

**VYSOKÁ ŠKOLA POLYTECHNICKÁ JIHLAVA**

Katedra technických studií

**Rezervační systém pro city.cz**

bakalářská práce

Autor práce: Martin Bulák

Vedoucí práce: PaedDr. František Smrčka, Ph.D.

Jihlava 2018

# Vysoká škola polytechnická Jihlava

Tolstého 16, 586 01 Jihlava

## ZADÁNÍ BAKALÁŘSKÉ/DIPLOMOVÉ PRÁCE

Autor práce: **Martin Bulák**

Studijní program: Elektrotechnika a informatika

Obor: Aplikovaná informatika

Název práce: **Rezervační systém pro city.cz**

Cíl práce: Cílem projektu je vytvoření univerzálního rezervačního systému (Popřípadě úprava rezervačního systému stávajícího), který bude registrovaným firmám poskytovat nástroj pro evidenci rezervací učiněných koncovými zákazníky. Zmíněný koncový zákazník má možnost vytváření nezávazných rezervací bez nutnosti registrace. Tyto rezervace se vytvářejí v předem definovaném formátu stanoveném funkcími projektu. Rezervační systém bude naprogramován v PHP frameworku Symfony a bude využívat moderních nástrojů určených ke tvorbě webových systémů (MySQL, jQuery, Javascript, CSS a další...).

PaedDr. František Smrčka, Ph.D.  
vedoucí bakalářské/diplomové práce

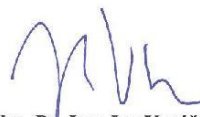
doc. Dr. Ing. Jan Voráček, CSc.  
vedoucí katedry  
Katedra technických studií

## ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Autor práce: **Martin Bulák**  
Studijní program: Elektrotechnika a informatika  
Obor: Aplikovaná informatika  
Název práce: **Rezervační systém pro city.cz**  
Cíl práce: Cílem projektu je vytvoření univerzálního rezervačního systému (Popřípadě úprava rezervačního systému stávajícího), který bude registrovaným firmám poskytovat nástroj pro evidenci rezervací učiněných koncovými zákazníky. Zmíněný koncový zákazník má možnost vytváření nezávazných rezervací bez nutnosti registrace. Tyto rezervace se vytvářejí v předem definovaném formátu stanoveném funkcími projektu. Rezervační systém bude naprogramován v PHP frameworku Symfony a bude využívat moderních nástrojů určených ke tvorbě webových systémů (MySQL, JQuery, Javascript, CSS a další...).



PaedDr. František Smrčka, Ph.D.  
vedoucí bakalářské práce



doc. Dr. Ing. Jan Voráček, CSc.  
vedoucí katedry  
Katedra technických studií

## **Abstrakt**

Je nutné vytvořit rezervační systém, který umožňuje koncovým zákazníkům zakládat rezervace bez nutnosti jejich registrace. Majitelé účtu v rezervačním systému musejí mít možnost tyto rezervace spravovat, ale také mají k dispozici funkce pro nastavení pravidel rezervování. Bakalářská práce je navržena ve frameworku Symfony a pro její návrh jsou využívány moderní technologie pro tvorbu webových stránek.

## **Klíčová slova**

Rezervační systém, PHP framework Symfony, architektura MVC, cross origin resource sharing, cross-site request forgery

## **Abstract**

It is necessary to create a booking system that allows the end customer to make a booking without registration. Booking system account owners must be able to manage these bookings, but also can use functions for setting up booking policies. The bachelor thesis is designed in Symfony framework and it uses modern technologies for website creation for its design.

## **Key words**

Booking system, PHP Symfony framework, MVC architecture, cross origin resource sharing, cross-site request forgery

Prohlašuji, že předložená bakalářská práce je původní a zpracoval/a jsem ji samostatně. Prohlašuji, že citace použitých pramenů je úplná, že jsem v práci neporušil/a autorská práva (ve smyslu zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů, v platném znění, dále též „AZ“).

Souhlasím s umístěním bakalářské práce v knihovně VŠPJ a s jejím užitím k výuce nebo k vlastní vnitřní potřebě VŠPJ.

Byl/a jsem seznámen/a s tím, že na mou bakalářskou práci se plně vztahuje **AZ**, zejména § 60 (školní dílo).

Beru na vědomí, že VŠPJ má právo na uzavření licenční smlouvy o užití mé bakalářské práce a prohlašuji, že **s o u h l a s í m** s případným užitím mé bakalářské práce (prodej, zapůjčení apod.).

Jsem si vědom/a toho, že užít své bakalářské práce či poskytnout licenci k jejímu využití mohu jen se souhlasem VŠPJ, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených vysokou školou na vytvoření díla (až do jejich skutečné výše), z výdělku dosaženého v souvislosti s užitím díla či poskytnutím licence.

V Jihlavě dne 19. dubna 2018

.....  
Podpis studenta/ky

## **Poděkování**

*Rád bych na tomto místě poděkoval mému zadavateli bakalářské práce Martinu Szablaturovi za cenné rady při vývoji a také bych chtěl poděkovat vedoucímu mé bakalářské práce PaedDr. Františku Smrčkovi, Ph.D. za připomínky ke zlepšení formální stránky tohoto dokumentu.*

## Obsah

|  |    |
|--|----|
| Úvod.....  | 13 |
| Motivace .....   | 13 |
| Cíl práce .....  | 14 |
| O firmě .....  | 14 |
| Spolupráce .....   | 14 |
| 1    Představení použitých nástrojů .....                | 15 |
| 1.1    Co je to API rozhraní rezervačního systému? ..... | 15 |
| 1.2    Použité softwarové nástroje .....                 | 15 |
| 1.2.1    Wamp server + virtual host.....                 | 15 |
| 1.2.2    Free commander .....                            | 16 |
| 1.2.3    PHP Storm .....                                 | 16 |
| 1.2.4    Bit bucket.....                                 | 16 |
| 1.2.5    Opera.....                                      | 16 |
| 1.3    Architektura MVC .....                            | 17 |
| 1.4    Symfony framework.....                            | 17 |
| 1.4.1    Představení.....                                | 17 |
| 1.4.2    PHP .....                                       | 18 |
| 1.4.3    Doctrine .....                                  | 18 |
| 1.4.4    Twig .....                                      | 19 |
| 1.4.5    Struktura adresáře .....                        | 19 |
| 1.4.6    Bundle systém.....                              | 20 |
| 1.4.7    Entity.....                                     | 20 |
| 1.4.8    Kontroléry .....                                | 20 |
| 1.4.9    Služby .....                                    | 21 |
| 1.4.10    Konfigurační soubory .....                     | 21 |
| 1.4.11    Voter .....                                    | 22 |
| 1.5    Databázový systém.....                            | 22 |
| 1.5.1    MySQL .....                                     | 22 |
| 1.5.2    PhpMyAdmin.....                                 | 23 |
| 1.6    Správce balíčků .....                             | 23 |
| 1.6.1    Bower components .....                          | 23 |
| 1.6.2    Composer .....                                  | 23 |
| 1.7    Nástroje pro tvorbu vzhledu.....                  | 23 |
| 1.7.1    HTML .....                                      | 24 |
| 1.7.2    CSS .....                                       | 24 |

|       |                                    |    |
|-------|------------------------------------|----|
| 1.7.3 | Bootstrap .....                    | 24 |
| 1.7.4 | Font awesome .....                 | 24 |
| 1.7.5 | Select2.....                       | 24 |
| 1.7.6 | AdminLTE .....                     | 24 |
| 1.8   | Skriptovací nástroje.....          | 25 |
| 1.8.1 | Javascript .....                   | 25 |
| 1.8.2 | jQuery .....                       | 25 |
| 1.8.3 | Ajax.....                          | 26 |
| 1.8.4 | Full calendar .....                | 26 |
| 1.9   | Použité bundly.....                | 27 |
| 1.9.1 | Ivory CK editor .....              | 27 |
| 1.9.2 | Swift mailer.....                  | 27 |
| 1.9.3 | Lexik form filter.....             | 27 |
| 1.9.4 | Knp paginator .....                | 28 |
| 1.9.5 | Knp menu.....                      | 28 |
| 1.9.6 | Nelmio cors .....                  | 29 |
| 1.9.7 | FOSJs routing .....                | 29 |
| 1.9.8 | Stof doctrine extension .....      | 29 |
| 2     | Návrh rezervačního systému.....    | 30 |
| 2.1   | Zvolený způsob řešení.....         | 30 |
| 2.2   | Datový model .....                 | 30 |
| 2.3   | Aktéři.....                        | 32 |
| 2.3.1 | Koncový zákazník.....              | 32 |
| 2.3.2 | Zaměstnanec .....                  | 32 |
| 2.3.3 | Správce firmy.....                 | 32 |
| 2.3.4 | Správce rezervačního systému.....  | 32 |
| 2.3.5 | City.cz .....                      | 32 |
| 2.4   | Základní grafické uspořádání ..... | 33 |
| 2.5   | Programování datového modelu ..... | 33 |
| 2.6   | Kontroléry backend .....           | 34 |
| 2.6.1 | FirmaController .....              | 35 |
| 2.6.2 | ProvozovnaController .....         | 35 |
| 2.6.3 | RezervaceController .....          | 35 |
| 2.6.4 | TerminController .....             | 35 |
| 2.6.5 | SluzbaController .....             | 36 |
| 2.6.6 | ZakaznikController .....           | 36 |
| 2.6.7 | ZamestnanecController .....        | 36 |



|        |   |    |
|--------|---|----|
| 2.6.8  | EmailController .....   | 36 |
| 2.7    | Kontroléry frontend.....  | 37 |
| 2.7.1  | LoginController .....   | 37 |
| 2.7.2  | ProvozovnaController .....                                      | 37 |
| 2.7.3  | RezervaceController .....                                       | 37 |
| 2.7.4  | TerminController .....  | 37 |
| 2.7.5  | ZamestnanecController .....                                     | 37 |
| 2.8    | Správa termínů .....  | 38 |
| 2.8.1  | Vlastní řešení .....  | 38 |
| 2.8.2  | Full calendar .....   | 38 |
| 2.8.3  | Popis principu .....  | 39 |
| 2.8.4  | Mazání a editace .....  | 40 |
| 2.8.5  | Výběr termínu .....   | 42 |
| 2.8.6  | Finální implementace.....                                       | 42 |
| 2.9    | Správa rezervací backend.....                                   | 43 |
| 2.9.1  | Vytvoření rezervace podle termínu.....                          | 43 |
| 2.9.2  | Vytvoření rezervace podle dodaných datumů .....                 | 44 |
| 2.9.3  | Mechanismus schvalování rezervací .....                         | 44 |
| 2.10   | Tvorba rezervací z API rozhraní .....                           | 45 |
| 2.10.1 | Vysvětlení problému.....  | 45 |
| 2.10.2 | Politika same-origin.....                                       | 46 |
| 2.10.3 | Cross origin resource sharing .....                             | 46 |
| 2.10.4 | Nelmio cors bundle .....  | 46 |
| 2.10.5 | Cross domain form submit.....                                   | 46 |
| 2.10.6 | Cross-site request forgery .....                                | 47 |
| 2.10.7 | Výsledná implementace .....                                     | 47 |
| 2.11   | Přidání nového zaměstnance .....                                | 48 |
| 2.11.1 | První zaměstnanec po založení firmy v rezervačním systému ..... | 48 |
| 2.11.2 | Další zaměstnanci .....   | 49 |
| 2.11.3 | Správa zaměstnanců.....   | 49 |
| 2.12   | Mazání v systému .....  | 49 |
| 2.12.1 | Softdelete .....  | 49 |
| 2.12.2 | Stof doctrine extensions bundle.....                            | 49 |
| 2.13   | Správa emailů .....   | 50 |
| 2.13.1 | Ukládání emailů do databáze.....                                | 50 |
| 2.13.2 | Nahrazení klíčových slov.....                                   | 50 |
| 2.13.3 | CK editor.....  | 50 |

|                                 |                                       |    |
|---------------------------------|---------------------------------------|----|
| 2.13.4                          | Swift mailer bundle .....             | 51 |
| 2.14                            | Návrh menu .....                      | 51 |
| 2.14.1                          | Menu bundle .....                     | 52 |
| 2.15                            | Návrh vzhledu .....                   | 52 |
| 2.15.1                          | AdminLTE .....                        | 52 |
| 2.15.2                          | Finální výstup .....                  | 53 |
| 2.16                            | Notifikace příchozích rezervací ..... | 53 |
| 2.16.1                          | Popis principu .....                  | 53 |
| 2.17                            | Testování .....                       | 54 |
| Závěr .....                     |                                       | 55 |
| Seznam použité literatury ..... |                                       | 57 |
| Přílohy .....                   |                                       | 59 |

## Seznam obrázků

|   |    |
|---|----|
| Obrázek 1 – Silvertown production s.r.o. ( <a href="http://www.silvertown.cz">www.silvertown.cz</a> , 2016) .....       | 14 |
| Obrázek 2 – Architektura MVC ( <a href="http://www.symfony.com/legacy/doc">www.symfony.com/legacy/doc</a> , 2011) ..... | 17 |
| Obrázek 3 – Ukázka kódu registrování služeb .....   | 21 |
| Obrázek 4 – Ukázka kódu Voteru .....  | 22 |
| Obrázek 5 – Šablonovací framework AdminLTE ( <a href="http://www.adminlte.io">www.adminlte.io</a> , 2014) .....         | 25 |
| Obrázek 6 – Full calendar náhled ( <a href="http://www.fullcalendar.io">www.fullcalendar.io</a> , 2018) .....           | 26 |
| Obrázek 7 – Vzhled CK editoru .....   | 27 |
| Obrázek 8 – Ukázka kódu pro generování prvků v Knp menu bundlu .....  | 28 |
| Obrázek 9 – Datový model .....  | 31 |
| Obrázek 10 – Hrubý grafický návrh .....   | 33 |
| Obrázek 11 – Ukázka entity provozovny .....   | 34 |
| Obrázek 12 – Proces denního opakování termínů .....   | 39 |
| Obrázek 13 – Proces měsíčního opakování termínů .....   | 40 |
| Obrázek 14 – Generování události pro Full calendar .....  | 40 |
| Obrázek 15 – Proces vytváření „mezery“ mezi termíny .....   | 41 |
| Obrázek 16 – Ukázka kódu pro vytváření „mezery“ mezi termíny .....  | 41 |
| Obrázek 17 – Výpis termínů .....  | 43 |
| Obrázek 18 – Tvorba rezervačního formuláře podle dodaného datumového rozsahu ...  | 44 |
| Obrázek 19 – Workflow mechanismu pro schvalování rezervací .....  | 45 |
| Obrázek 20 – API rozhraní rezervačního systému .....  | 48 |
| Obrázek 21 – Vykreslení horní lišty .....   | 51 |
| Obrázek 22 – Ukázka kódu pro nalezení aktuálně zvolené firmy .....  | 52 |
| Obrázek 23 – Finální vzhled .....   | 53 |

## **Seznam použitých zkratek**

|      |                                    |
|------|------------------------------------|
| PHP  | Hypertext preprocesor              |
| CMS  | Content management system          |
| SŘBD | Systém řízení báze dat             |
| MVC  | Model view controller              |
| CSS  | Cascading style sheets             |
| SQL  | Structured query language          |
| HTTP | Hypertext transfer protocol        |
| FTP  | File transfer protocol             |
| IDE  | Integrated development environment |
| API  | Application programming interface  |
| URL  | Uniform resource locator           |
| DQL  | Doctrine query language            |
| ORM  | Object relational mapping          |
| JSON | Javascript object notation         |
| XML  | Extensible markup language         |
| CORS | Cross origin resource sharing      |

## Úvod

Firmy a živnostníci v dnešní době často hledají nástroje, které jim zjednoduší a zpřehlední práci se zákazníky. Jedním z takových nástrojů je právě rezervační systém. Jedná se o formu informačního systému, který má za úkol jednoduchým a intuitivním způsobem evidovat požadavky koncových zákazníků. Celý proces evidence je možný bez nutnosti osobního setkání nebo telefonické domluvy, pouze za použití internetu. Vlastníkovi účtu v rezervačním systému jsou poté tyto požadavky v přehledné formě poskytnuty.

Účelem rezervačního systému není pouze požadavky poskytovat, jeho úkolem je také ulehčovat jejich správu a definovat hranice, v jejichž rozsahu může být rezervace koncovým zákazníkem učiněna.

Práce právě na takovémto rezervačním systému mi byla zaměstnavatelem nabídnuta. Projekt je modulem redakčního portálu city.cz a má být navržen tak, aby umožňoval případné propojení s moduly dalšími. Je také vhodné zmínit, že jmenovaný portál má již v sobě funkcionalitu rezervačního systému zabudovanou. Mým úkolem je tento systém přepracovat do samostatného modulu, který není přímo závislý na redakčním portálu a provést změny podle požadavků zaměstnavatele.

Zadavatelem mi bylo také sděleno, že se nemám jakkoliv inspirovat aktuální implementací rezervačního systému a mám vytvořit systém vlastní. V tomto dokumentu se tedy nebudu původním programem zabývat a v rámci jeho obsahu se nebude vyskytovat ani žádná forma komparace.

Pro vytvoření rezervačního systému použiji PHP framework Symfony, který je webovým aplikačním frameworkem, vycházejícím z návrhového vzoru MVC. V rámci tohoto nástroje je možné využívat tzv. Bundle systém. Jedná se o komponentu, která umožňuje do projektu vkládat funkcionalitu od vývojářů třetích stran. Mimo prostředky, získané prostřednictvím komponenty Bundle systém, použiji také tyto nástroje: Javascript, jQuery, Ajax, CSS, MySQL, Bootstrap a další.

## Motivace

Protože je má bakalářská práce zároveň požadavkem od mého zaměstnavatele, jedním z důvodů pro její vytvoření je právě finanční ohodnocení, které získám jejím dokončením.

Mým druhým důvodem pro práci na tomto projektu je osvojení si nejnovějších trendů z oblasti webových technologií a zdokonalení mých programátorských dovedností ve frameworku Symfony.

## Cíl práce

Výstupem bakalářské práce má být rezervační systém, který umožňuje registrované firmě zadávat, evidovat a zpracovávat rezervace provedené koncovými zákazníky.

Rezervační systém má umožňovat zadávání rezervací jak koncovým zákazníkem, tak vlastníkem účtu v rezervačním systému. API rozhraní rezervačního systému, které je určené pro zadávání rezervací bez nutnosti registrace, musí být možné vložit na stránky registrované firmy. Jak jeho vykreslování, tak jeho zpracování musí být možné provádět napříč doménami. Administrátorská část musí obsahovat tyto funkce:

- Vytváření, schvalování a stornování rezervací.
- Přidávání, editování a mazání zaměstnanců, provozoven, služeb a emailů.
- Stanovení hranic API rozhraní pomocí termínů.
- Evidenci zákazníků.

## O firmě

Společnost, ve které pracuji a pro kterou zpracovávám tuto bakalářskou práci, se nazývá Silvertown production s.r.o. Firma se zabývá zpravodajstvím, grafikou, reklamou a mimo jiné také tvorbou webových stránek. Primární službou této společnosti je aktuálně redakční portál city.cz. Jedná se o regionální zpravodajský portál zaměřující se na nejnovější události pod spravovanými kraji.



KDYŽ MÁTE CO ŘÍCT, ALE NEVÍTE JAK

Obrázek 1 – Silvertown production s.r.o. ([www.silvertown.cz](http://www.silvertown.cz), 2016)

## Spolupráce

Ve firmě jsem aktuálně zaměstnán formou dohody o provedení práce.

## **1 Představení použitých nástrojů**

V teoretické části návrhu rezervačního systému se budu zabývat převážně nástroji, které jsem při tvorbě této bakalářské práce využil. Popíši, co je to API rozhraní rezervačního systému, zmíním se zde o mém pracovním prostředí, frameworku Symfony, ale také o softwarových řešení nabízených vývojáři a společnostmi třetích stran, které jsem do projektu implementoval.

### **1.1 Co je to API rozhraní rezervačního systému?**

Protože má rezervační systém umožňovat vytváření rezervací i na webových stránkách firmy, která je evidována v systému, je nutné pro tento účel navrhnout specializovaný mechanismus. Ten jsem pojmenoval API rozhraní rezervačního systému.

Jeho úkolem je poskytnutí jednoduchého způsobu vykreslení rezervačního formuláře. Povinností tohoto rozhraní je možnost fungovat na odlišné doméně, než je samotný rezervační systém, a to bez nutnosti složitého nastavení na straně registrované firmy.

### **1.2 Použité softwarové nástroje**

Při tvorbě softwarového produktu je dobré si obstarat nástroje, které mohou výrazně zefektivnit jeho vývoj. V této kapitole se budu takovými nástroji zabývat. Popíši prostředky, které jsem osobně při vývoji rezervačního systému použil a bez kterých by vývoj často ani nebyl možný.

#### **1.2.1 Wamp server + virtual host**

Pro vývoj a zprovoznění webové aplikace se není možné obejít bez webového serveru. Jedním takovým je právě Apache. Jedná se o open-source, multiplatformní HTTP server. Protože samotný Apache nedokáže obsluhovat celou webovou aplikaci, je zapotřebí využít nějaké mohutnější řešení.

Tímto řešením je balíček nástrojů nazývaný Wamp server. Jedná se o vývojové prostředí určené pro operační systémy Windows. Toto prostředí nám umožňuje vytvářet webové aplikace využívající PHP, MySQL a má v sobě také implementovanou výše zmíněnou funkcionalitu webového serveru. Je vhodné zmínit, že prostředí vytvořené pomocí Wamp serveru je dostupné pouze na lokální stanici, kde je projekt uložen.

Pro pohodlnější práci na rezervačním systému se doporučuje nastavit tzv. virtual host. Jedná se o konfigurační nastavení Wamp serveru, které nám umožní přistupovat z prohlížeče do adresáře projektu pomocí námi definovaného doménového jména. Toto doménové jméno bude dostupné pouze na našem počítači a nebude tedy možné k němu přistoupit z internetu.

### **1.2.2 Free commander**

I přesto, že vývoj bude prováděn na lokální stanici, je nutné vzít v potaz, že některé koncepty projektu není možné odzkoušet bez vývojového prostředí na skutečném serveru. Abychom se na tento server připojili, potřebujeme nějakého vhodného FTP klienta.

Pro tento úkol jsem se rozhodl využít freeware software Free commander. Jedná se o správce souborů s integrovanou výše zmíněnou funkcionalitou, a i přesto že se na trhu nachází lepší řešení, pro mé využití je tento program dostačující.

### **1.2.3 PHP Storm**

Aby psaní programu bylo co možná nejpohodlnější, je zapotřebí zvolit vhodné IDE s funkcionalitami, které může programátor webových aplikací potřebovat.

Jedním, vývojářskou komunitou nejlépe hodnoceným prostředím, je software nazývaný PHP Storm. Jeho hlavní nevýhodou je fakt, že je placený. Protože je ale nabízen pro studenty vysokých škol zdarma, rozhodl jsem se ho pro návrh této bakalářské práce využít.

### **1.2.4 Bit bucket**

Protože modul rezervačního systému není malým programem, je zapotřebí evidovat změny, které jsem v něm provedl. Za tímto účelem jsem se rozhodl zaregistrovat do služby Bit bucket, která bude mít zálohování a verzování naprogramované aplikace na starost.

### **1.2.5 Opera**

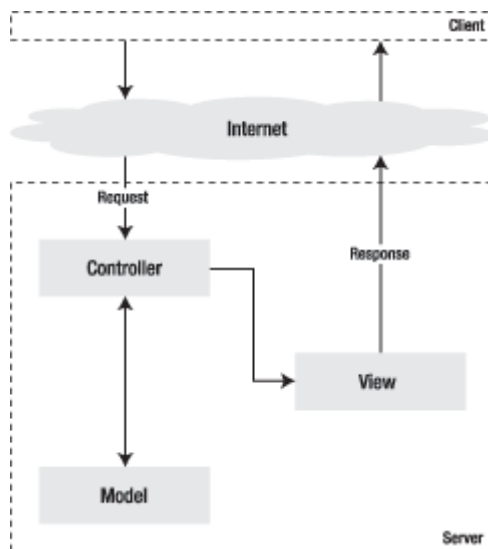
Pro vývoji rezervačního systému jsem využil prohlížeč Opera.



### 1.3 Architektura MVC

MVC je softwarovou architekturou, jejíž primárním cílem je rozdělení aplikace na tři logické části.

První částí je Model, který má na starost reprezentaci dat a business logiku aplikace. Druhou částí je View, jehož úkolem je zobrazení výstupu uživateli. A třetí částí je Controller, jenž reaguje na události a komunikuje mezi modelem a pohledem.



Obrázek 2 – Architektura MVC (www.symfony.com/legacy/doc, 2011)

Účelem tohoto rozdělení je umožnit provádění změn ve zmíněných částech, a to s minimálním dopadem na části ostatní. (Bernard, 2009)

### 1.4 Symfony framework

Jak již bylo v předchozích kapitolách zmíněno, projekt bude naprogramovaný v PHP frameworku Symfony. V této kapitole bych chtěl tento framework představit, zmínit bych se o jeho souborovém systému a na závěr bych popsal hlavní typy tříd, které jsou nedílnou součástí frameworku.

#### 1.4.1 Představení

Symfony framework je open-source nástrojem vydávaným pod licencí MIT. Zakladatel projektu Symfony, Fabien Potencier popisuje framework takto: „*Symfony2 is a reusable set of standalone, decoupled, and cohesive PHP components that solve common web development problems. Then, based on these components, Symfony2 is also a full-stack web framework*“. (Potencier, 2011)

Mezi hlavní přednosti tohoto frameworku patří jeho stabilita, kvalitní dokumentace a aktivní komunita vývojářů.

Přesto, že je všeobecně prezentováno, že Symfony framework je založený na MVC architektuře. Je nutné si uvědomit, že toto tvrzení nemusí být vždy pravdivé. Symfony skutečně poskytuje nástroje pro kontrolérovou a náhledovou část MVC architektury. Neposkytuje ale nástroje pro část modelovou. Je na vývojáři, jestli použije výchozí ORM knihovnu Doctrine nebo nějaké vlastní řešení.

Z toho důvodu je výše zmiňovaným zakladatelem architektura frameworku popsána takto: „*Symfony2 is an HTTP framework; it is a Request/Response framework. That's the big deal. The fundamental principles of Symfony2 are centered around the HTTP specification*“. (Potencier, 2011)

#### **1.4.2 PHP**

Protože je Symfony framework poháněn právě programovacím jazykem PHP, je na místě se o něm v této publikaci zmínit.

PHP je open-source skriptovacím jazykem, který se vykonává na straně serveru. Je navržen pro vývoj webových stránek, ale je také možné ho použít jako obecný programovací jazyk. PHP kód může být vestavěný přímo v HTML dokumentu, nebo může být použit v kombinaci s různými šablonovacími systémy, CMS nebo, jako v našem případě, s webovým aplikačním frameworkem. (Naramore, 2006)

#### **1.4.3 Doctrine**

Doctrine v MVC architektuře Symfony frameworku reprezentuje modelovou část. Jak již bylo řečeno, pro tuto roli není nutné použít právě Doctrine. V našem případě nám ale plně dostačuje.

Tento balík PHP knihoven je ORM frameworkem. Jeho účelem je převádění tabulkových dat do objektů a tím usnadnit vývojáři práci s daty. Jednou z klíčových vlastností tohoto frameworku je možnost psaní databázových dotazů v jazyce DQL. Jedná se o objektově orientovanou alternativu jazyka SQL.

Fragment DQL kódu po provedení předá databázovému stroji upravený SQL dotaz. Úpravy spočívají ve vyřešení nezbytných závislostí tabulek v dotazu bez nutnosti zásahu programátora. (Doctrine: Getting Started with Doctrine, 2010)

#### 1.4.4 Twig

Náhledovou část MVC architektury frameworku Symfony zastupuje šablonovací systém Twig, který byl pro tento framework přímo navržen.

Prioritní prací šablonovacích systémů je vkládání záznamů, získaných třeba z databáze, do statických elementů webových stránek. Tímto odpadá nutnost vytváření nové stránky pro každý záznam a zároveň je snížen požadavek na diskovou kapacitu.

Protože Twig je formou skriptovacího jazyka, je nutné, aby měl definovanou svojí syntaxi a formát souboru. Jeho vykonáním získáme čistý HTML kód. (Twig, c2010-2018)

#### 1.4.5 Struktura adresáře

Projekty navržené ve frameworku Symfony mají vlastní adresářovou strukturu, kterou je doporučeno následovat. Není to ale pravidlem a z důvodu nevhodné konfigurace hostingového serveru jsem byl sám nucen strukturu adresáře rezervačního systému upravit.

**Tabulka 1 – Doporučená struktura adresáře Symfony projektu**

| Složka  | Obsahuje                                |
|---------|---|
| app/    | Konfigurace aplikace, šablony, překlady |
| bin/    | Spouštěcí soubory (např. bin/console)   |
| src/    | PHP kód projektu                        |
| tests/  | Automatické testy                       |
| var/    | Generované soubory (cache, logy atd.)   |
| vendor/ | Závislosti třetích stran                |
| web/    | Kořenový adresář webu                   |

Zdroj: ([www.symfony.com/doc/3.4/quick\\_tour/the\\_architecture.html](http://www.symfony.com/doc/3.4/quick_tour/the_architecture.html), 2010)

Adresáře, které v rezervačním systému zůstaly na svém místě, jsou složka app, bin, src, var a vendor. Protože v projektu nevyužívám automatických testů, adresář tests se v navrhovaném projektu nenachází.

Z důvodu již zmíněné problémové konfigurace jsem byl nucen odebrat složku web. Problém je v tom, že na vývojovém a produkčním serveru není z internetu, mimo routy kontrolérů, přístup do adresářové struktury projektu povolen. Tedy ani do adresáře web, a proto všechny soubory typu public museli být přesunuty mimo adresářovou strukturu projektu.

#### **1.4.6 Bundle systém**

U řady softwarových produktů je možné se setkat s termínem plugin. Jedná s formu zásuvného modulu, který není sám o sobě funkční, ale je schopný poskytnout funkcionalitu v kombinaci se základním programem.

Bundle je pluginům velmi podobný. Zásadním rozdílem je fakt, že v Symfony projektu je bundlem vše. Od jádra frameworku až po mnou vytvářený projekt. Tato vlastnost poskytuje vývojářům pružnost při použití vestavěných funkčních balíků, které se nacházejí v bundlech vývojářů třetích stran. (Symfony: The Bundle System, c2010-2018)

Tento systém při vývoji budu často používat a jednotlivým balíkům se budu věnovat v samostatných kapitolách.

#### **1.4.7 Entity**

Programátoři, kteří pracují s databází bez žádného ORM frameworku mohou často narazit na limitaci práce s daty, které jsou řádkově strukturované. Je vhodné takové záznamy prvně převést do objektové struktury, se kterou se poté snáze pracuje. Touto objektovou strukturou je právě entita.

Entity jsou konceptem, jejichž správu má na starost ORM framework Doctrine, konkrétně metoda entity manager, která je jeho součástí. Právě pomocí této metody může vývojář k entitám snadno přistupovat, může je vyhledávat, mazat, ale také má možnost je pomocí tohoto nástroje snadno vytvářet. (Doctrine: Getting Started with Doctrine, 2010)

#### **1.4.8 Kontroléry**

Ve chvíli kdy klient provede požadavek na server, je to právě kontrolér, který má za úkol tento požadavek zpracovat a vrátit klientovi odpověď. Ta může být ve formátu HTML, JSON, XML případně jiném.

Ačkoliv je možné v kontrolérech provádět libovolnou výpočetní logiku, není doporučeno provádět v něm složité operace přímo. Pro tyto účely v Symfony existuje komponenta služby. (Symfony: Controller, c2010-2018)

### 1.4.9 Služby

Často se stane, že programátor potřebuje definovat funkcionalitu, která je dostupná pro ostatní funkce napříč projektem, ale je nežádoucí, aby k ní měl přímý přístup klient. Z tohoto důvodu je v Symfony definována komponenta služeb.

Ve chvíli, kdy je služba vytvořena a zaregistrována v konfiguračním souboru, stane se dostupnou v komponentě nazývané přepravka služeb. Po explicitním definování služby je k ní, případně ostatním službám, možné z přepravky přistoupit v projektu téměř odkudkoliv. (Symfony: Service container, c2010-2018)

#### 1.4.10 Konfigurační soubory

Rolí konfiguračních souborů je nastavování globálních politik napříč projektem. Tato nastavení je možné aplikovat jak na projekt, tak na instalované balíky od vývojářů třetích stran.

```
AppBundle\Entity\RezervacniSystem\ZamestnanecRSRepository:
    factory: 'doctrine.orm.entity_manager:getRepository'
    arguments: [ AppBundle\Entity\RezervacniSystem\ZamestnanecRS ]
gedmo.listener.softdeleteable:
    class: Gedmo\SoftDeleteable\SoftDeleteableListener
    tags:
        - { name: doctrine.event_subscriber, connection: default }
    calls:
        - [ setAnnotationReader, [ "@annotation_reader" ] ]
automatic_creator:
    class: AppBundle\Service\AutomaticCreator
    arguments: [ "@doctrine.orm.default_entity_manager", "@security.token_storage", "@session", "@twig", "@router" ]
    public: true
automatic_emailer:
    class: AppBundle\Service\AutomaticEmailer
    arguments: [ "@twig", "@session", "@mailer" ]
    public: true
calendar_interface:
    class: AppBundle\Service\CalendarInterface
    arguments: [ "@doctrine.orm.default_entity_manager", "@router" ]
    public: true
url_generator:
    class: AppBundle\Service\UrlGenerator
    arguments: [ "@doctrine.orm.default_entity_manager" ]
    public: true
current_finder:
    class: AppBundle\Service\CurrentEntityFinder
    arguments: [ "@doctrine.orm.default_entity_manager", "@request_stack", "@security.token_storage" ]
    public: true
```

Obrázek 3 – Ukázka kódu registrace služeb

Pomocí konfigurací je možné vytvářet globální proměnné, registrovat služby, nastavovat přístupy k databázi, měnit chování již implementovaných funkcionalit, spravovat systém rolí a mnohem více.

### 1.4.11 Voter

I přesto že Symfony framework obsahuje propracovaný bezpečnostní systém, může se stát, že programátorovi nevyhovuje jeho globální charakter. V případě, že chce uživatel nastavit práva v závislosti na konkrétní entitě, stane se pro něj globální bezpečnostní systém nepoužitelný.

```
protected function voteOnAttribute($attribute, $subject, TokenInterface $token)
{
    $user = $token->getUser();

    if (!$user instanceof User) {
        return false;
    }

    $firma = $subject;

    switch ($attribute) {
        case self::ACCESS:
            return $this->canAccess($firma, $user);
        case self::ADMINACCESS:
            return $this->canAdminAccess($firma, $user);
    }

    throw new \LogicException('Chyba voteru');
}

private function canAccess($firma, $user)
{
    return $this->zamestnanecRSRepository->isZamestnanec($firma, $user);
}

private function canAdminAccess($firma, $user)
{
    return $this->zamestnanecRSRepository->isAdmin($firma, $user);
}
```

Obrázek 4 – Ukázka kódu Voteru

Pro tyto účely Symfony využívá komponentu Voter. Jeho účelem je na základě předaných parametrů rozhodnout, zda uživatel má, či nemá přístup k funkci, na kterou je Voter aplikován. Rozdíl je v tom, že oproti bezpečnostnímu systému Symfony, může být porovnávaným parametrem Voteru libovolná proměnná.

## 1.5 Databázový systém

Pro správné fungování rezervačního systému je zapotřebí využít některý z dostupných databázových systémů. Protože je ale databázový systém součástí námi nainstalované aplikace Wamp serveru, není potřeba hledat alternativní řešení.

### 1.5.1 MySQL

Rezervační systém bude pro ukládání dat využívat SŘBD MySQL. Je to softwarový nástroj, který vytváří komunikační rozhraní mezi programem a uloženými daty. Pro řízení toku dat v tomto prostředí lze využívat dotazovacího jazyka SQL. (Delisle, 2004)

### **1.5.2 PhpMyAdmin**

Pro správu obsahu databáze má Wamp server v sobě importovaný nástroj PhpMyAdmin. Jedná se o webovou aplikaci napsanou v jazyce PHP, která umožňuje programátorovi manipulovat s databázovou strukturou či obsahem ze svého internetového prohlížeče. (Delisle, 2004)

## **1.6 Správce balíčků**

Symfony nabízí možnost přidávání knihoven od společností, případně vývojářů třetích stran. Tyto balíčky jsou dostupné na nezávislém uložišti, ze kterého si programátor může potřebné nástroje kdykoliv stáhnout. Jedná se efektivní způsob rozšíření funkčnosti vyvíjených projektů. Jejich primární funkcí je instalace balíčků včetně potřebných závislostí a podpora případných aktualizací.

### **1.6.1 Bower components**

Bower je balíčkovacím systémem cíleným převážně na frontendové nástroje. Spravuje prvky obsahující HTML, CSS, Javascript, fonty a další elementy určené pro vývoj klientské části. (Bower, c2012)

### **1.6.2 Composer**

V případě, že programátor potřebuje funkcionalitu pro vývoj na backendové části, Composer mu poskytuje řešení ve formě správce PHP závislostí a požadovaných knihoven. (Composer: Getting Started, c2012)

## **1.7 Nástroje pro tvorbu vzhledu**

Cílem webu je poskytnout návštěvníkovy webových stránek užitek. Pokud ale stránka ve čtenáři nevzbudí vhodný první dojem, je možné že už vaši stránku nenavštíví.

Z toho důvodu vznikla mezioborová disciplína nazývaná webdesign. Účelem webdesignu je vytváření funkčních webových stránek a aplikací. Pro tento záměr vznikly různé nástroje, které tuto disciplínu usnadňují a umožňují vývojářům vytvořit atraktivní webovou stránku bez zbytečných složitostí.

### **1.7.1 HTML**

Téměř každá webová stránka je strukturovaná ve formátu nazývaném HTML. Jedná se o formu značkovacího jazyka, určenou pro publikaci dokumentů na internetu a je jedním z hlavních jazyků v systému World Wide Web. (Hauser, 2006)

### **1.7.2 CSS**

Kaskádové styly, z překladu plného názvu zkratky CSS, jsou nástrojem pro definování vzhledu a rozložení prvků v HTML dokumentu. Hlavní motivací pro používání kaskádových stylů je jejich znovu použitelnost kódu napříč projektem. Zdrojový soubor stačí přiložit do hlavičkové části hlavní stránky projektu. (Hauser, 2006)

### **1.7.3 Bootstrap**

Programátor často narazí na problém, že potřebuje navrhnout vzhled webu tak, aby měl responsivní vlastnosti a správně se zobrazoval na různých zařízeních. Touto funkcionalitou disponuje sada knihoven Bootstrap, která slouží pro vývoj náhledových částí webových stránek. Součástí tohoto balíku jsou HTML, CSS a Javascriptové šablony, které má programátor při vývoji kdykoliv k dispozici. (Bootstrap, c2010)

### **1.7.4 Font awesome**

Tento soubor nástrojů disponuje sérií kdykoliv použitelných ikon. K dnešnímu dni je těchto ikon v sadě více než 2700. (Font awesome, c2017)

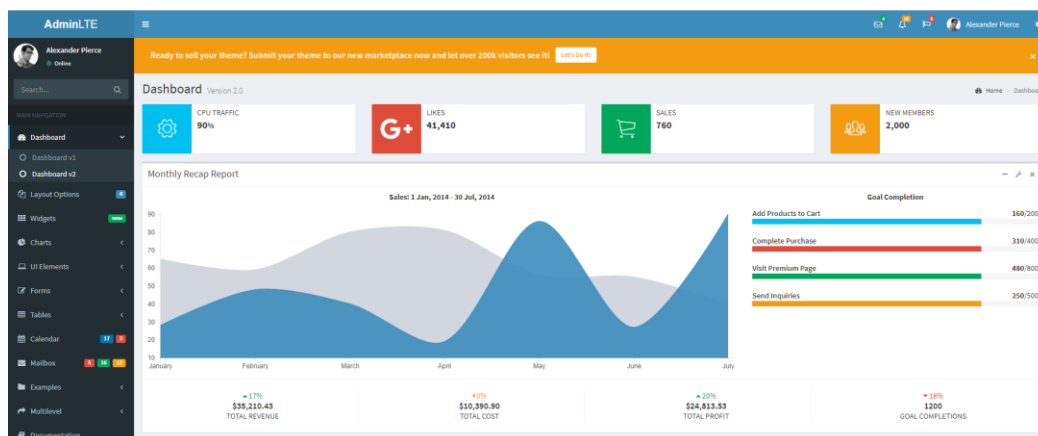
### **1.7.5 Select2**

Select2 je nad jQuery vybudovaná knihovna, která slouží jako nahrazení dnešních select boxů v HTML. Mezi její vlastnosti patří vyhledávání v záznamech, nahrávání obsahu pomocí Ajaxu a také nekonečné posouvání nalezených výsledků.

### **1.7.6 AdminLTE**

Při vytváření webových stránek se často vyplatí nalézt hlavní šablonu pro administrační a ovládací panely. Mojí volbou v tomto směru je open-source šablona AdminLTE. Jedná se o responsivní HTML šablonu založenou na frameworku Bootstrap.





Obrázek 5 – Šablonovací framework AdminLTE (www.adminlte.io, 2014)

AdminLTE je navržen tak, aby byl snadno přizpůsobitelný a bylo možné ho bez problému rozšiřovat.

## 1.8 Skriptovací nástroje

Tato kapitola se zabývá klientskými skriptovacími nástroji, které jsem při práci na projektu použil. Tyto typy nástrojů jsou tvořeny něčím, co se nazývá klientský skriptovací jazyk. Je charakteristický tím, že se provede až na straně klienta v jeho prohlížeči. Fragmenty kódu takového jazyka lze vložit přímo do HTML dokumentu, anebo je možné jejich přiložení ve formě samostatného souboru.

### 1.8.1 Javascript

Javascript je stále jedním z klíčových prvků dnešního internetu. Je využíván pro řadu interaktivních prvků, a to od tlačítek až po různé animace. Vývoj webových stránek si lze dnes bez Javascriptu jen velmi těžko představit. (Hauser, 2006)

### 1.8.2 jQuery

Protože programování v čistém Javascriptu, může být často těžkopádné, skupina jQuery team přišla se sadou Javascriptových knihoven, které výrazně ulehčují vytváření klientských skriptů.

jQuery je nástrojem, který je určený pro navigaci v HTML dokumentu, vytváření animací, a využívání Ajaxových funkcí které jsou v této knihovně dostupné. (jQuery: What is jQuery, c2018)

Nad jQuery knihovnou je také postavena knihovna jQuery-ui. Jedná se o rozšíření určené pro zatraktivnění elementů v uživatelském prostředí.

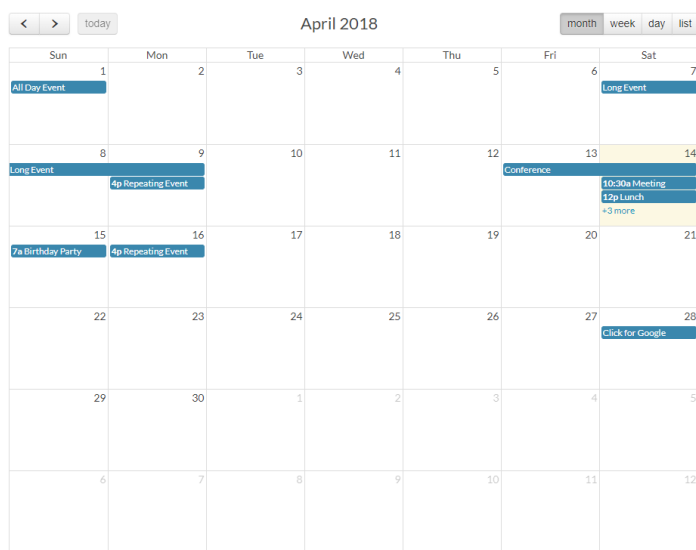
### 1.8.3 Ajax

Pro změnu obsahu na webové stránce bez nutnosti jejího obnovení se používá knihovna Ajax. Jedná se o nástroj, který umožňuje prohlížeči získat data ze serveru a následně pomocí asynchronního zpracování změnit obsah webové stránky.

Asynchronní zpracování označuje metodiku vykonání kódu mimo hlavní linii provádění programu. Díky této vlastnosti je tedy možné změnit obsah webové stránky bez nutnosti jejího znovu načtení. (Ajax: Introduction, c1999-2018)

### 1.8.4 Full calendar

Je požadavkem, aby rezervační systém uměl vhodně pracovat s daty. Rozhodl jsem se proto zvolit nějaké kalendářové řešení. Po chvilce hledání jsem nakonec našel Javascriptovou knihovnu nazývanou Full calendar.



Obrázek 6 – Full calendar náhled (www.fullcalendar.io, 2018)

Knihovna je komplexním řešením, které nabízí pokročilé funkce pro vykreslení kalendáře. Důležitá je také funkce pro správu událostí, včetně dynamického načítání obsahu pomocí Ajaxu.

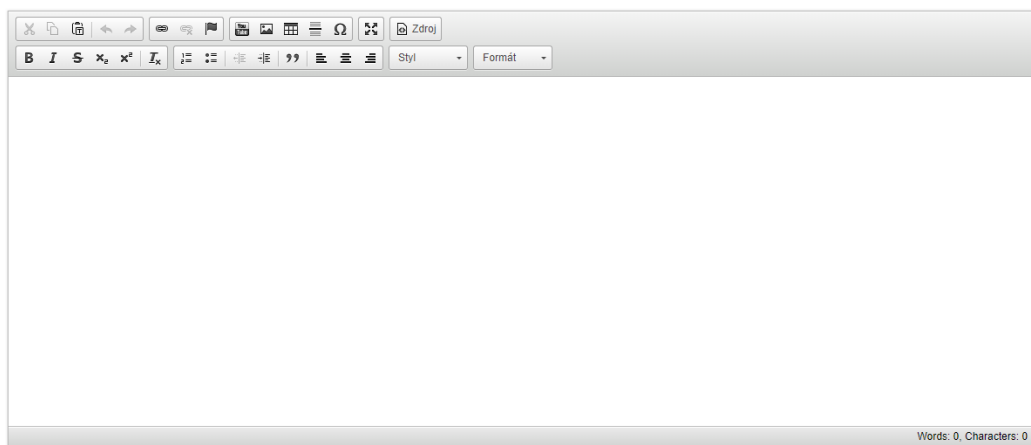
## 1.9 Použité bundly

Symfony framework umožňuje do projektu implementovat tzv. bundles. Od zahrnutých modulů zmíněných v předešlých kapitolách se liší v tom, že dodávají funkcionalitu ve formě PHP kódu. Metody obsažené v těchto knihovnách se tedy provádějí na straně serveru.

V této kapitole popíši moduly, které jsem formou bundlu implementoval do rezervačního systému.

### 1.9.1 Ivory CK editor

Pro vytváření šablon emailů chce mít uživatel k dispozici možnost vytváření těla ve formátu HTML.



Obrázek 7 – Vzhled CK editoru

Za tímto účelem použiji Ivory CK editor bundle, který integruje Javascriptový modul CK editoru přímo do formulářové komponenty Symfony.

### 1.9.2 Swift mailer

Takto vytvořené emaily je také nutné odesílat koncovým zákazníkům. Z toho důvodu je vhodné do projektu zahrnout funkci Swift mailer bundlu, která výrazně usnadňuje způsob jejich odesílání.

### 1.9.3 Lexik form filter

Protože Symfony framework neumožňuje ve výchozí sadě žádný jednoduchý způsob filtrování záznamů a vytváření výchozích formulářů za tímto účelem může být pracné,

implementoval jsem do systému knihovnu Lexik form filter bundle. Jejím účelem je předávání dodaných parametrů z formulářového okna přímo do WHERE podmínky SQL dotazu. Tento způsob umožňuje snadné a efektivní třídění ve vyhledávaných záznamech.

### 1.9.4 Knp paginator

V případě, že nám SQL dotaz vrátí více záznamů, je nutné tyto záznamy uspořádat na stránku tak, aby nezabírali příliš místa. Knp paginator bundle obsahuje metody pro stránkování obsahu a díky tomu může vývojář záznamy přehledně uspořádat do více stránek.

### 1.9.5 Knp menu

I přesto, že Symfony nedoporučuje míchat náhledovou a kontrolérovou vrstvu MVC architektury, můžeme právě menu považovat za jednu z výjimek. Nejenže je pro nás v kontrolérové vrstvě snazší menu generovat, intuitivnějším se stal také přístup k modelové vrstvě a volba aktuálně zvoleného prvku v menu v podstatě přestane být problémem.

```
$menu->addChild( child: 'header', [
    'label' => 'OVLÁDACÍ PANEL',
    'attributes' => ['class' => 'header'],
]);
$menu->addChild( child: 'dashboard', ['route' => 'firma_dashboard',
    'routeParameters' => ['firma' => $firma->getId()],
    'label' => '<i class="fa fa-dashboard"></i><span>Dashboard</span>',
    'extras' => ['safe_label' => true],
]);
$menu->addChild( child: 'zamestnanci', ['route' => 'zamestnanec_index',
    'routeParameters' => ['firma' => $firma->getId()],
    'label' => '<i class="fa fa-user"></i><span>Správa zaměstnanců</span>',
    'extras' => ['safe_label' => true],
]);
$menu->addChild( child: 'provozovny', ['route' => 'provozovna_index',
    'routeParameters' => ['firma' => $firma->getId()],
    'label' => '<i class="fa fa-home"></i><span>Správa provozoven</span>',
    'extras' => ['safe_label' => true],
]);
$menu->addChild( child: 'nastaveni', ['route' => 'firma_edit',
    'routeParameters' => ['firma' => $firma->getId(), 'type' => 'rezervace_info'],
    'label' => '<i class="fa fa-cog"></i><span>Nastavení firmy</span>',
    'extras' => ['safe_label' => true],
]);
```

Obrázek 8 – Ukázka kódu pro generování prvků v Knp menu bundlu

Sada knihoven, kterou pro kontrolérové vykreslování menu použijí, se nazývá Knp menu bundle. Jedná se o objektově orientované řešení s řadou pokročilých funkcí včetně Voterů sloužících pro nalezení aktuálně zvoleného prvku menu.

### **1.9.6 Nelmio cors**

Jednou z klíčových vlastností rezervačního systému je jeho API rozhraní, které musí být schopné pracovat na odlišných doménách. Kvůli politice same-origin ale není toto chování dovoleno. Klíčovou vlastností této politiky je totiž zakazování odpovědí prohlížečem, které na zobrazovaný web dorazili mimo doménové hranice.

Pro explicitní povolení těchto odpovědí je dostupné řešení ve formě Nelmio cors bundle. Tato knihovna umožní programátorovi stanovit, které URL cesty chce pro napříč doménové odesílání povolit. Takto povolená cesta se přestane řídit výše zmíněnou politikou.

### **1.9.7 FOSJs routing**

Symfony obsahuje funkce pro generování relativních, či absolutních URL cest pomocí dodaného názvu kontroléru a jeho parametrů. Problém nastává ve chvíli, kdy si potřebujeme takto vygenerovat cestu pomocí Javascriptu. Tímto úkolem je pověřen FOSJs routing bundle, který toto generování umožňuje.

### **1.9.8 Stof doctrine extension**

Stof doctrine extension bundle je rozšířením Doctrine knihoven. Mezi jeho funkce patří např. překládání, logování, řazení anebo softdelete který bude v projektu použit.

## 2 Návrh rezervačního systému

V praktické části návrhu rezervačního systému se budu zabývat jak jeho analýzou, tak následnou implementací systému. Analýza se bude skládat z datového modelu, základního grafického uspořádání, seznamu aktérů a výpisu požadavků ve formě popisu kontrolérů. Praktická část poté bude obsahovat popis postupu realizace výsledného produktu. Na závěr bych se také zmínil o mém způsobu testování.

### 2.1 Zvolený způsob řešení

Modul pro správu rezervací bude pracovat na samostatné doméně, s minimem závislosti na jiných částech redakčního portálu city.cz. Je nutné ho ale navrhnout tak, aby bylo možné v případě potřeby systém s jinými částmi propojit.

Systém dále nabízí možnost vygenerování unikátního HTML kódu. Ten slouží pro vykreslení API rozhraní na stránkách registrované firmy. Pomocí něj si může koncový zákazník zprostředkovat rezervaci. Pro správné fungování rozhraní musí správce firmy zahrnout knihovnu, která provede jeho obsluhu a na své stránky vložit již zmíněný fragment HTML kódu. Požadavkem pro správné vykreslení API rozhraní je jeho doménová nezávislost.

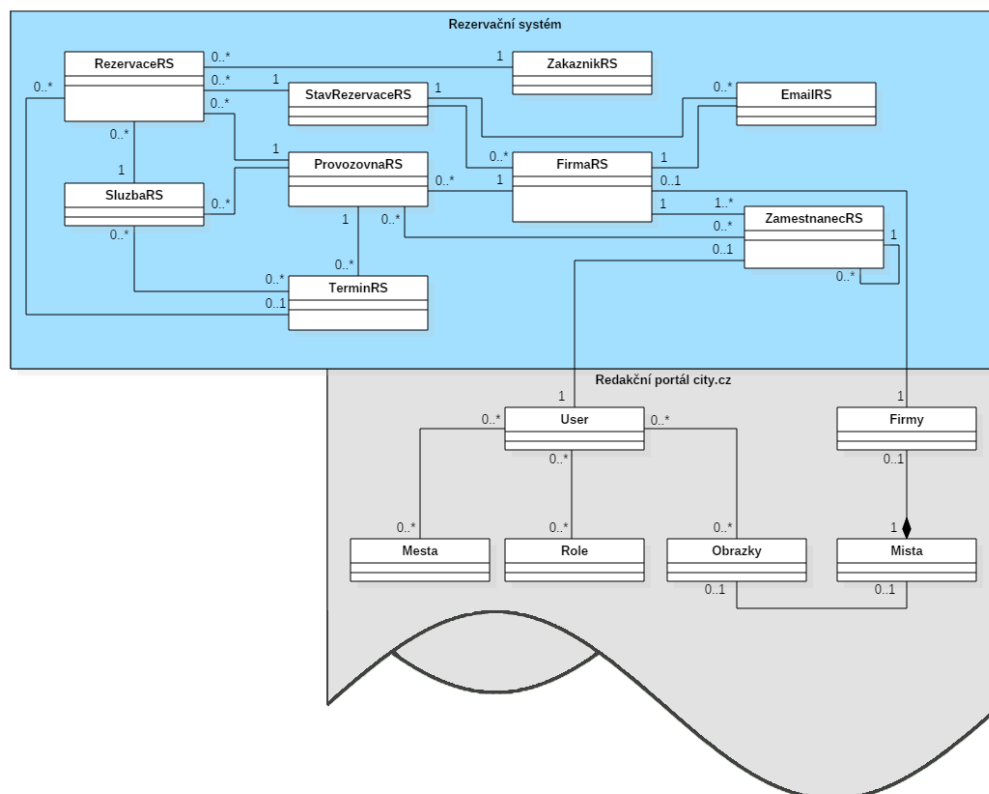
Po přihlášení do rezervačního systému bude mít uživatel s právy správce k dispozici nástroje pro správu své virtuální firmy. Jedním z těchto nástrojů je možnost manipulace s termíny. Tato funkcionalita umožňuje definování hranic, ve kterých si může koncový zákazník v rámci API rozhraní zprostředkovat rezervaci.

### 2.2 Datový model

Datový model se skládá ze dvou částí. Tou první je část rezervačního systému. Obsahuje jedenáct tabulek, z kterých dvě jsou vazební.

Pro začátek zmíním, že tabulky zaměstnanec, firma, email, provozovna, termín a služba v části rezervačního systému obsahují atributy aktivní a smazáno. Účelem atributu aktivní je povolovat, případně zakazovat zobrazení entity v API rozhraní rezervačního systému. Atribut smazáno má za úkol zakázat zobrazení entity napříč projektem.

Rezervační systém také eviduje autora a datum přidání jednotlivých záznamů. Pro tyto účely je v tabulkách zaměstnanec, firma, email, provozovna, služba, termín a rezervace atribut datum vytvoření a zadal.



Obrázek 9 – Datový model

Protože rezervace je možná založit koncovým zákazníkem bez registrace, je na místě zmínit, že atribut zadal entity rezervace, nemusí být vyplněný. V některých entitách se nachází atribut hash. Ten slouží jako unikátní identifikátor pro obsluhu API rozhraní rezervačního systému.

Druhou částí v datovém modelu je vazba do modulu redakčního portálu city.cz. Rezervační systém, ačkoliv je navržen jako samostatný modul, se neobejde bez vazby do databáze uživatelů a firem, který se v redakčním portálu nachází. Důvodem je potenciální návaznost na ostatní moduly redakčního portálu, která je nezbytností zmíněnou v předchozí kapitole. Na závěr dodám, že se ve schématu nenachází kompletní datový model redakčního portálu a tento model nebude v bakalářské práci přiložen.

Kompletní datový model rezervačního systému včetně atributů je přiložen v příloze této publikace.

## **2.3 Aktéři**

V této části popíšeme jednotlivé aktéry neboli role, které mohou nějakým způsobem zasahovat do rezervačního systému.

### **2.3.1 Koncový zákazník**

- Má možnost si založit rezervaci bez nutnosti registrace.
- Může zakládat rezervace pouze v rámci definovaných termínů.

### **2.3.2 Zaměstnanec**

- Má v systému možnost pracovat pod firmami, které mu jsou přiřazeny.
- Pod firmou může přistupovat pouze k provozovnám, které mu jsou přiřazeny.
- Může zakládat rezervace, a to jak podle termínů, tak podle datumového rozsahu.
- Má přístup ke schvalovacímu mechanismu rezervací.
- Může nastavovat chování API rozhraní prostřednictvím termínů.
- Má přístup k přehledu zákazníků, kteří pod provozovnou provedli rezervaci.

### **2.3.3 Správce firmy**

- Má práva zaměstnance.
- Může manipulovat s firemním nastavením.
- Má právo vytvářet, editovat a mazat: zaměstnance, provozovny a služby.
- Je mu umožněn přístup ke všem provozovnám pod firmou.

### **2.3.4 Správce rezervačního systému**

- Je to role v backendové části redakčního portálu city.cz.
- Má právo vytvářet a mazat firmy v rezervačním systému.
- Musí mít také možnost přidání zaměstnance s právy správce pod firmu.

### **2.3.5 City.cz**

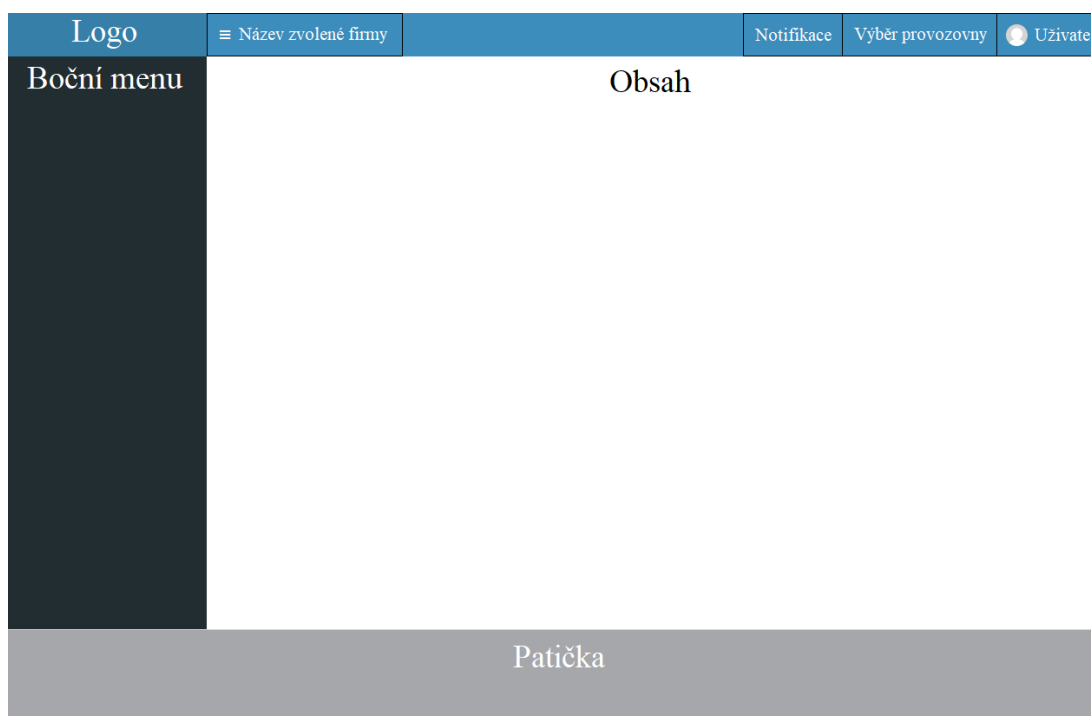
- Redakční portál.
- Dělí se na backendovou a frontendovou část.
- Rodičovský modul navrhovaného rezervačního systému.



## 2.4 Základní grafické uspořádání

Protože pro tvorbu grafického rozhraní využiji knihovnu AdminLTE, rozhodl jsem se použít výchozí rozložení, které tato knihovna poskytuje.

V horním pravém rohu se bude nacházet notifikační ikona upozorňující uživatele na nové rezervace, výpis provozoven a přehled uživatele. Obsah notifikační ikony a seznamu provozoven se mění v závislosti na zvolené firmě či provozovně. Nahoře vlevo se bude nacházet logo rezervačního systému a název zvolené firmy.



Obrázek 10 – Hrubý grafický návrh

Vlevo se bude nacházet menu rezervačního systému. Obsah tohoto menu se bude měnit podle zvolené firmy či provozovny. Na pravé straně od bočního menu se bude nacházet webový obsah a na spodní části stránky se bude nacházet patička s kontaktními a jinými informacemi.

Barevné rozložení webové stránky se nebude lišit od hrubého grafického návrhu. Jednotlivá tlačítka budou ale ve finálním produktu odlišná.

## 2.5 Programování datového modelu

Programování entit ve frameworku Symfony je velmi podobné jako v jiných programovacích jazycích. Nejdříve se definuje název neboli třída entity, poté se definují atributy, jejich datový typ, případně zvolená vazba, a nakonec se nastaví gettery a settery.

Doctrine se postará, aby se jednotlivé řádky v databázi převedly do objektů a uživatel měl poté k datům pohodlný přístup. Případně objekty přepíše zpět do řádků v případě, že uživatel provádí ukládání záznamu do databáze.

```
namespace AppBundle\Entity\RezervacniSystem;

use Doctrine\ORM\Mapping as ORM;
use Gedmo\Mapping\Annotation as Gedmo;

/**
 * AppBundle\Entity\RezervacniSystem\ProvozovnaRS
 */
/**
 * @ORM\Table()
 * @ORM\Entity(repositoryClass="AppBundle\Entity\RezervacniSystem\ProvozovnaRSRepository")
 * @Gedmo\SoftDeletable(fieldName="smazano", timeAware=false)
 */
class ProvozovnaRS
{
    /**
     * @var integer
     */
    /**
     * @ORM\Column(type="integer")
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    private $id;

    /**
     * @var string
     */
    /**
     * @ORM\Column(type="string", nullable=false, unique=true)
     */
    private $hash;

    /**
     * @var AppBundle\Entity\RezervacniSystem\FirmaRS
     */
    /**
     * @ORM\ManyToOne(targetEntity="AppBundle\Entity\RezervacniSystem\FirmaRS", inversedBy="provozovny")
     * @ORM\JoinColumn(name="firma", referencedColumnName="id")
     */
    private $firma;
```

Obrázek 11 – Ukázka entity provozovny

Na obrázku 11 si můžete všimnout, že se nad atributy, případně samotnou třídou, nacházejí anotace `@ORM` nebo `@Gedmo`. Jedná se o alias knihoven Doctrine, případně jejího rozšíření Stof, které Doctrine rozšiřuje o dodatečné funkce.

Symfony velmi doporučuje pro entity definovat inkrementální identifikační číslo (v našem konkrétním případě se jedná o atribut `id`). Framework je na to připravený a obsahuje řadu funkcí, které s tímto identifikačním číslem přímo počítají.

## 2.6 Kontroléry backend

Backendová část rezervačního systému je značně rozsáhlejší než část frontendová. Nejenže je vyžadováno, aby poskytovala funkce pro správu rezervací. Je nezbytné, aby umožnila správu provozoven, zaměstnanců, služeb, termínů a zákazníků. A to vše musí být umístěno do přehledného a funkčního uživatelského prostředí.

### **2.6.1 FirmaController**

Ve chvíli kdy se uživatel přihlásí, je to právě tento kontrolér, který mu poskytne firmy, pod kterými může v rezervačním systému pracovat. Naopak, pokud nemá žádnou firmu přiřazenou, tak mu přístup zamítne.

Mezi další funkce, které tato třída pro firmy nabízí je možnost jejich editace a zobrazení detailu a firemního dashboardu.

Firmu může editovat pouze správce firmy.

### **2.6.2 ProvozovnaController**

Protože firma nemusí sídlit pouze pod jedním místem určení, je potřeba firmám nabídnout možnost definování různých provozoven.

ProvozovnaController nabízí možnost výpisu provozoven pod firmou, jejich přidávání, editaci, mazání a také zobrazení detailu a dashboardu provozovny.

Přidávání, editace a mazání provozovny je dostupné pouze pro správce firmy.

### **2.6.3 RezervaceController**

Rezervace jsou neodmyslitelnou součástí rezervačního systému. Poskytují zaměstnancům firmy informace o požadavcích koncových zákazníků.

Backendová verze kontroléru RezervaceController umožňovat rezervace vypisovat, vytvářet je, a to ne jenom podle termínů, ale také podle datumového rozsahu, editovat je a nabízí také k dispozici schvalovací mechanismus rezervací.

### **2.6.4 TerminController**

Termíny stanovují hranice pro vytváření rezervací v API rozhraní. Lze definovat hranice pro datumový rozsah, počet osob na rezervaci, počet rezervací nebo rozsah služeb.

Kontrolér termínů má k dispozici funkce pro přidávání, editování a mazání termínů. Musí také umět termíny vykreslit a zobrazit jejich detail.

### **2.6.5 SluzbaController**

Pokud se koncový zákazník rozhodne provést rezervaci, je nutné mu poskytnout případné služby, ze kterých si může vybírat. Tento kontrolér umožňuje takovýto výběr spravovat.

Kontrolér určený pro služby obsahuje funkce výpisu, přidávání, editování a mazání služeb.

Funkcionalita tohoto kontroléru je určena pouze pro správce firmy.

### **2.6.6 ZakaznikController**

V případě, že by firma potřebovala přehled zákazníků, kteří si u ní zřídili rezervaci, nabízí systém jejich jednoduchý přehled ve formě výpisu. Je také možné zákazníky editovat, tato funkcionalita je ale dostupná pouze správcům. Zaměstnanci mohou editovat pouze zákazníky, které sami přidali. V jiném případě mohou k zákazníkovi přidat pouze poznámku.

### **2.6.7 ZamestnanecController**

Pod firmou mohou pracovat různí zaměstnanci pod různými účty. Dělí se na obvyčejné zaměstnance, kteří mají na starost schvalovací mechanismus rezervací, a správce, kteří mají dostupné pokročilé funkce rezervačního systému.

Zaměstnance je z tohoto kontroléru možné vypisovat, přidávat, editovat a mazat. Z této třídy je také možné zobrazit jejich detail.

Funkce tohoto kontroléru je dostupná pouze správci firmy.

### **2.6.8 EmailController**

Emaily jsou nedílnou součástí rezervačního systému. Nejenže musejí koncové zákazníky informovat o schválení rezervace, jejich úkolem je také zákazníka informovat o případných změnách, které byly v jejich rezervaci provedeny.

Pro tento účel je vytvořen právě EmailController. jeho úkolem je výpis, přidávání, úprava, mazání a odesílání emailů které jsou pod firmou přihlášeného uživatele přiřazeny.

Přidávání, úprava a mazání emailů je dostupné pouze pro správce firmy.

## **2.7 Kontroléry frontend**

Kontroléry na frontendové části mají na starost úvodní stránku rezervačního systému včetně funkce pro přihlášení. Dále umožňují potvrdit zaměstnance, obsloužit klientské API rozhraní a poskytují koncovým zákazníkům přístup k detailům jejich rezervací. Také připomínám, že k entitě pod API rozhraním je možné přistoupit pouze pomocí hash kódu.

### **2.7.1 LoginController**

Prací tohoto kontroléru je obsluha přihlašovacího formuláře. Funkce, které jsou v této třídě k dispozici, slouží pouze pro přihlašování a odhlašování do systému.

### **2.7.2 ProvozovnaController**

Tento kontrolér poskytuje koncovým zákazníkům výpis provozoven, který se vykreslí v API rozhraní rezervačního systému. Je vhodné zmínit, že se přehled vypíše pouze tehdy, pokud má firma pod sebou více než jednu provozovnu. V případě, že vlastní provozovnu pouze jednu, je koncovému zákazníkovi poskytnut přímo kalendář termínů této provozovny a pokud firma nemá provozovnu žádnou, API rozhraní předloží koncovému zákazníkovi výjimku.

### **2.7.3 RezervaceController**

Protože na rezervacích je rezervační systém založen, RezervaceController se stal nejrozsáhlejším kontrolérem na frontendové části rezervačního systému. Jeho úkolem není pouze rezervace vytvářet, je nutné, aby umožňoval rezervace stornovat, editovat a také zobrazovat jejich detail.

### **2.7.4 TerminController**

Jedinou starostí třídy TerminController je vykreslení kalendáře a poskytování termínů, které se nacházejí pod vykreslovanou provozovnu.

### **2.7.5 ZamestnanecController**

V tomto kontroléru se nachází funkce pro potvrzení přidávaných zaměstnanců pod firmu v rezervačním systému.

## 2.8 Správa termínů

Backendová část rezervačního systému obsahuje mechanismus termínů. Jedná se o funkcionalitu tvorby hranic v API rozhraní, která striktně definuje, kdy si zákazník může a kdy nemůže vytvořit rezervaci. Ačkoliv se tato definice zdá jednoduchá, je to právě tento správce termínů, který mi při vytváření rezervačního systému dělal největší problémy.

Hlavní komplikací je fakt, že termín může obsahovat řadu stavů, které mohou ovlivňovat jeho zobrazení. Může být bez opakování, může se opakovat denně, měsíčně, může mít stanovený konec opakování nebo může mít opakování nekonečné.

Dále je nutné vzít v potaz, že se v databázi může nacházet pouze jedna instance termínu na opakování a poznáte, že ladění takového mechanismu není triviální záležitostí.

### 2.8.1 Vlastní řešení

Protože jsem nedokázal nalézt řešení, které by splňovalo všechny požadavky zaměstnavatele, rozhodl jsem se vytvořit řešení vlastní.

Bylo zapotřebí vyřešit vytváření termínů pouze na základě parametrů z jediného záznamu. Prvním parametrem, který je nutné vzít v potaz je typ opakování. Tento parametr ovlivňuje, jestli se termín má opakovat denně, měsíčně anebo vůbec. Druhým parametrem je povolení konce opakování. V případě že není povolen, termín se opakuje donekonečna, pokud ale povolen je, musí se řídit atributem konec opakování.

Posledním parametrem jsou atributy dnů a měsíců. Pokud je typ opakování nastaven na denní nebo měsíční, tak právě tyto atributy ovlivňují, který den pro denní opakování, eventuálně který měsíc v případě měsíčního opakování, se může nebo nesmí zobrazit.

A na závěr se všechny tyto atributy musí aplikovat do rozsahu datumů kalendáře.

### 2.8.2 Full calendar

Pro funkci kalendáře jsem se rozhodl využít jQuery knihovny Full calendar. Jejím primárním úkolem je přebírání termínů v předem specifikovaném formátu a jejich následné zobrazení v kalendáři.

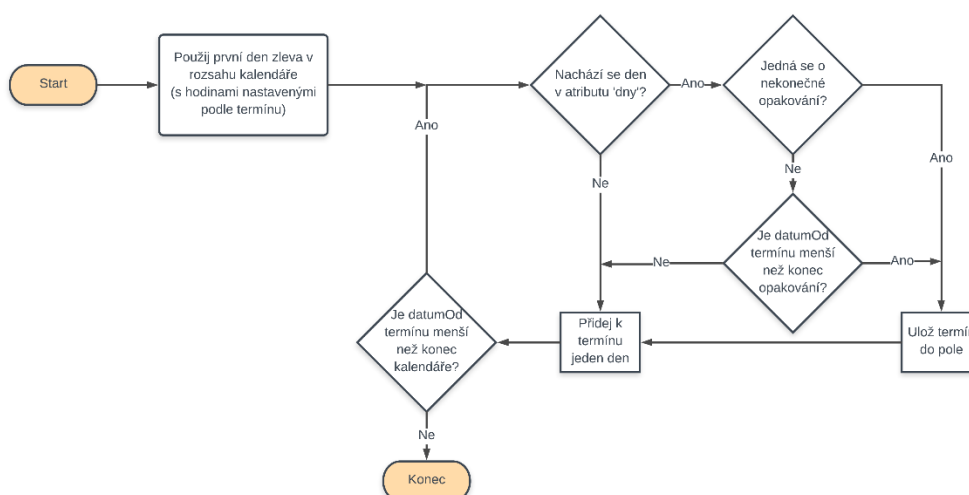
### 2.8.3 Popis principu

Prvním krokem pro zprovoznění mechanismu termínů je DQL dotaz, který získá relevantní termíny. Požadavkem je získání všech termínů, které spadají pod požadovanou provozovnu a které se:

- Neopakují a mají začátek v rozsahu kalendáře.
- Opakují a je pro ně nastaveno nekonečné opakování.
- Opakují, je pro ně nastaveno konečné opakování, a jejich konec opakování je větší než začátek kalendáře.

Data, která jsme získaly, ale není ještě možné použít. Nejdříve je musíme přizpůsobit rozsahu kalendáře, převést do správné struktury a vložit do pole, které se poté vykreslí.

V případě že je termín bez opakování, je možné ho přepsat do událostního objektu pro Full calendar a bez dalších úprav ho vložit do vykreslovaného pole.

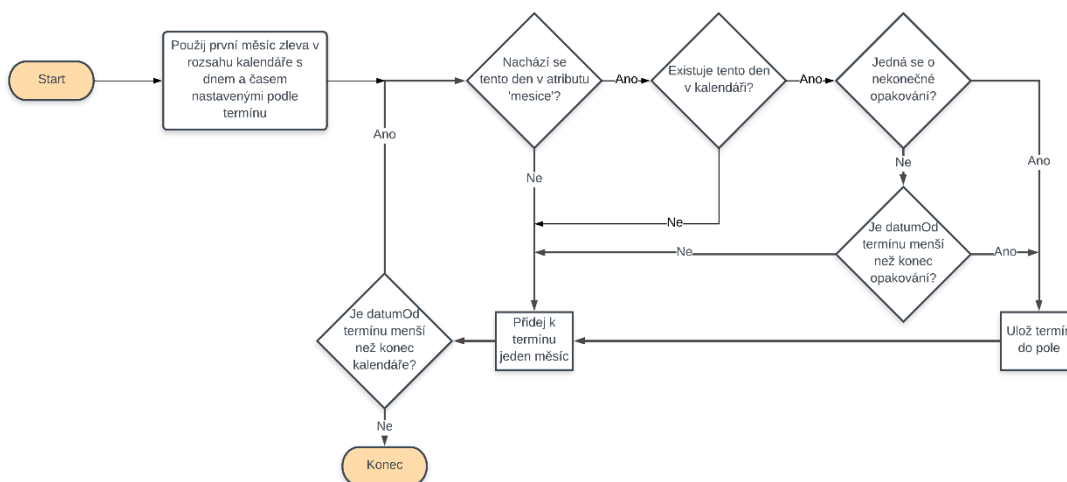


Obrázek 12 – Proces denního opakování termínů

U denního opakování se začne mechanismus trochu komplikovat. Protože se originální termín nemusí (ale může) nacházet v rozsahu kalendáře, je nutné nejdříve definovat první prvek zleva v kalendářovém rozsahu, který splňuje podmínky stanovené výše představenými počátečními parametry.

Ve chvíli, kdy je tento prvek nalezen, zbývá získat všechny termíny od nalezeného termínu zprava, které také odpovídají podmínkám. Toto generování po pravé straně pokračuje až do konce rozsahu kalendáře.

Skutečné problémy nastávají až u měsíčního opakování. Jádrem problému je fakt, že systém musí pracovat s měsíci.



Obrázek 13 – Proces měsíčního opakování termínů

Pokud tedy dokáží k termínu najít první prvek z levé strany kalendáře, který odpovídá parametrům, mám k tomuto prvku přidat třicet nebo třicet jedna dní, abych získal další termín zprava? Právě toto dilema mě donutilo pracovat s měsíci jako s integerem, který se pomocí cyklu navyšuje a ověřuje se na platnost počátečních parametrů. Tímto ale vzniká potenciální problém. Představme si, že 31. ledna navýšíme na 31. února. Protože tento den v roce neexistuje, je nutné k původním počátečním podmínkám přidat také ověření na existenci datumu v kalendáři.

```

public function jsonTerminForRezervace($termin, $podtermin) {
    return [
        'title' => $podtermin->getNazev(),
        'start' => $podtermin->getDatumOd()->format('Y-m-d H:i'),
        'end' => $podtermin->getDatumDo()->format('Y-m-d H:i'),
        'color' => $podtermin->getBarva(),
        'url' => $this->router->generate('rezervace_add_termin', [
            'termin' => $termin->getId(),
            'poradi' => $podtermin->getPoradi($termin)
        ]),
    ],
};
}
  
```

Obrázek 14 – Generování události pro Full calendar

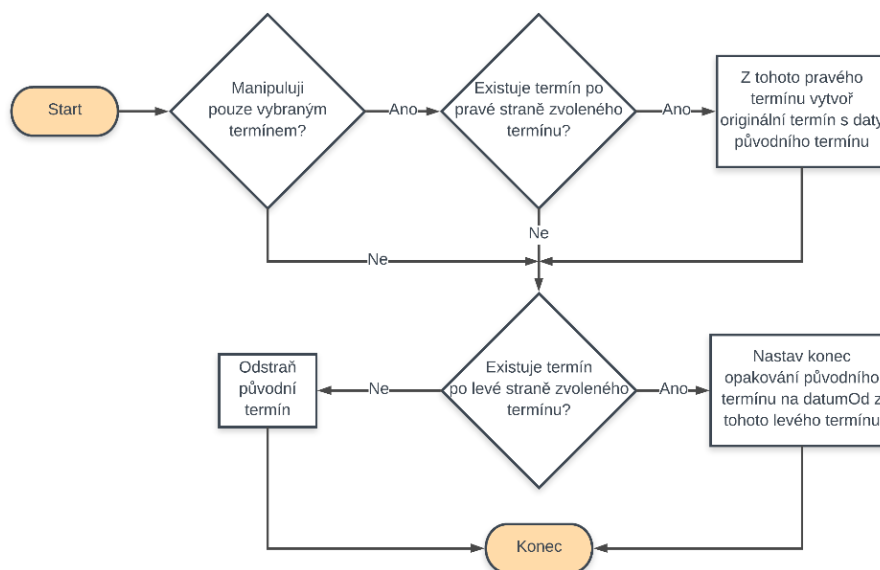
Po změně přístupu na tento model jsme poté schopni pracovat s termíny podobným způsobem jako při denním opakování.

## 2.8.4 Mazání a editace

I přesto, že je popsáno řešení funkční, je zapotřebí vzít v potaz také fakt, že je nutné tyto termíny mazat a editovat. Protože termíny, které se vykreslí v kalendáři, nemusejí vždy



existovat v databázi, narazíme často na problém, že záznam, který chceme smazat, či upravit vlastně neexistuje.



Obrázek 15 – Proces vytváření „mezery“ mezi termíny

Pro mazání termínů jsem navrhl skript, který vytváří „mezeru“ mezi termíny pro aktuálně zvolený termín. Program prvně zkontroluje, zda uživatel povolil manipulaci pouze se zvoleným termínem a zda má zvolený termín instanci termínu po své pravé straně. Pokud ano, je z původního originálního termínu vytvořen nový originální termín, ale s datumy od a do získaných z termínu, který byl nalezen po pravé straně toho zvoleného.

```

public function adjustEnvironmentAfterChange(TerminRS $original, TerminRS $termin_new, $all)
{
    if (!$all && (null !== $termin_right = $termin_new->getTerminRight($original))) {
        $termin_right = $original->createPodterminFromThis($termin_right->getDatumOd(),
                                                            $termin_right->getDatumDo(),
                                                            $original->getZadal(),
                                                            $original->getProvozovna());
        $this->em->persist($termin_right);
    }
    if (null !== $termin_left = $termin_new->getTerminLeft($original)) {
        $original->setKonecOpakovani($termin_left->getDatumOd());
        $original->setKonecOpakovaniAllow( $konecOpakovaniAllow: 1);
        $this->em->persist($original);
    }
    else { $this->em->remove($original); }

    return;
}
  
```

Obrázek 16 – Ukázka kódu pro vytváření „mezery“ mezi termíny

Poté je nutné zkontrolovat, zda má zvolený termín instanci termínu po své levé straně. Pokud ano, nastaví se konec opakování originálního termínu na hodnotu datumu od získaného z termínu, který byl nalezen po levé straně toho zvoleného. V případě že je originální termín nastaven na nekonečné opakování, je toto chování potřeba explicitně

vypnout. A pokud se po levé straně zvoleného termínu žádný termín nenachází, je zvoleným termínem termín originální a je tedy potřeba ho odstranit.

Vytvořením volného místa mezi dvěma termíny pomocí tohoto skriptu jsme schopni provést mazání. Pokud chceme provést editaci, tak musíme námi editovaný záznam prvně vytvořit.

Poté nám stačí spustit předchozí skript pro mazání a náš nový záznam vložit do nově uvolněného místa. Tímto způsobem jsme schopni provést také editaci.

### **2.8.5 Výběr termínu**

V případě zvolení termínu pro nějakou akci je nutné tento termín správně identifikovat. Z toho důvodu jsem termínům přiřadil pořadí, které je rovno počtu dnů, které se nacházejí mezi originálním termínem a zvolenou kopií. Tato hodnota se počítá přímo v getteru a není proto nutné pro tuto informaci zakládat nový sloupec v databázi.

Pro získání zvoleného termínu mi tedy stačí pouze přidat počet dnů, které jsou rovny dodanému pořadí, k originálnímu termínu.

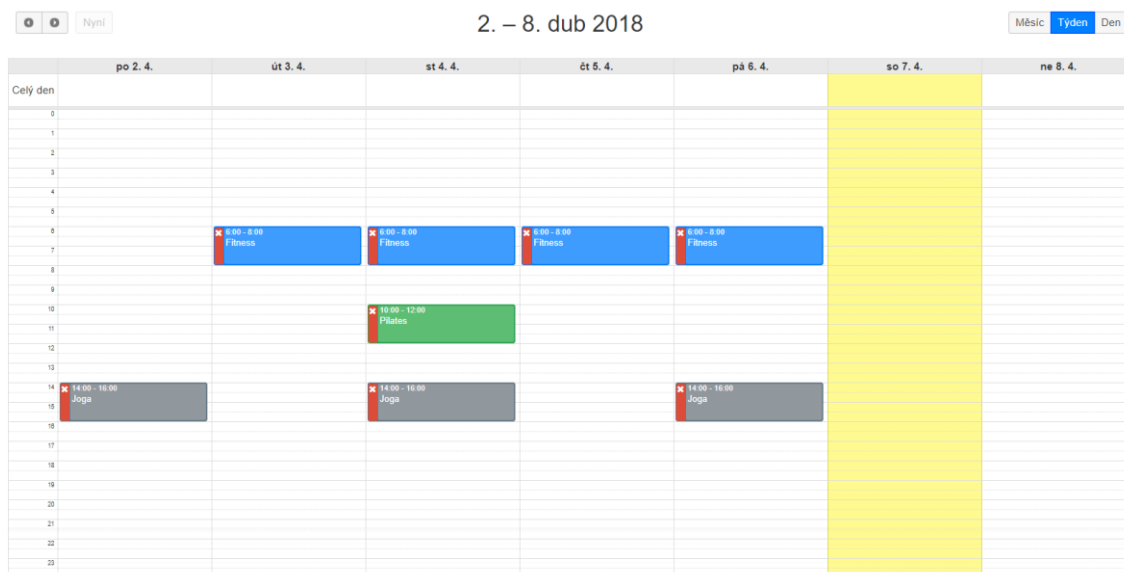
Tento přístup mi nejen umožňuje zachovat integritu termínů (není možné přepsat datумы od a do které jsou k termínům přiřazeny), je také velmi jednoduché takovéto kopie termínů validovat na správnost již popisovaných parametrů.

### **2.8.6 Finální implementace**

Nově navržený mechanismus termínů nám umožňuje vytvářet, mazat a editovat termíny. Vytvořené termíny se mohou opakovat na denní a měsíční bázi, a to bez nutnosti vytváření zbytečných záznamů do databáze. Mazání a editace lze provádět jak pro jeden prvek, všechny prvky, tak pro všechny prvky zprava od prvku zvoleného.

Je vhodné také zmínit, že programování mechanismu termínů se neobešlo bez problémů. V průběhu implementace měsíčního opakování termínu jsem narazil na problém s PHP objektem DateTime. Ten se chová nevhodně v případě zadávání neexistujících datumů. Pokud do tohoto objektu vložíme již zmíněného 31. února, objekt nám nevrátí výjimku, jak by měl, ale vrátí nám 2. nebo 3. března podle toho, jestli je nebo není přestupný rok. Protože je pro mě toto chování nepřijatelné (primárně u porovnávání dvou datumů),

rozhodl jsem se v případě, že termín je typu měsíční opakování, převádět datum na integer a až to je vhodné, zpět na DateTime.



Obrázek 17 – Výpis termínů

## 2.9 Správa rezervací backend

Administrátorská část rezervačního systému vyžaduje, aby měl uživatel k dispozici funkce pro vytváření rezervací, a to ne pouze podle dodaného termínu. Občas uživatel potřebuje vytvořit rezervaci mimo hranice stanovené těmito termíny. Je tedy proto nutné mu umožnit jejich vytváření pouze z předaného datumového rozsahu.

Uložené rezervace je také zapotřebí schválit eventuálně zamítnout. Požadavkem pro správné fungování rezervačního systému je tedy vytvoření adekvátního schvalovacího mechanismu.

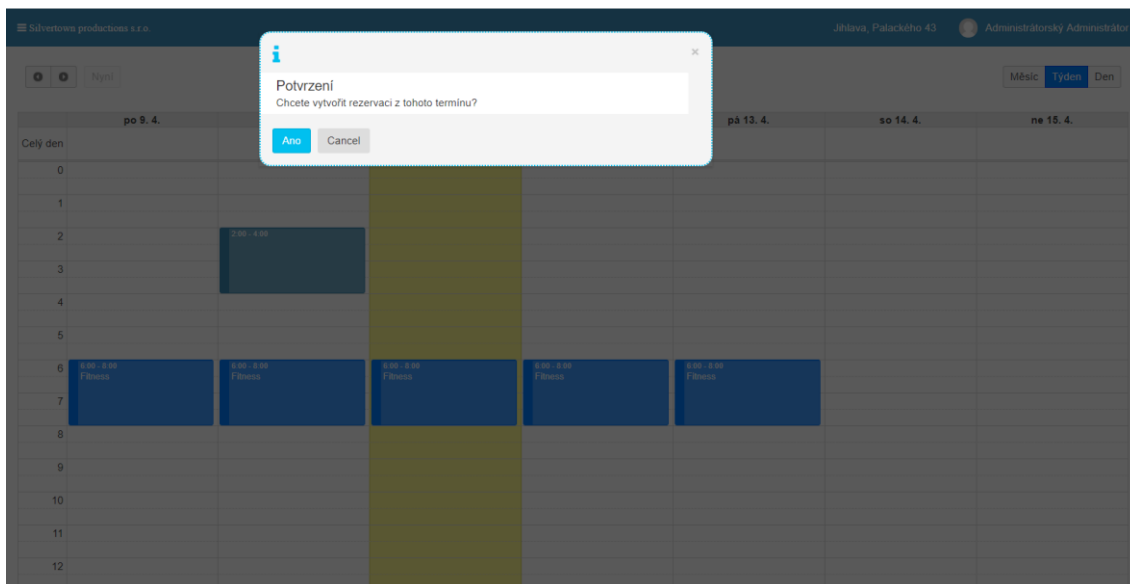
### 2.9.1 Vytvoření rezervace podle termínu

Pro vytvoření rezervace pomocí termínu je prvně nutné vykreslit kalendář s dostupnými termíny. Po kliknutí na požadovaný termín se vykreslí formulář rezervací. Tento formulář neobsahuje datum od a do. Je to způsobené tím, že změna těchto dat by porušila integraci stanovenou termínovou hranicí.

Po vyplnění a odeslání formuláře je vytvořena rezervace s daty od a do získanými z použitého termínu.

## 2.9.2 Vytvoření rezervace podle dodaných datumů

V případě, že se uživatel rozhodne založit rezervaci pouze na základě časového rozsahu, je možné využít funkci select, která je poskytnuta knihovnou Full calendar. Tato funkce dodává kalendáři možnost výběru časového rozsahu kliknutím a posunutím kurzoru v kalendáři.



Obrázek 18 – Tvorba rezervačního formuláře podle dodaného datumového rozsahu

Protože tímto způsobem vytvořený formulář není omezen hranicí stanovenou termínem, je možné uživateli poskytnout formulářové okno pro změnu datumu od a do.

Po vyplnění a odeslání formuláře se vytvoří rezervace s daty od a do získanými z těchto formulářových oken

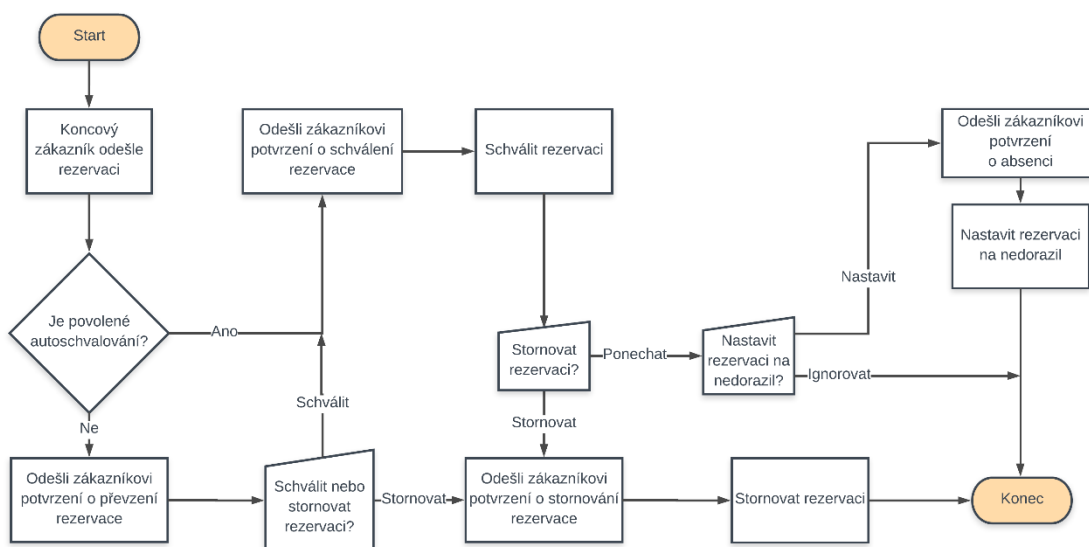
## 2.9.3 Mechanismus schvalování rezervací

Backendová část rezervačního systému uživateli umožňuje také správu již vytvořených rezervací. Tímto úkolem je pověřen implementovaný schvalovací mechanismus.

Správce firmy si prvně stanoví, zda si přeje rezervace přímo schvalovat, případně pokud chce tuto roli přenechat zaměstnancům. Ve chvíli kdy do systému dorazí neschválená rezervace, je uživatel upozorněn notifikačním panelem. V tu chvíli je rezervace dostupná v celkovém přehledu rezervací.

Tuto rezervaci může uživatel schválit anebo stornovat. Stornováním se koncovému zákazníkovi odešle email, který ho o této změně stavu informuje a rezervace samotná se uživateli schová (stornované rezervace je i nadále možné v systému dohledat).

Schválením rezervace se systému potvrdí, že se s ní počítá a koncovému zákazníkovi je emailem odeslána informace o jejím schválení.



**Obrázek 19 – Workflow mechanismu pro schvalování rezervací**

Ve schváleném stavu je možné rezervaci kdykoliv stornovat. Pokud koncový zákazník na vytvořenou rezervaci nedorazí, je k ní možné přiřadit stav o absenci.

V případě že koncový zákazník nebude pravidelně docházet na schválené rezervace, systém bude uživatele informovat o jeho nedůvěryhodnosti.

## 2.10 Tvorba rezervací z API rozhraní

Primárním účelem vytváření rezervačního systému je právě jeho API rozhraní. Firmy chtějí svým zákazníkům poskytnout možnost vytvoření rezervace pouze s využitím internetu. Proto požadují nástroj, který jim umožní vložením fragmentu kódu na své webové stránky zprovoznit funkcionalitu online rezervačního systému.

### 2.10.1 Vysvětlení problému

Zásadním problémem toho řešení je fakt, že se webové stránky registrovaných firem nachází na odlišných doménách než samotný rezervační systém. To způsobí, že se na projekt začne vztahovat omezení CSRF ochrany a politiky same-origin. Z toho důvodu je nutné navrhnout API rozhraní tak, aby mohlo správně fungovat i v prostředí s těmito omezeními.

V této kapitole všechny problémy proberu včetně způsobu jejich řešení a představím také finální způsob implementace.

### **2.10.2 Politika same-origin**

Same-origin politika je důležitým prvkem bezpečnostního modelu webových aplikací. Omezuje způsob jak dokumenty nebo skripty načtené z jedné domény mohou spolupracovat se zdroji na doméně jiné. (Ruderman, 2018)

V našem případě tato politika způsobí, že API rozhraní nedostane přístup k HTML obsahu z našeho rezervačního systému. Není ho tedy možné přes tuto politiku vykreslit.

### **2.10.3 Cross origin resource sharing**

Abychom získali povolení přístupu k tomuto obsahu, musíme využít mechanismu nazývaném Cross origin resource sharing (dále jen CORS). Ten využívá dodatečnou HTTP hlavičku, která umožňuje klientovi získat povolení pro přístup k vybraným zdrojům na serveru. A to i přesto, že se tento server nachází na odlišné doméně. (Ruderman, 2018)

### **2.10.4 Nelmio cors bundle**

Pro využití mechanismu CORS slouží právě Nelmio cors bundle. Jeho účelem je stanovení hranic, na které se politika same-origin nevztahuje.

Není vhodné z bezpečnostních důvodů aplikovat funkcionalitu tohoto bundlu na celou webovou aplikaci. CORS mechanismus je nutné aplikovat pouze na kontroléry, které mají na starost správu API rozhraní rezervačního systému.

### **2.10.5 Cross domain form submit**

I přesto, že jsme nyní schopni získat HTML obsah pro API rozhraní rezervačního systému, stále zbývá otázka, jak na server ukládat samotné rezervace učiněné koncovými zákazníky. Rozhodl jsem se využít přímý submit formuláře na server. A to zadáním URL cesty obslužného kontroléru do parametru action formulářového elementu v HTML kódu. Po provedení submitu se rezervace uloží a server učiní přesměrování na URL stránku zadanou správcem firmy do backendové části rezervačního systému.

Ačkoliv je tento přístup funkční, nastává komplikace ve formě CSRF ochrany, která se spustí po předložení formuláře serveru.

### **2.10.6 Cross-site request forgery**

Symfony framework je navržen tak, že se každý, pomocí frameworku sestavený formulář, vybaví CSRF ochranou. Ačkoliv je toto chování správné, pro účely API rozhraní je velmi nevhodné, a to právě z důvodu, že je ochrana vázaná na konkrétní doménu. Otázka tedy zní, zdali je možné tuto formu ochrany explicitně pro API rozhraní vypnout bez vytvoření nebezpečné bezpečnostní díry.

Věřím, že je namístě nejdříve vysvětlit co to vlastně CSRF ochrana je. Cross-site request forgery je formou útoku na webovou aplikaci, která zneužívá důvěru webové stránky v uživatelův prohlížeč. V době přihlášení do libovolné webové aplikace je uživateli prohlížeči přidělen unikátní kód nazývaný session. Ten opravňuje prohlížeč pro přístup do privilegované části webu. Útočníkovi tedy stačí vytvořit skript, kdy za něj takto privilegovaný prohlížeč může například submitovat formuláře.

Ochrana proti tomuto útoku je vytvořením generovaného kódu nazývaného token. Ten se připojí jako neviditelné pole k vykreslenému formuláři a server zakáže přebírání všech formulářů, který mají tento kód zadán špatně. Protože útočník nemá žádný efektivní způsob, jak tento kód získat, server zakáže všechny požadavky, které by mohl provést. (Pejša, 2008)

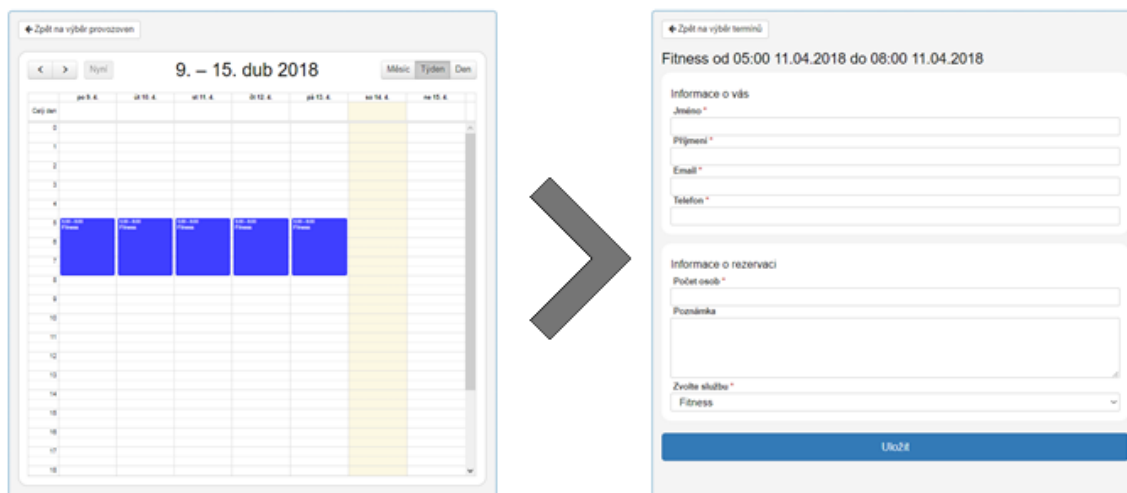
Z této definice můžeme dojít k závěru, že se CSRF útok týká pouze formulářů, ke kterým má přístup pouze privilegovaný uživatel. Protože ale API rozhraní žádná privilegia nevyžaduje, můžeme pro něj CSRF ochranu vypnout.

### **2.10.7 Výsledná implementace**

V případě že se firma rozhodne zprovoznit API rozhraní pro správu rezervací, je po správci firmy požadováno přiložení speciální knihovny do hlavičky webové aplikace své firmy. Poté si může v backendové části rezervačního systému zkopírovat unikátní kód, který po vložení do webové stránky vygeneruje popisované API rozhraní.

Rozhraní se skládá ze tří částí. V případě že má firma pod sebou více než jednu provozovnu, tak se jako první část vypíše jejich seznam. V případě, že se pod firmou

nachází pouze jedna provozovna anebo si zákazník některou z vypsaných vybere, vykreslí se kalendář s dostupnými termíny. Po výběru termínu se již vykreslí rezervační formulář, který po vyplnění a odeslání uloží rezervaci do databáze. Uložení rezervace dostane server požadavek o přesměrování uživatele na adresu webu zadanou správcem firmy do backendové části.



Obrázek 20 – API rozhraní rezervačního systému

Veškeré prvky ovládané pomocí API mají unikátní kód zvaný hash, pomocí kterého se rozhraní ovládá.

## 2.11 Přidání nového zaměstnance

Zaměstnanec je v podstatě formou vazební tabulky mezi registrovaným uživatelem a firmou v rezervačním systému. Protože je nutné zajistit, aby se do firemního účtu tito zaměstnanci dali jednoduše přidat, musíme poskytnout správcům nástroj pro jejich přidávání.

### 2.11.1 První zaměstnanec po založení firmy v rezervačním systému

První zaměstnanec, kterého je nutné pod firmu přidat, musí mít práva správce. Je přidán z backendové části redakčního portálu city.cz. Jedná se o inicializačního zaměstnance a bez něho nemůže virtuální firma pracovat. Je to právě on, kdo má na starost přidávání nových zaměstnanců.



### **2.11.2 Další zaměstnanci**

Každý další zaměstnanec může být pod firmu přidán kterýmkoliv zaměstnancem s právy správce. Do formulářového okna pro jeho přidání se vloží email uživatele registrovaného v redakčním portálu city.cz a formulář se předloží serveru.

Poté tomuto uživateli do emailové schránky přijde zpráva s odkazem, po jehož kliknutí se zaměstnanec potvrdí a je mu povolen přístup do firmy.

### **2.11.3 Správa zaměstnanců**

Správce firmy má poté k dispozici řadu nástrojů, které mu umožňují takto přidané zaměstnance dále spravovat. Má dovoleno zaměstnancům poskytovat práva správce, může jim dočasně zakázat přístup do firemního účtu anebo je může napevno odebrat.

Důležitou funkcí je také možnost povolovat, či zakazovat přístup zaměstnanců k jednotlivým provozovnám. Je ale na místě zmínit, že se toto omezení neaplikuje na zaměstnance s právy správce.

## **2.12 Mazání v systému**

Protože se v informačních systémech nedoporučuje mazání záznamů, které mají vazbu do jiných tabulek, je vhodné použít některý ze způsobů, který místo přímého mazání provádí pouze jakousi změnu stavu.

### **2.12.1 Softdelete**

Právě toto chování poskytuje softdelete. Jedná se o funkcionalitu, která vytvořením dodatečného atributu v entitě poskytne rozhraní zakazující jejich výskyt v systému. Fyzicky jsou ale v databázi dostupné a je možné k nim v případě potřeby získat přístup.

### **2.12.2 Stof doctrine extensions bundle**

Funkce softdeletu je obsažena v Doctrine rozšíření Stof doctrine extensions bundle. Jeho princip je jednoduchý. Nejdříve k entitě, na kterou chceme softdelete aplikovat, přiložíme atribut datového typu DateTime a explicitně ho nastavíme jako softdeletable.

Bundle poté přiřadí ke každému dotazu, který se na tuto entitu naváže, where podmínku, která zakáže zobrazení všech záznamů, které mají ve vytvořeném atributu datum přidané.

## **2.13 Správa emailů**

Je nutné zajistit, aby si všechny firmy mohly definovat vlastní emaily pro každý stav rezervace, nabídnout nástroj pro jejich tvorbu a také poskytnout odesílací funkci. Problémem, který tato funkcionalita představuje je fakt, že si správce firmy často přeje některé dynamické proměnné jako je jméno zákazníka, či služby které jsou dostupné pouze programátorovi. Je tedy nutné navrhnout prostředí, které všechny tyto nástroje poskytne.

### **2.13.1 Ukládání emailů do databáze**

V databázi se nachází předměty a těla emailů. Tělo jako takové je uloženo ve formátu HTML. Emaily jsou vázány na firmy a jejich použití je dostupné napříč všemi provozovny.

Protože se pod jedním rezervačním stavem může nacházet více emailů, je nutné je v přehledné formě poskytnout uživatelům. To bude provedeno ve formě drop down okna pod tlačítkem pro změnu stavu rezervace.

### **2.13.2 Nahrazení klíčových slov**

Dynamické proměnné se ukládají přímo do těla emailu a jsou v projektu explicitně definované programátorem. Pokud si správce firmy například přeje název služby, ke které se rezervace váže, zadá do těla emailu „/# sluzba #/“ a při odesílání emailu se provede přepsání tohoto prvku názvem služby, která je pod rezervací přiřazena.

Modul emailů je navržen tak že v případě požadavku je snadné definovat nové dynamické proměnné.

### **2.13.3 CK editor**

Protože je nutné ukládat emaily ve formátu HTML, je nutné zvolit některé z dostupných editorů. Pro tento úkol jsem zvolil Ivory CK editor bundle, který poskytuje komplexní řešení textové editace včetně přepínání náhledu do HTML pohledu.

#### 2.13.4 Swift mailer bundle

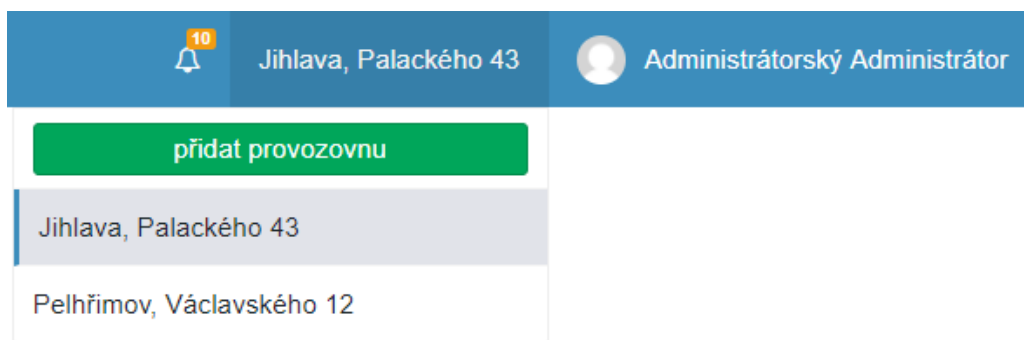
Velká řada projektů se bez odesílání emailů neobejde. Právě proto nabízí Symfony framework ve výchozí konfiguraci pro tento účel knihovnu nástrojů Swift mailer bundle.

Tato sada velmi usnadňuje odesílání emailů a postará se také o kódování a případné přílohy.

#### 2.14 Návrh menu

Při práci na menu v rezervačním systému jsem zjistil, že je pro mě statická varianta nedostačující. Hlavním důvodem byla nutnost dynamického generování obsahu v závislosti na zvolené firmě či provozovně.

Po přihlášení se uživatel, v případě že je evidován jako zaměstnanec alespoň pod dvěma firmami, ocitne ve výběru firmy. Protože ale nemá ještě žádnou firmu zvolenou, není možné vykreslit obsah jak do elementů bočního menu, výběru provozoven, tak do notifikace rezervací, které se nachází v horní liště.



Obrázek 21 – Vykreslení horní lišty

Po vybrání jedné z firem je možné do bočního menu vložit některé prvky týkající se manipulace s firmou. Je také možné vyplnit v horním menu seznam provozoven spolu s notifikačním panelem.

Teprve až po výběru provozovny je možné manipulovat přímo s rezervacemi a položkami vázanými přímo na provozovnu. Můžeme tedy překreslit boční menu i s notifikačním panelem podle zvolené provozovny.

Z důvodu vysoké variability položek v menu je pro tento úkol vhodné zvolit specializované řešení.

### 2.14.1 Menu bundle

Menu bundle je objektově orientovaným nástrojem pro tvorbu menu. Samotná výstavba jednotlivých menu položek se provádí na kontrolérové straně MVC architektury, a proto je mnohem snazší do modulu dodávat informace.

Bundle je využíván pro generování jak horního, tak bočního menu. V horním se nachází správa uživatele, výběr provozoven případně notifikace rezervací. Boční menu obsahuje tlačítka pro správu aktuálně zvolené entity firmy nebo provozovny.

```
public function getCurrentFirma() {
    $r = $this->requestStack->getCurrentRequest();

    switch(true) {
        case null !== $r->get( 'key: email' ):
            return $this->em->getRepository( className: AppBundle\Entity\RezervacniSystem\EmailRS::class )->findOneById( $r->get( 'key: email' ) )->getFirma();
        case null !== $r->get( 'key: firma' ):
            return $this->em->getRepository( className: AppBundle\Entity\RezervacniSystem\FirmaRS::class )->findOneById( $r->get( 'key: firma' ) );
        case null !== $r->get( 'key: provozovna' ):
            return $this->em->getRepository( className: AppBundle\Entity\RezervacniSystem\ProvozovnaRS::class )->findOneById( $r->get( 'key: provozovna' ) )->getFirma();
        case null !== $r->get( 'key: rezervace' ):
            return $this->em->getRepository( className: AppBundle\Entity\RezervacniSystem\RezervaceRS::class )->findOneById( $r->get( 'key: rezervace' ) )->getProvozovna()->getFirma();
        case null !== $r->get( 'key: sluzba' ):
            return $this->em->getRepository( className: AppBundle\Entity\RezervacniSystem\SluzbaRS::class )->findOneById( $r->get( 'key: sluzba' ) )->getProvozovna()->getFirma();
        case null !== $r->get( 'key: termin' ):
            return $this->em->getRepository( className: AppBundle\Entity\RezervacniSystem\TerminRS::class )->findOneById( $r->get( 'key: termin' ) )->getProvozovna()->getFirma();
        case null !== $r->get( 'key: zamestnanec' ):
            return $this->em->getRepository( className: AppBundle\Entity\RezervacniSystem\ZamestnanecRS::class )->findOneById( $r->get( 'key: zamestnanec' ) )->getFirma();
        default: return null;
    }
}
```

Obrázek 22 – Ukázka kódu pro nalezení aktuálně zvolené firmy

Pro zajištění těchto entit je vytvořena funkce, která zajistí jejich získání z libovolných parametrů, které jsou dodávány do kontrolérů.

## 2.15 Návrh vzhledu

V případě že originalita vizuální stránky projektu není prioritní, je vhodné použít některý ze šablonovacích frameworků. Pro rezervační systém jsem zvolil řešení ve formě AdminLTE.

Jde o atraktivní řešení, které dobře vypadá a jeho nasazení není velkým problémem.

### 2.15.1 AdminLTE

Po přiložení potřebných knihoven do hlavičkové části HTML dokumentu se šablona aplikuje na projekt. Změny se poté budou týkat všech elementů, které jsou definovány v rámci dokumentace. Mezi tyto elementy patří např. horní lišta, boční menu, různé vysunovací plošiny a jiné.

Mezi hlavní přednosti patří jeho responsivní vlastnosti. Díky tomu, že je postaven nad CSS frameworkem Bootstrap se vhodně zobrazuje i na mobilních zařízeních a odpadá tedy nutnost vytváření vlastního řešení.

## 2.15.2 Finální výstup

Popisovaný šablonovací framework se nabízí v různých barevných provedení. Pro rezervační systém jsem zvolil výchozí tmavě modrou barvu.

The screenshot shows the admin interface of the Rezervace city.cz system. It features a dark sidebar with navigation links like 'Dashboard', 'Kalendář', 'Rezervace', 'Termíny', 'Služby', 'Zákazníci', and 'Nastavení'. The main content area displays a table of reservations with columns for date, customer, dates, service, number of people, status, phone, email, and termination. The table lists several reservations, some with status icons (green for confirmed, red for cancelled, yellow for pending).

| Datum vytvoření | Zákazník     | Datum od        | Datum do        | Služba  | Počet osob | Stav        | Telefon   | Email                  | Termin  |
|-----------------|--------------|-----------------|-----------------|---------|------------|-------------|-----------|------------------------|---------|
| 9 07 04 04 2018 | Honza Vetchý | 0 00 27 03 2018 | 2 00 27 03 2018 | Fitness | 4          | Neschválená | 196333255 | vetchos@gmail.com      | Fitness |
| 8 59 28 03 2018 | Pepa Brambor | 0 00 10 04 2018 | 2 00 10 04 2018 | Fitness | 2          | Neschválená | 194638999 | brambor.pepa@gmail.com | Fitness |
| 8 23 28 03 2018 | Marek Horák  | 0 00 03 04 2018 | 2 00 03 04 2018 | Fitness | 2          | Schválená   | 194688597 | horak.marek@gmail.com  | Fitness |
| 8 22 28 03 2018 | Honza Vosmek | 0 00 29 03 2018 | 2 00 29 03 2018 | Fitness | 1          | Nedorazil   | 196357495 | honza.vosmek@gmail.com | Fitness |
| 8 21 28 03 2018 | Jiří Novák   | 0 00 27 03 2018 | 2 00 27 03 2018 | Fitness | 3          | Stornováno  | 167599822 | novak.jirka@gmail.com  | Fitness |

At the bottom of the interface, there is a footer with contact information for the independent media portal, technical support, and the CITYKARTA logo.

Obrázek 23 – Finální vzhled

Z AdminLTE šablony jsem v rezervačním systému použil horní lištu a boční menu. Dále jsem napříč projektem použil některé typy vysunovacích plošin a tlačítek.

## 2.16 Notifikace přichozích rezervací

Na horní liště menu se nachází ikona s rychlým přehledem nejnovějších rezervací. V případě, že se pod virtuální firmou vytvoří nová rezervace, je úkolem tohoto tlačítka na ni upozornit.

### 2.16.1 Popis principu

Po kliknutí na tlačítko notifikace se pomocí Javascriptu uloží do proměnné cookies aktuální datum. Toto datum, spolu s novým aktuálním datem označuje rozsah, ve kterém se rezervace považuje za nově přichozí.

Pomocí SQL dotazu se provede hledání všech rezervací, které do tohoto rozsahu spadají. Spočítají se a jejich zkrácený výpis se vloží do drop-down okna pod notifikačním tlačítkem.

Tento princip je velmi jednoduchý, pokud provádíme obnovení stránky. Za předpokladu, že si přejeme uživatele informovat pravidelně v nějakém časovém rozsahu, je nutné již využít asynchronní přístup.

Pro implementaci asynchronní vlastnosti využijeme knihovnu Ajax. Klient bude v časovém intervalu třiceti sekund odesílat na server požadavek s výše zmíněným časovým rozsah a ten se opět vloží do již popisovaného SQL dotazu. Získané záznamy se pomocí speciálně vytvořené Twig šablony vykreslí a odešlou ve formátu JSON zpět. Záznamy se poté pomocí Javascriptu vloží do notifikačního panelu.

## **2.17 Testování**

Testování rezervačního systému probíhalo převážně na mém lokálním zařízení. Měl jsem, ale také k dispozici vývojový server, na kterém bylo nutné testovat některé koncepty projektu.

Mezi takový koncept patřilo zejména testování API rozhraní rezervačního systému. Kdy bylo nutné nahrát aktuální verzi projektu na vývojový server a na mém lokálním stroji otestovat chování rozhraní mezi různými doménami.

Protože se jedná primárně o agilní typ projektu, ve většině ostatních případů jsem provedl pouze řadu namátkových testů po dokončení některé z větších funkcionalit. Pokud se objeví nějaká chyba v aktivním provozu aplikace, je mojí povinností ji prioritně opravit.

## **Závěr**

V případě, že koncový zákazník chce provést rezervaci bez nutnosti registrace, je pro něj připraveno API rozhraní rezervačního systému. Pro správu těchto rezervací má zaměstnanec k dispozici jejich přehled včetně schvalovacího mechanismu.

Pro správu virtuální firmy zaregistrované v systému má její správce přístup k řadě nástrojů, které jsou nedílnou součástí každého rezervačního systému. Mezi tyto nástroje patří například možnost přidávání a odebírání zaměstnanců, provozoven, emailů až po možnost nastavování hranic API rozhraní pomocí termínů.

Mezi největší problémy se kterými jsem se při vývoji setkal, patří zejména navržení systému správy termínů a API rozhraní rezervačního systému. Nevhodné fungování PHP objektu DateTime způsobilo nepřijatelné chování u termínů, které měli nastavené měsíční opakování. Z toho důvodu jsem byl nucen přepracovat tuto část tak, aby pracovala s datumy jako s integerem. Tato změna, ačkoliv vyřešila problém, výrazně znepráhlednila kód implementující celou část této funkcionality.

Největším problémem u API rozhraní je jeho nutnost fungování napříč doménami. Bylo nutné nejdříve obejít omezení same-origin politiky, která znemožňovala vykreslování rozhraní na jiné doméně než rezervační systém. To se provedlo pomocí přidání dodatečné CORS hlavičky do protokolu HTTP. Pro možnost předkládání rezervačního formuláře na server bylo také nutné vypnout jeho CSRF ochranu.

Napříč faktu, že jsem na vývoj tohoto projektu vynaložil značné úsilí, je nutné dodat, že systém má do dokonalosti daleko. Mezi funkce, které bych rád do systému implementoval, mimo rozsah této bakalářské práce, patří například navržení prostředí pro uživatele, kteří nejsou v rámci rezervačního systému registrováni a vlastní tak účet pouze v redakčním portálu city.cz.

Dále bych chtěl rozšířit funkcionality pro limitaci správy hranic formou termínů. Je vhodné doplnit tento nástroj o možnost generování termínových událostí pouze každý lichý týden, každé tři týdny, případně jiný počet týdnů. Další možností, kterou bych rád do systému správy termínů zahrnul je omezení jejich vytváření v rámci státních svátků.

Jedná se často o funkce, jejichž vývoj se může setkat s řadou komplikací a bohužel pro ně nezbyl v rámci rozsahu této bakalářské práce čas. Na druhou stranu je vhodné zmínit,

že systém disponuje nezbytným minimem, které průměrný rezervační systém musí obsahovat.

Práce na této bakalářské práci pro mě byla velmi prospěšná. Velmi jsem se zdokonalil v práci se Symfony frameworkem a také jsem si osvojil řadu postupů pro tvorbu webových aplikací.

Naučil jsem se lépe pracovat s knihovnamy od vývojářů třetích stran, pomocí kterých jsem snížil množství kódu, který bych byl jinak nucený napsat sám. Osvojil jsem si metodiky tvorby responsivních webů pomocí CSS frameworku Bootstrap a také jsem pochopil význam rozsáhlých šablonovacích frameworků typu AdminLTE.



## Seznam použité literatury

- Ajax: Introduction. *W3schools.com* [online]. W3.CSS, c1999-2018 [cit. 2018-04-15].  
Dostupné z: [https://www.w3schools.com/js/js\\_ajax\\_intro.asp](https://www.w3schools.com/js/js_ajax_intro.asp).
- BERNARD, Borek. Úvod do architektury MVC. *Zdroják.cz* [online]. 7. 5. 2009 [cit. 2018-04-15]. Dostupné z: <https://www.zdrojak.cz/clanky/uvod-do-architektury-mvc>.
- Bootstrap. *Bootstrap* [online]. Bootstrap, c2010 [cit. 2018-04-15]. Dostupné z: <https://getbootstrap.com>.
- Bower. *Bower* [online]. Bower, c2012 [cit. 2018-04-15]. Dostupné z: <https://bower.io>
- Composer: Getting Started. *Composer* [online]. Private packagist, c2012 [cit. 2018-04-15]. Dostupné z: <https://getcomposer.org/doc/00-intro.md>.
- DELISLE, Marc. *PhpMyAdmin: efektivní správa MySQL*. Brno: Zoner Press, 2004. Encyklopedie webdesignera. ISBN 80-86815-09-9.
- Doctrine: Getting Started with Doctrine. *Doctrine* [online]. Doctrine, c2010 [cit. 2018-04-15]. Dostupné z: <https://www.doctrine-project.org/projects/doctrine-orm/en/2.6/tutorials/getting-started.html>.
- Font awesome. *Font awesome* [online]. Fonticons, c2017 [cit. 2018-04-15]. Dostupné z: <https://fontawesome.com>.
- HAUSER, Marianne, Tobias HAUSER a Christian WENZ. *HTML a CSS: velká kniha řešení*. Brno: Computer Press, 2006. ISBN 80-251-1117-2.
- jQuery: What is jQuery?. *jQuery* [online]. jQuery team, c2018 [cit. 2018-04-15]. Dostupné z: <https://jquery.com>.
- NARAMORE, Elizabeth. *Vytváříme webové aplikace v PHP5, MySQL a Apache*. Brno: Computer Press, 2006. ISBN 80-251-1073-7.
- PEJŠA, Jan. Co je Cross-Site Request Forgery a jak se mu bránit. *Zdroják.cz* [online]. Devel, c2012-2018, 24. 11. 2008 [cit. 2018-04-15]. Dostupné z: <https://www.zdrojak.cz/clanky/co-je-cross-site-request-forgery-a-jak-se-branit>.
- POTENCIER, Fabien. What is Symfony2?. *Fabien Potencier* [online]. October 25, 2011 [cit. 2018-04-15]. Dostupné z: <http://fabien.potencier.org/what-is-symfony2.html>.
- RUDERMAN, Jesse. Same-origin policy. *MDN Web Docs* [online]. Mozilla, c2005-2018, Mar 2, 2018 [cit. 2018-04-15]. Dostupné z: [https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin\\_policy](https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy).
- Symfony: Controller. *Symfony* [online]. SensioLabs, c2010-2018 [cit. 2018-04-15]. Dostupné z: <https://symfony.com/doc/3.4/controller.html>.

Symfony: Service Container. *Symfony* [online]. SensioLabs, c2010-2018 [cit. 2018-04-15]. Dostupné z: [https://symfony.com/doc/3.4/service\\_container.html](https://symfony.com/doc/3.4/service_container.html).

Symfony: The Bundle System. *Symfony* [online]. SensioLabs, c2010-2018 [cit. 2018-04-15]. Dostupné z: <https://symfony.com/doc/3.4/bundles.html>.

Twig. *Twig* [online]. SensioLabs, c2010-2018 [cit. 2018-04-15]. Dostupné z: <https://twig.symfony.com>.

## **Přílohy**

Všechny přílohy a jejich součásti jsou dostupné na přiloženém CD.

*Příloha A* Zdrojový kód rezervačního systému.

*Příloha B* Kompletní verze datového modelu.