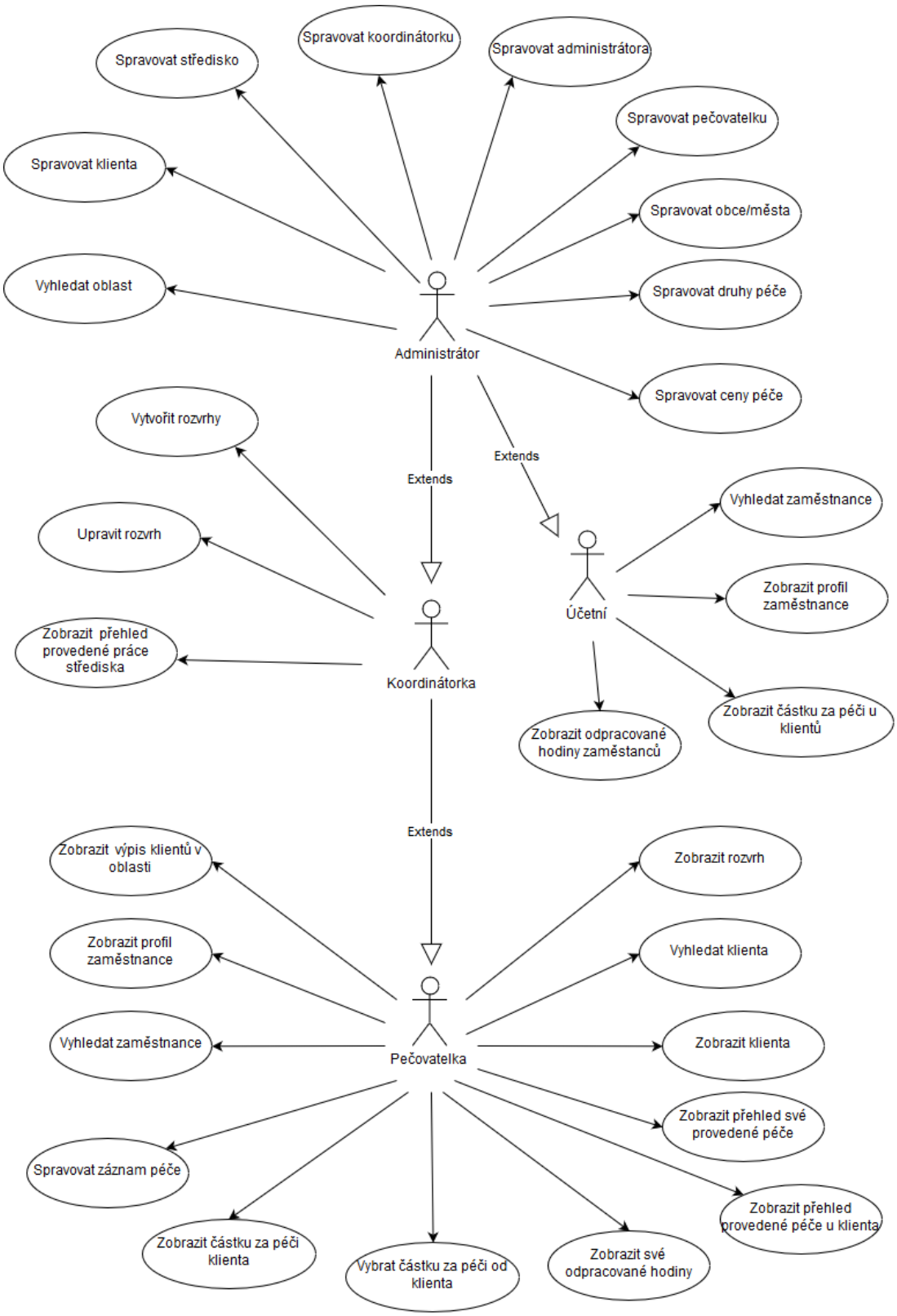
Pečovatelská organizace je rozdělena na správní oblasti – střediska. Každé středisko má jednu koordinátorku a několik pečovatelek. Koordinátorka a pečovatelky z jednoho střediska nemohou zadávat péči klientům z jiného střediska. Počet zaměstnaných pečovatelek na jednom středisku se odvijí od množství klientů v této oblasti. Každé středisko má na starost klienty z několika okolních obcí (případně klienty pouze z jednoho města), ve kterých provádí potřebnou péči u klientů doma či na domech pro seniory, které pod pečovatelskou organizaci spadají. Částku za provedenou péči uhrazují klienti na účet pečovatelské organizaci nebo v hotovosti do rukou pečovatelkám nebo koordinátorkám v dané oblasti, kde je prováděna péče o tohoto klienta. Celá pečovatelská organizace zaměstnává jednu účetní, která zpracovává odpracované hodiny zaměstanců a zodpovídá za vybrání částky za provedenou péči od klientů, a jednoho administrátora, který spravuje veškeré údaje potřebné k evidenci klientů, zaměstnanců a prováděné péči. V systému se tedy vyskytují čtyři role:

- Administrátor – má na starost správu systému, do které spadá přidávání, odebírání či úprava středisek, druhů péče poskytovaných pečovatelskou organizací, klientů, zaměstnanců a ostatních věcí potřebných k evienci výše zmíněných osob v systému a

péče. Každou oblast má na starost koordinátorka, kterou administrátor přidává ke konkrétnímu středisku.

- Pečovatel/ka – zadává do systému údaje o provedené péči u klienta a její zbývající denní pracovní činnosti. Může si zobrazit a vytisknout přehled své provedené péče a může zadat převzetí peněžní částky od klienta za provedenou měsíční péči, má přístup ke svému rozvrhu.
- Koordinátor/ka – provádí stejné činnosti jako pečovatelka. Také si může zobrazit a vytisknout přehled své provedené péče, navíc může zobrazit přehled odvedené práce ostatních pečovatelek ve středisku spadajícím pod ni. Dále vytváří rozvrhy pro pečovatelky, kontroluje celkovou částku vybranou za měsíční péči od klientů spadající pod její středisko a může měnit individuální péči u klientů.
- Účetní – má přístup k odpracovaným hodinám zaměstnanců v jednotlivých oblastech a podle nich zpracovává mzdy. Dále má přístup k částkám za provedenou péči u klientů v jednotlivých oblastech za celý měsíc.

5.2 Diagram případů užití

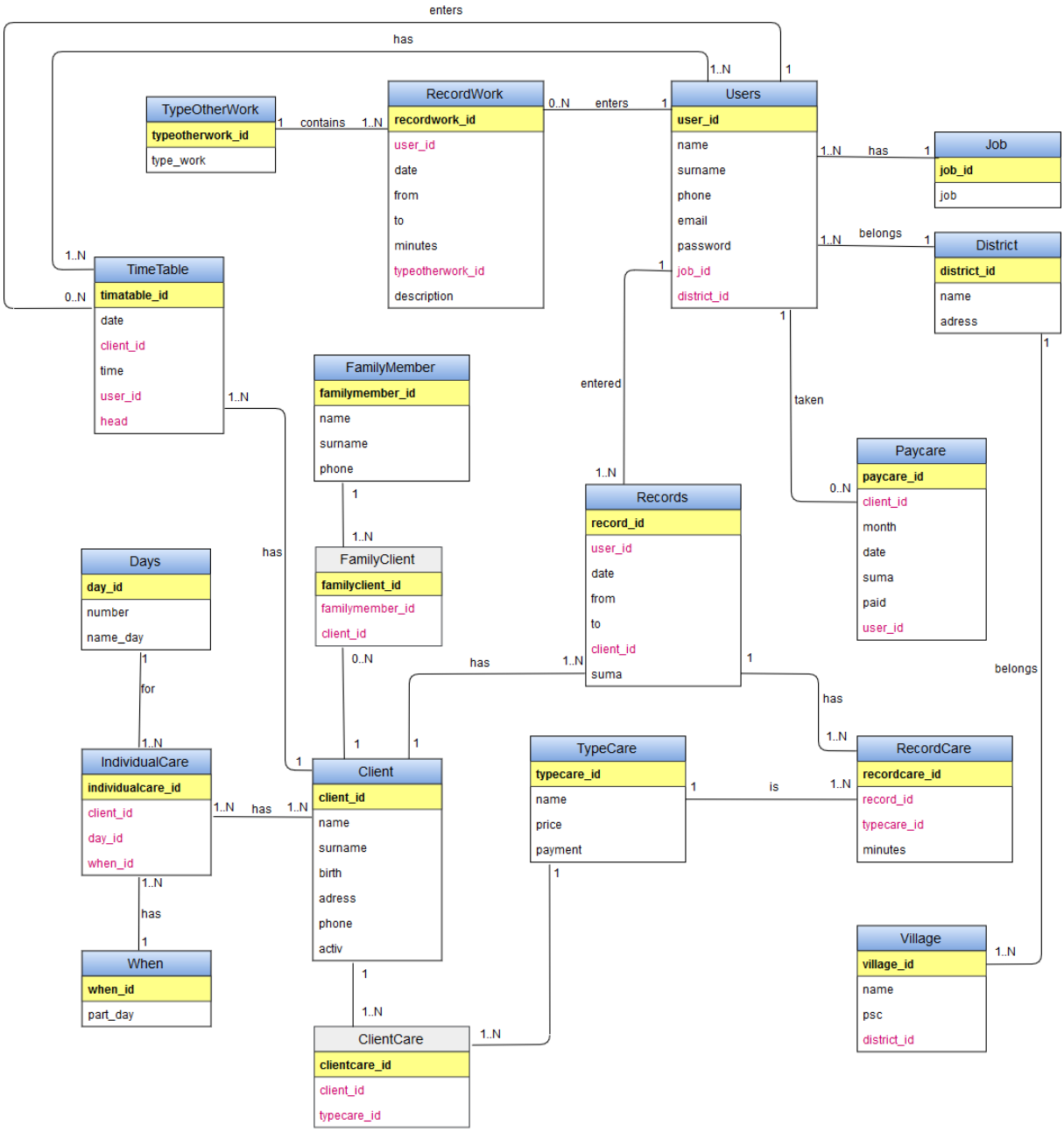


5.3 Návrh databáze

V databázi je třeba evidovat údaje o střediscích, vesnicích (městech), pečovatelkách/koordinátorkách, administrátorovi, účetní, klientech a o provedené péči a ostatních činostech, které pečovatelky provádí. U jednotlivých objektech je třeba evidovat následující údaje:

- Středisko – název střediska, adresa, poštovní směrovací číslo, vesnice (město), ve které se středisko nachází a koordinátora pro dané středisko.
- Vesnice (město) – název, poštovní směrovací číslo, středisko, pod které vesnice spadá.
- Zaměstnanec – osobní a kontaktní údaje, pracovní pozice, určení střediska, pod které zaměstnanec spadá, přihlašovací údaje do systému.
- Klient – osobní údaje, kontaktní údaje na rodinu, individuální péči pro klienta, celkovou částku za provedenou péči za měsíc.
- Péče – komu byla provedena, kdo ji provedl, kdy ji provedl, co zahrnovala poskytnutá péče, jak dlouho se pečovatelská činnost prováděla, jaká je částka za provedenou péči.
- Ostatní činnosti – kdo ji provedl, kdy ji provedl, co to bylo za činnost, jak dlouho ji pečovatel/koordinátor prováděl.

5.4 Schéma databáze



První schéma databáze obsahovalo třináct tabulek. V počátcích práce na implementaci logiky informačního systému bylo třeba schéma databáze upravit. Byly přidány tabulky, které obsahují data pro evidování menších položek a do některých tabulek byly přidaný další potřebné atributy.

Kapitola 6

Implementace

Tato kapitola se zabývá implementací informačního systému dle návrhu z předešlé kapitoly. Informační systém pro pečovatelskou organizaci je dostupný na webové adrese http://karakaterina.cz/. Přihlašovací údaje pod jednotlivými pracovními pozicemi lze nalézt v tabulce níže.

6.1 Použité technologie a nástroje

Informační systém má formát webové aplikace, což sebou přináší velkou výhodu. Webová aplikace se nemusí instalovat na lokální počítač a je uživateli dostupná z každého zařízení, které má přístup k internetu a má nainstalovaný webový prohlížeč. Tato vlastnost značí, že je webová aplikace multiplatformní. Pečovatelská organizace informační systém používá převážně na stolním počítači či notebooku, časem snad i na mobilních zařízení jako je tablet. Vytvořený informační systém má uživatelské rozhraní, které se přizpůsobuje velikosti zobrazovací plochy na zařízení, na kterém je informační systém používán – mobil, tablet, notebook, počítač.

Informační systém pro pečovatelskou organizaci je implementován za použití frameworku Laravel, který používá scriptovací jazyk PHP (přesněji objektové PHP). K použití tohoto frameworku bylo využito lokálního vývojového prostředí Homestead, které poskytuje veškeré systémové požadavky pro Laravel. Rámec Laravel je založený na architektonickém vzoru MVC, u kterého je aplikační logika oddělena od zobrazovací. Jako databázový server byl použit server MySQL, který je rozšířený a dostupný téměř na každém webovém hostingu.

Vzhled informačního systému je tvořen šablonou AdminLTE, což je volně dostupný administrativní dashboard obsahující kontrolní panel. AdminLTE využívá dva aplikační rámce – rámec Boostrap jazyka CSS a rámec jQuery pro jazyk Javascript. Šablona obsahuje vlastní

Role	Přihlašovací email	Přihlašovací heslo
Administrátor	admin@mail.com	123456
Účetní	ucetni@mail.com	123456
Koordinátor	p1@mail.com	123456
Pečovatel	p2@mail.com	123456

Tabulka 6.1: Tabulka přhlašovacích údajů

CSS styly, které lze aplikovat na jednotlivé HTML komponenty jako jsou formuláře, tabulky a další. Jazyk JavaScript je v informačním systému použit na vyhledávání klientů a zaměstnanců – Livesearch, a také na zobrazení kalendáře – datepicker.

6.2 Databázové tabulky

Schéma jednotlivých databázových tabulek informačního systému bylo vytvořeno pomocí nástroje frameworku Laravel `Migration`, což je nástroj, který slouží k jednoduché práci s databází a také pomáhá ke snadné synchronizaci databáze (především v týmech) bez složitých mechanizmů. Příkazem `php artisan make:migration create_myTable_table` se vytvoří soubor, který obsahuje třídu `CreateDistrictTable`. Tato třída obsahuje dvě funkce – `up()` a `down()`. Funkce `up()` vytvoří schéma tabulky a funkce `down()` umožní její smazání.

Po vytvoření migračních souborů se příkazem `php artisan migrate` tabulky vytvoří v databázi, která je specifikovaná v souboru `.env`.

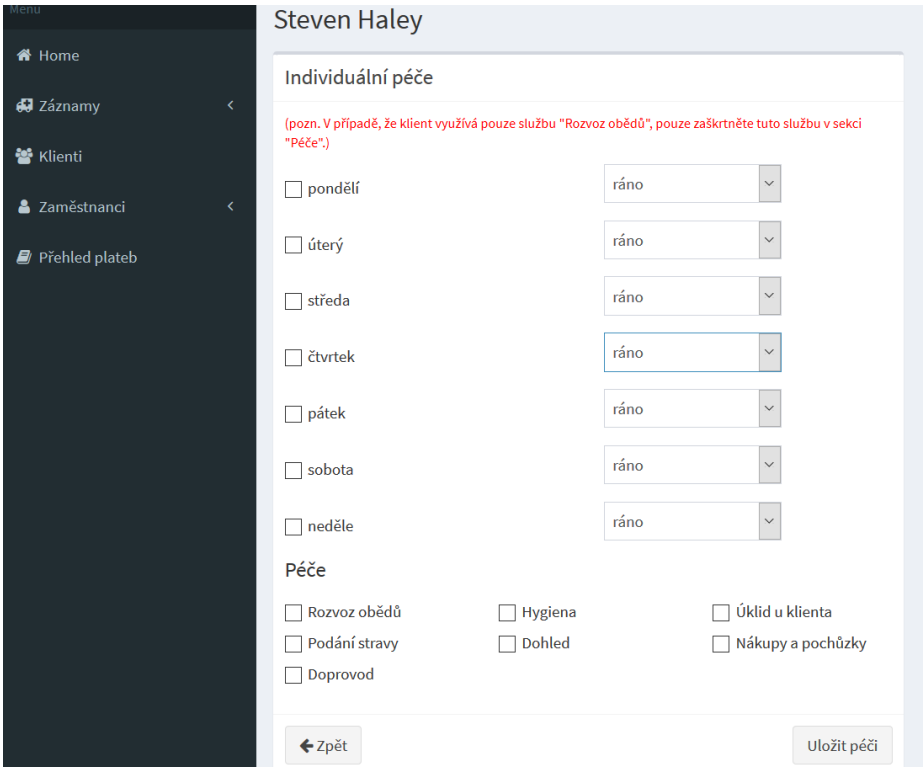
6.3 Logika informačního systému

Veškera logika informačního systému se nachází v kontrolerech, kterých je v tomto projektu celkem jedenáct. Čtyři z nich byly vytvořeny příkazem `php artisan make:auth`, který vytvořil přihlašovací rozhraní informačního systému spolu s registrací, přihlášováním a odhlášováním uživatele. Ostatní kontolery byly vytvořeny příkazem `php artisan make:controller myConstroller`. Do všech kontrolerů, kromě autentizačních kontrolerů, má přístup pouze přihlášený uživatel. Každý z kontrolerů pracuje s určitými daty. V kontroleru `HomeController` se nachází funkce pro správu (přidávání, upravování, mazání) středisek, vesnic/měst, pracovních pozic, druhů péče a ostatních činností pečovatelek. Kontroler `FamilymemberController` slouží pro správu členů rodiny klientů. Pracovat s těmito dvěma kontrolery smí pouze administrátor (admin) neboť se jedná o informace, kterou jsou využívány ostatními uživateli. `ClientController` obsahuje funkce pro správu údajů o klientovi. Do tohoto kontoleru spadá i definování individuální péče pro klienta a zobrazení jeho aktuální částky za provedenou péči. U individuální péče klienta se zadávají dny, ve kterých má být péče u klienta prováděna, části dne neboli kolikrát denně má návštěva u klienta proběhnout a druhy péče, které klient potřebuje. Údaje o tom, které dny a kolikrát denně se má klient navštívit, jsou využity k usnadnění sestavení rozvrhů koordinátorkou/rem. Ukázka rozhraní pro zadávání individuální péče klientovi se nachází na obrázku 6.1

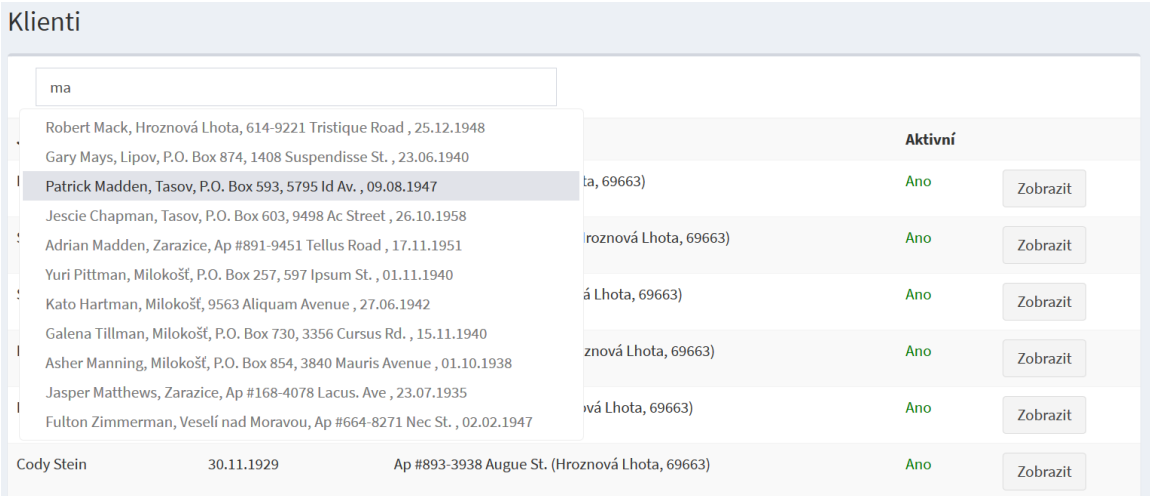
Dalším kontrolerem je `UserController`. Tento kontroler spravuje údaje o uživateli. Jsou zde implementovány funkce na zobrazení osobních údajů uživatele, zobrazení záznamů uživatele a přehled odpracovaných hodin. Funkce `user_records()` zobrazuje záznamy uživatele za vybraný měsíc, funkce `all_hours()` zobrazuje odpracované hodiny zaměstnanců a dvě funkce pro převod HTML stránky do PDF dokumentu. Funkce `tisk_rec()` převádí HTML stránku se záznamy provedené péče a ostatních činností uživatele do PDF dokumentu, který slouží jako výkaz práce uživatele, a funkce `tisk_hours()`, která převede HTML stránku s počtem odpracovaných hodin jednotlivých zaměstnanců do PDF dokumentu obsahující tyto údaje. Pro převod je použita knihovna TCPDF, kde jsou nastaveny patřičné atributy pro získání požadovaného výstupu.

Oba zmíněné kontrolery, `UserController` a `ClientController`, obsahují navíc funkci `fetch()`, která získá výsledky vyhledávání uživatelů či klientů a tyto výsledky jsou pro-

střednictvím JavaScriptu sprostředkovány do komponenty, která vysledky zobrazuje a mění v reálném čase podle zadaného klíče. Tato metoda je známá jako Livesearch. Ukázka metody je na obrázku 6.2



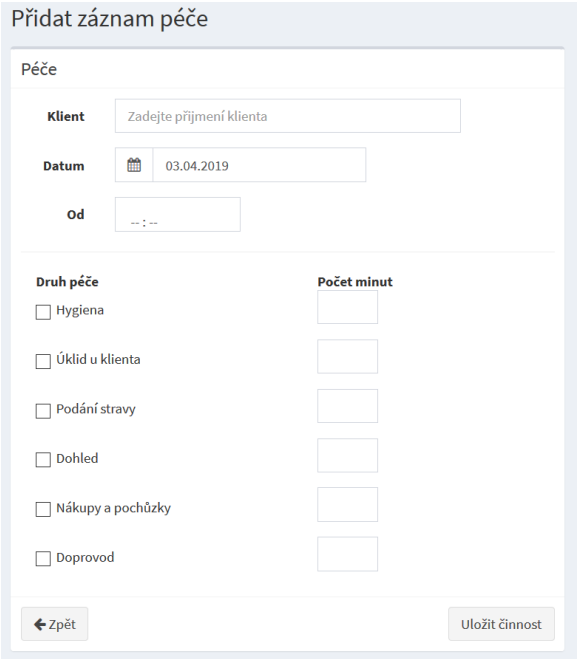
Obrázek 6.1: Rozhraní pro zadávání individuální péče klientovi



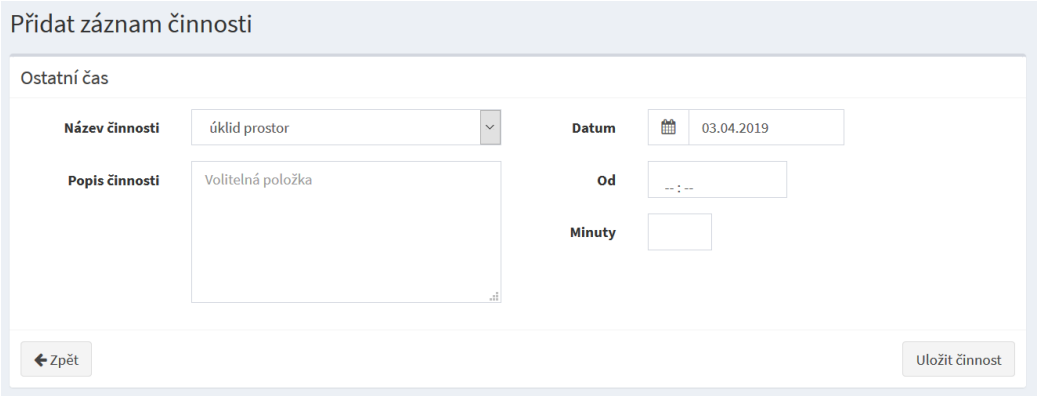
Obrázek 6.2: Livesearch

V kontroleru `RecordController` jsou implementovány funkce pro zadávání a správu záznamů. Uživatelé vytváří záznamy dvojího druhu – záznamy péče o klienty a záznamy ostatních činností pečovatelů a koordinátorů. Při zadání péče provedené u klienta se vytvoří

záznam o provedené péči, který obsahuje i částku za provedené úkony. Do ostatních činností patří úklid společných prostor, přejezdy mezi klienty, administrativa, schůze, školení a rozvoz obědů. Činnost rozvoz obědů je brána jako činnost, kterou provádí zaměstnanec po určitou dobu. Rozváží obědy klientům. Po uložení záznamu tohoto typu se zobrazí seznam klientů, kteří mají tuto péči objednanou. Zadavatel vybere klienty, kterým doručil oběd a ke každému takovému klientovi se vytvoří záznam s částkou za tuto péči, která se přičítá k ostatním částkám uloženým v ostatních záznamech tohoto klienta. Ukázka rozhraní pro zadávání péče je na obrázku 6.3 a pro zadávání ostatních činností na obrázku 6.4.



Obrázek 6.3: Rozhraní pro zadávání péče



Obrázek 6.4: Rozhraní pro zadávání ostatních činností

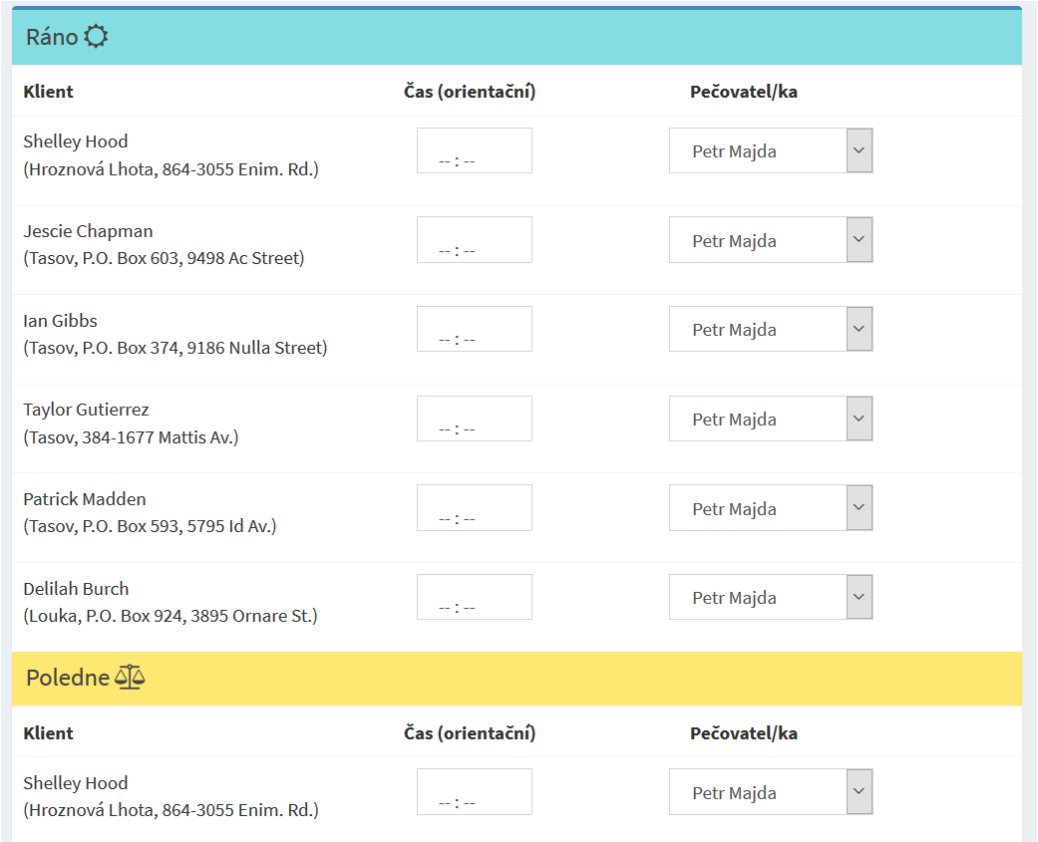
Posledním kontrolerem je `TimeTableController`, který slouží pro vytváření a správu rozvrhů pro pečovatele a koordinátory. Prvním krokem sestavení rozvrhu pro den je vybrat datum. Zvolený datum je vstupem do funkce `start()`. V této funkci se nejprve pomocí

funkce `dayOfWeek`[1] zjistí číslo dne v týdnu. Následně pomocí tří dotazů na databázi získáme tři pole – ráno (`$r`), poledne (`$p`) a večer (`$v`). Každé z těchto polí obsahuje klienty, kteří mají objednanou péči v daný den a část tohoto dne. Koordinátorovi se poté vykreslí pohled, ze kterého lze vyčíst, kteří klienti mají objednanou péči na danou část dne. Koordinátor nasledně k těmto klientům uvede orientační čas péče a vybere pečovatele, který u nich péči provede. Data se následně uloží do tabulky `timetable`.

Podle existujícího zavedeného systému se na jednu pečovatelku počítá odhadem pět klientů na jednu část dne (ráno, poledne či večer), aby stihla obstarat klienty a ještě ostatní činnosti, které má dané. Na každého klienta se orietačně počítá 30 minut na provedení péče.

Pro správné vykreslení rozvrhu je třeba do tabulky uložit tyto údaje:

- id klienta
- datum péče o klienta
- čas
- id pečovatele



Obrázek 6.5: Rozhraní pro vytváření rozvrhu

Po kliknutí na klienta zobrazeného v rozvrhu se zobrazí tabulka s detaily plánu péče – funkce `show_client_plan($plan_id)`. Koordinátor tento plán může smazat či změnit čas péče a pečovatele, který má tuto péči provést.

[1] `dayOfWeek` je funkce balíčku `Carbon`, který rozšiřuje třídu `DateTime`. Tento balíček obsahuje spoustu funkcí, které usnadňují práci s datem a časem.

6.4 Optimalizace rozvrhů

Po vytvoření rozvrhu koordinátorkou se může stát, že přejezdy mezi klienty jsou neefektivní z důvodu zbytečných přejezdů. Problém, který může nastat, je, že pečovatel je u prvního klienta ve vesnici A, pak jede k druhému klientovi do vesnice B a poté ke třetímu klientovi opět do vesnice A. Aby se zabránilo těmto zbytečným přejezdům, je třeba rozvrh přeskládat. Jednou z variant je seskupit vesnice podle poštovního směrovacího čísla (dálé psč). Některé vesnice mají stejné psč, což mimo jiné značí, že tyhle vesnice by si měly být blíže než s jinou vesnicí. Podle toho se dá usoudit, že pokud existují klienti, kteří potřebují péči ve vesnici A a taky klienti ve vesnici B a C, z toho vesnice A a B mají stejné psč, je vhodnější pro pečovatele obstarat péči u klientů A a B a následně jet za klienty do vesnice C. Pro optimalizování rozvrhu je vhodné z tabulky `timetable` získat plány všech klientů na tento den, rozčlenit je do tří polí (klienti, kteří mají naplánovanou péči na ráno, poledne a večer) a seřadit nejprve podle psč, poté názvu vesnice a násdledně času. Tak získáme tři pole, se kterými se bude dobře pracovat. Následně každé z těchto polí necháme zpracovat funkcí `optimize()`, která klienty vhodně přeskládá. Funkce `optimize()` má tři argumenty – `$arr` zpracovávané pole, `$users` pole pečovatelů, `$t_start` počáteční čas pro péči. Algoritmus prochází vstupní pole třikrát ve třech cyklech. V prvním cyklu se přiřadí jeden a ten samý pečovatel všem klientům, kteří jsou ze stejné vesnice nebo jejichž vesnice má stejné psč jako vesnice předchozí. Když se objeví vesnice neshodná s předchozí nebo s neshodným psč jako je předchozí, tak se tomuto klientovi z této vesnice přiřadí jiný pečovatel, kterému se opět přiřadí další klienti podle předešlých podmínek. V druhém cyklu se pomocí pomocného pole `$new[]` kontroluje počet klientů, které má na starost jeden pečovatel. V poli `$new[]` je uložen aktuální počet klientů pro každého pečovatele. Na počátku je všem pečovatelům přiřazen počet nula. V prvním průchodu druhého cyklu se do pole `$new[]` na pozici pečovatelova id, které se nachází v prvním záznamu vstupního pole, uloží počet 1 a do pomocné proměnné `$prev` se přiřadí tento celý záznam, který slouží na porovnávání jako záznam předchozí se záznamem aktuálním. V každém průchodu se proměnná `$prev` mění, po provedení všech operací v průchodu se do ní přiřadí aktuální záznam.

V ostatních průchodech tohoto cyklu se postupně upravují hodnoty v poli `$new` značící počet klientů na pečovatele. Pokud počet dosáhne čísla pět, nejprve se zjistí jestli se v poli `$new[]` nenechází uživatel, který má počet klientů větší než nula a zároveň je tento počet menší než 5. Pokud existuje takový pečovatel, je mu přiřazen tento klient. V případě, že jsme takového pečovatele nenašli, přiřadíme tohoto klienta prvnímu pečovateli s počtem nula.

V posledním cyklu dochází ke korekci času, aby na sebe jednotliví klienti navazovali. Počáteční čas u prvního klienta každého pečovatele se nastaví na hodnotu argumentu `$t_start`. Tato hodnota se liší v závislosti na části dne (ráno, poledne, večer). S každým dalším klientem pečovatele se čas posunuje o 30 minut. Nakonec se aktualizují záznamy v tabulce `timetable`, aby jsme optimalizovaný rozvrh mohli zobrazit.

6.5 Zabezpečení informačního systému

Velmi podstatnou součástí infomačního systému je bezpečnost. V informačním systému jsou uloženy citlivé údaje o klientech a zaměstnancích, které by potenciální útočních mohl získat a zneužít. Útočník by se tak mohl například dostat k osobním či kontaktním údajům klientů pečovatelké služby.

Mimo stránky na přihlášení a stránky na obnovu hesla, jsou ostatní stránky informač-

ního systému pro pečovateslkou oraganizaci zabezpečeny takzvaným middleware, což je prostředek, který filtruje HTTP požadavky zasílané uživatelem na server. Pokud HTTP požadavek neodpovídá danému pravidlu pro danou stránku, je tento požadavek odmítnut a zpracován jiným způsobem než kdyby byl přijat. Middleware zabezpečuje každou ze stránek informačního systému proti vstupu neověřeného uživatele. Toto ověření se provádí v konstruktoru každého z kontrolerů informačního systému. V případě, že uživatel není přihlášen nebo vyprší doba, po kterou přihlášený uživatel je neaktivní, je tento uživatel přesměrován na přihlašovací stránku. Kód konstruktoru lze vidět níže:

```
public function __construct()
{
    $this->middleware('auth');
}
```

Součástí zabezpečení informačního systému je i CSRF ochrana. Tato ochrana slouží prosti CSRF (cross-site request forgery) útokům, což jsou falešné žádosti na aplikaci, při nichž jsou neautorizované žádosti prováděny pod ověřeným uživatelem. Laravel automaticky generuje CSRF token, který slouží k ověření, že žádost o provedení nějaké akce, podal opravdu ověřený uživatel a ne někdo jiný. Implementace CSRF ochrany je součástí middlewaru.

Ověření, zda má uživatel s danou pracovní pozicí oprávnění vstoupit na danou stránku je prováděno klasickým způsobem bez použití rolí.

Poslední zabezpečení informačního systému se zabývá zabezpečením komunikace s MySQL serverem, které je zajištěno použitím takzvaných prepared statements.

K přihlášení do informačního systému je třeba zadat uživatelský email a heslo. Heslo se do databáze ukláda zakódované pomocí algoritmu bcrypt.

Kapitola 7

Testování a zpětná vazba

Testování je poslední fází při vytváření libovolného projektu ačkoliv u každého projektu vypadá jinak. Po vypracování informačního systému je vhodné otestovat, to co bylo implementováno, abychom konečnému zákazníkovi poskytli plně funkční a bezchybný systém, který tak může plnit svůj účel.

Informační systém pro pečovateskou organizaci obsahuje spoustu praktických funkcí pro správu a zadávání péče. Ačkoli se může zdát, že funguje vše správně, nemusí tomu tak být. Pro otestování funkčnosti informačního systému byli využiti zaměstnanci pečovatelské organizace Charita Veselí nad Moravou, podle jejichž informačního systému byl sestaven informační systém popsaný v této práci. Zaměstnanci byli nejprve provedeni informačním systémem a následně dostali úkoly, které měli v informačním systému provést.

7.1 Průvod informačním systémem

Po přihlášení do informačního systému uživatel vidí druh úvodní stránky dle jeho pracovní pozice. Koordinátor vidí rozvrh klientů s možností přepnutí na vlastní rozvrh, ve kterém má přiřazené klienty, či rozvrh jiného pečovatele ve svém středisku. Také může přistoupit ke správě rozvrhů, kde se nachází samotné tvoření rozvrhu a možnost jeho úpravy. Běžný pečovatel vidí vlastní rozvrh, účetní a admnistrátor vidí menu s úkony, které mohou v informačním systému provádět. Samozřejmě administrátor může provadět i jiné úkony než jsou uvedené v tomto menu.
V informačním systému vpravo nahoře se nachází odkaz na profil prihlášeného uživatele a odhlašovací tlačítko. Na levé straně se nachází hlavní menu obsahující sekci Home (úvodní stránka), zaměstnanci, klienti, dále sekci pro správu záznamů a sekci s přehledem plateb. V sekci záznamů se lze dostat k zadávání záznamů a k jejich přehledu, v sekci zaměstnanců se lze dostat k profilům zaměstnancům a k přehledu odpracovaných hodin. Při zadávání záznamu je třeba zvolit mezi dvěma druhy záznamů – zadat péči a zadat ostatní.

7.2 Testování funkčnosti

Pečovatelům byl předložen lístek s přihlašovacími údaji. Po provedení informačním systémem dostali pečovatelé za úkol vytvořit dva záznamy péče a poté jeden z nich upravit nebo smazat. Při náhledu na detail záznamu se projevila chyba spočívající v nenastavení 24 hodinového pásma – po zadání péče, která začínala ve 12:00 a trvala 60 minut se čas konce péče nastavil na 01:00 místo 13:00. Dalším úkolem bylo zadat záznam ostatní péče

se specifikací na úkon rozvoz obědů, aby pečovatelé věděli, jak se eviduje tato činnost.
Při úkolu pro koordinátora – změnit individuální péči klienta, byla nalezena chyba v oprávnění, která koordinátorovi neumožňovala tuto změnu provést. Dalším úkolem pro koordinátora bylo vytvořit rozvrh pro celé středisko, kde nenastal žádný problém. Při pokusu vytisknout rozvrh na další týden (ne aktuální) se ukázalo, že není možné vytisknout rozvrh pro jiný týden než je aktuální. Pečovatelé i koordinátor si mimo zadané úkoly prošli celý informační systém včetně pokusů o přístup k částem informačního systému, kam nemají mít oprávnění.

Účetní má pouze přístup k přehledům odpracovaných hodin, přehledu plateb a profilům zaměstnancům a klientů, proto nebylo potřeba nějakou funkčnost testovat.

7.3 Zpětná vazba

Informační systém pro pečovatelskou oraganizaci se zaměstanancům líbil, avšak měli určité výhrady.
Menší výhrada byla k tisku výkazů, které by rádi tiskli po týdnech a ne až po měsících. Při zadávání ostatní činnosti typu přejezdy mezi klienty byl vznesen požadavek, aby přejezdy byly zadávány buď hromadně nebo vůbec kvůli pracnosti zadávání – mezi každými dvěmi záznamy provedené péče musí být evidován i přejezd. S tímto požadavkem souvisí počátek a konec pracovní doby. Při absenci zadávání přejezdů mezi klienty, by pro úplnost evidence odpravovaných hodin měl být zadán počátek a konec pracovní doby. Další požadavek byl na zpracování statistik o prováděné péči a následně jejich tisk – četnost provádění určitého druhu péče, kolikrát týdně se ke klientům dochází a další. Tyto požadavky zatím nebyly implementovány do informačního systému.

Kapitola 8

Závěr

Terenní pečovatelská organizace je v dnešní době velmi žádaná a díky její činnosti mohou klienti dostávat potřebnou péči v jejich domovech bez odloučení od rodiny a přátel. Základem pro pečovatelskou oraganizaci je vhodně naplánovat péči klientům k jejich spokojenosti a potřebě a následně ji zaevidovat. Aby pečovatelská organizace mohla efektivně plnit výše zmíněné dva úkony, je vhodné používat nějakou aplikaci, která ulehčí plánování péče a následně umožní efektivě ji zaevidovat. Takovou aplikací může být informační systém popsaný v této práci. Informační systém pro pečovatelskou organizaci byl inspirován existujícím informačním systémem, který používá pečovatelská organizace Charita Veselí nad Moravou. Implementovaný informační systém obsahuje potřebné funkce pro plánování a evidování péče, správu klientů, zaměstnanců a rozvrhů, a funkce pro tisk výkazů, rozvrhů a přehledu odpracovaných hodin.

Informační systém byl implementován pomocí dostupných webových technologií, díky kterým je multiplatformí a lze jej tedy používat na libovolném elektrickém zařízení, které má nainstalovaný webový prohlížeč a má k dispozici internetové připojení. Také má přizpůsobující se uživatelské rozhraní, které se mění v závislosti na velikosti displeje elektrického zařízení, na kterém uživatel informační systém používá.

Informační systém lze dále rozšiřovat o různé funkce. Z fungování pečovatelské organizace plyne, že by bylo vhodné informační systém rozšířit o evidenci nemocí a dovolených zaměstnanců. Tato evidence by ještě více ulehčila plánování péče koordinátorovi. Dále by mohl být rozšířen o databázové triggery, které by se staraly o mazání neaktivních či zemřelých klientů či starých záznamů a tím tak změnšovaly objem dat v databázi.

Informační systém pro pečovatelskou organizaci popsaný v této práci poskytuje praktický nástroj pro správný chod pečovatelské organizace a ulehčuje práci pečovatelům i koordinátorovi při plánování a evidování péče.

Literatura

[1] *CodeIgniter User Guide*. [Online; navštíveno 29.10.2018].
URL https://codeigniter.com/user_guide/

[2] *Databáze*. [Online; navštíveno 17.04.2019].
URL http://www.databaze.chytrak.cz/objekty.htm

[3] *Installation*. [Online; navštíveno 29.10.2018].
URL https://laravel.com/docs/5.7

[4] *Strategy*. [Online; navštíveno 23.11.2018].
URL https://www.algoritmy.net/article/1639/Strategy

[5] *Top 5 : Best open source PDF generation libraries for PHP*. [Online; navštíveno 01.04.2019].
URL https://ourcodeworld.com/articles/read/226/top-5-best-open-source-pdf-generation-libraries-for-php

[6] *Webové stránky a PDF* . [Online; navštíveno 01.04.2019].
URL https://helpx.adobe.com/cz/acrobat/11/using/converting-web-pages-pdf.html

[7] *What is Symfony*. [Online; navštíveno 30.10.2018].
URL https://symfony.com/what-is-symfony

[8] Alex Ivanovs: *Top 32 Free Responsive HTML5 Admin Dashboard Templates 2018*. [Online; navštíveno 18.1.2019].
URL https://colorlib.com/wp/free-html5-admin-dashboard-templates/

[9] Anna Monus: *10 PHP Frameworks For Developers – Best of*. [Online; navštíveno 29.10.2018].
URL https://www.hongkiat.com/blog/best-php-frameworks/

[10] Bernard Kohan: *Guide to Web Application Development*. [Online; navštíveno 30.10.2018].
URL https://www.comentum.com/guide-to-web-application-development.html

[11] Borek Bernard: *Úvod do architektury MVC*. [Online; navštíveno 24.10.2018].
URL https://www.zdrojak.cz/clanky/uvod-do-architektury-mvc/

[12] David Čápka: *MVC architektura*. [Online; navštíveno 24.10.2018].
URL https://www.itnetwork.cz/navrh/mvc-architektura-navrhovy-vzor

[13] doc. Ing. Jaroslav Zendulka, Ing. Ivana Rudolfová: *Databázové systémy IDS*. [Online; navštíveno 19.04.2019].
URL
https://www.fit.vutbr.cz/study/courses/IDS/private/opora/IDS_predn.pdf

[14] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides: *Návrh programů pomocí vzorů*. Grada, 2003, ISBN 8024703025.

[15] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides: *Design Patterns: Elements of Reusable Object-Oriented Software*. Computer Press, 2009, ISBN 0-201-63.

[16] Ilja Kraval: *Analytické modelování informačních systémů pomocí UML v praxi*. Object Consulting, 2010, ISBN 978-80-254-6986-6.

[17] Jakub Kulhan: *Normalizace relačních databází*. [Online; navštíveno 18.04.2019].
URL http://programujte.com/clanek/2008071900-normalizace-relacnich-databazi/

[18] Jan Škrášek, Zdeněk Lehocký: *PHP frameworky*. [Online; navštíveno 23.10.2018].
URL http://programujte.com/clanek/2008022000-php-frameworky/

[19] Judith Bishop: *C 3.0 Design Patterns*. O'Reilly Media, 2007, ISBN 978-0-596-52773-0.

[20] K Hong: *Design Patterns - Model View Controller (MVC) Pattern* . [Online; navštíveno 25.10.2018].
URL https://www.bogotobogo.com/DesignPatterns/mvc_model_view_controller_pattern.php

[21] Margaret Rouse: *framework*. [Online; navštíveno 23.10.2018].
URL https://whatis.techtarget.com/definition/framework

[22] Radoslav Niče: *Relační databáze a dotazovací jazyk SQL*. [Online; navštíveno 17.04.2019].
URL
https://www.distancne.cz/kurz/relacni-databaze-a-dotazovaci-jazyk-sql/

[23] Rudolf Pecinovský: *Návrhové vzory*. Computer Press, 2013, ISBN 978-80-251-1582-4.

[24] Stephanie Reigns: *11 Best PHP Frameworks for Modern Web Developers in 2018*. [Online; navštíveno 24.10.2018].
URL https://coderseye.com/best-php-frameworks-for-web-developers/

Příloha A

Obsah přiloženého CD

- `care_laravel`
- `database`
- `documentation`

Složka `care_laravel` obsahuje zdrojové kódy implementovaného informačního systému. Složka `database` obsahuje SQL scripty pro vytvoření tabulek v databázi a také testovací data. Složka `documentation` obsahuje zdrojové soubory dokumentace v LaTeXu a taky vygenerované PDF.