FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Informační systém pro podporu setkávání komunity Taizé.
Student:	Tomáš Jakl
Vedoucí:	Mgr. Martin Podloucký
Studijní program:	Informatika
Studijní obor:	Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce zimního semestru 2019/20

Pokyny pro vypracování

Analyzujte stávající stav informačního systému pro podporu každoročního celosvětového setkání komunity Taizé.

Navrhněte a implementujte rozšíření tohoto systému o:
1) správu ubytování,
2) správu alokace účastníků do územních celků,
3) správu hromadné dopravy,
4) správu uživatelů a omezení přístupu do systému,
5) správu návštěvníků a dokumentů územních celků,
6) modul pro logování změn nezávislý na tomto systému.

Aktualizujte aplikaci na framework Symfony 3 a otestujte funkčnost implementace nových požadavků.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 6. března 2018

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Informační systém pro podporu setkávání komunity Taizé.

Tomáš Jakl

Katedra Softwarového Inženýrství
Vedoucí práce: Ing. Martin Podloucký

29. června 2018

Poděkování

V první řadě bych chtěl poděkovat mé manželce za neustálou podporu. Dále pak vedoucímu práce Ing. Martinu Podlouckému za bezproblémové vedení mé práce a v neposlední řadě bratru Andrášovi z komunity Taizé za dobrou spolupráci na celém projektu.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 29. června 2018 .....

Odkaz na tuto práci

Jakl, Tomáš. *Informační systém pro podporu setkávání komunity Taizé.*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.

Abstrakt

Cílem této práce je analýza, návrh a implementace druhé části webové aplikace pro podporu celoevropského komunitního setkání. Tato druhá část rozšiřuje aplikaci o správu uživatelů, oprávnění, hromadné dopravy a ubytování.

Klíčová slova 2m3, Komunitní Evropské setkání, komunita Taizé.

Abstract

This work analyzes, design and implements a second part of a web application for support of Europe-wide community meetings. The second part of the application supports management of users, permissions, public transport and accommodation.

Keywords 2m3, Community European meeting, Taizé community.

Obsah

Seznam obrázků

Seznam tabulek

Úvod

2m3 je komunitní webová aplikace dostupná na adrese `2m2.taize.fr`. Tato aplikace byla vytvořena za účelem podpory Evropského setkání komunity Taizé. V provozu je již od roku 2014 a za dobu jejího fungování se stala neodmyslitelnou součástí přípravy Evropského setkání.

Vývoj byl zahájen již v roce 2013 v rámci mé první bakalářské práce, kterou jsem obhájil roku 2014. Byla vyvinuta částečná náhrada za předchozí velice zastaralý systém. Na evropském setkání v Praze 2014 byla aplikace poprvé nasazena spolu s předchozím systémem 2m2. Nemohla být nasazena samostatně z důvodu chybějících modulů.

V této bakalářské práci se zaměřím na analýzu stávajícího řešení. Zhodnotím předchozí návrh aplikace, kvalitu použitého kódu a použitých knihoven a modulů třetích stran.

Dalším krokem pak bude oprava chyb v systému a odtržení částí kódu, tak aby jej bylo možné znovupoužít v jiné aplikaci. Jedná se především o systém logování změn.

V poslední řadě navrhnu, implementuju a otestuju chybějící části, tak aby bylo aplikaci možné použít bez jakékoliv další podpory.

Evropské setkání

Evropské setkání je pořádáno francouzskou komunitou Taizé. Každý rok v jiném evropském městě. Je určeno pro mladé lidi do věku 35 let. Účast osob se velice liší rok od roku. Vždy je ale počítáno s více než 10 tisíci i v místech východní Evropy. [1]

Velice důležitou roli hraje angažovanost a ochota místních lidí spolupracovat na celkové přípravě. Pomoci mohou v několika oblastech, od organizování vítacích míst a dopravy až po nabídnutí přístřešku účastníkům. Ve většině případů jsou lidé ubytovaný v rodinách. Není to ovšem samozřejmost

a pro zbytek účastníků je nutné zajistit hromadné ubytování ve spolupráci se školami nebo městem samotným.

Kapitola 1

Cíl práce

Na této webové aplikaci pracuji již od roku 2013. Z počátku bylo velice těžké shromáždit veškeré informace a případy užití, neboť přípravný tým se mění každým rokem, a tím pádem i požadavky na systém. Velkou část dat jsem shromážďoval ze staré aplikace, která již dávno neplnila svůj účel. Jediným kontaktem s komunitou Taizé byl bratr Andráš. S přípravným týmem jsem se tedy za celou dobu neměl možnost setkat. Proto nebylo možné detailně analyzovat celou problematiku přípravy Evropkého setkání.

Aplikace 2m3, která je výsledkem mé předchozí bakalářské práce, byla navrhnuta pro požadavky evropského setkání v Praze. Kvůli nedostatku informací a nedostatku komunikace s přípravným týmem jsem nekladl velký důraz na dynamičnost celé aplikace. Prvním cílem této práce je proto identifikovat požadavky, které se mění každým rokem a dokázat navrhnout aplikaci tak, aby ji bylo možné lehce přizpůsobit potřebám přípravného týmu.

Taizé je komunita skládající se z více než 200 bratrů z celého světa, žijící v malé vesnici na jihu Burgundska. Na přípravě jejích aktivit spolupracuje nemalá skupina stovky dobrovolníků. V posledních letech tato komunita klade velký důraz na rozvoj převážně webových aplikací pro zjednodušení celkového chodu. Druhým cílem je tedy identifikovat části aplikace, které jsou užitečné k dalšímu vývoji systémů a vytvořit znovupoužitelné moduly. A tím potlačit duplikaci kódu napříč všemi aplikacemi.

Posledním cílem je rozšíření systému o dodatečné funkce tak, aby mohl být použit samostatně bez jakékoliv další podpory.

1. **Správa uživatelů a rolí**

   Modul pro správu uživatelů a rolí byl vyvíjen dobrovolníkem z Taizé. Jedním z cílů předchozí bakalářské práce bylo napojení se na tuto komponentu a její nakonfigurování pro potřeby našich požadavků. Od této doby se ovšem dále nepokračovalo v jejím rozšíření s důvodu velkých závislostí na modulech třetích stran. Dalším problémem byla závislost na aplikaci 2m3. Cílem je vytvořit kompletně novou komponentu pro

správu uživatelů, rolí a uživatelských skupin. Dále je nutné vytvořit rozhraní pro přihlašování a registraci uživatelů a správu uživatelského profilu.

2. **Omezení přístupu k systému**

   Omezení přístupu k systému bylo v předchozí verzi aplikace zajištěno pomocí rolí. Role byly fixně definovány v kódu. Počítalo se tedy s tím, že bude existovat pouze několik typů uživatel, které mají každý rok stejná práva. To se po evropském setkání v Praze ukázalo jako špatný předpoklad. Při analýze systému se došlo k závěru, že je potřeba mnohem více rolí. Dalším požadavkem pak bylo dynamické definování přístupu ke každé části aplikace. Je tedy nutné přepracovat systém k omezení přístupu a navrhnout nové řešení tak, aby jej později bylo možné použít v jiné aplikaci.

3. **Logování změn**

   Jednou z důležitých částí aplikace 2m3 je možnost logování veškerých změn v systému. Modifikace musí být dostupné uživateli k nahlédnutí. V prvním návrhu bylo počítáno s logováním jen některých změn. Proto došlo k implementaci, která je existenčně závislá na jednotlivých třídách. Tedy každá třída má svou historickou variantu. I zde se pak ukázalo, že je nutné počítat se změnou požadavků a tak ani v tomto ohledu aplikaci není možné rozšířit bez modifikace tohoto modulu. V první řadě je potřeba navrhnout komponentu tak, aby bylo snadné rozšířit logování o další jednotky. Dále pak odtrhnout tento modul od aplikace a tím zajistit snadné znovupoužití tohoto kódu v jiném projektu.

4. **Akceptace změn**

   Evropské setkání je závislé na pomoci dobrovolníků. Je jich celá řada a proto není možné spolehnout se pouze na systém logování změn. Přípravný tým chce mít typicky přehled o všech změnách, které provede v systému dobrovolník s menší sadou práv. Tyto akce uživatele musí být akceptovány, aby se staly reálnými. Akceptace změn má doplnit komponentu logování, tak aby bylo možné modifikace vrátit do předchozího stavu nebo je označit za potvrzené.

5. **Komponenta dopravy**

   Přípravný tým město typicky rozdělí do několika částí, kam jsou pak účastníci přiřazeni dle národnosti nebo jazyka, kterým hovoří. Komunita také spolupracuje s institucemi, které zajišťují dopravu v oblasti konání mýtinku. Každý den večer, v době setkání, se všichni účastníci scházejí na typicky dvou nebo třech místech ve městě. Je tedy velmi důležité zajistit spolu s institucemi bezproblémový chod veškeré hromadné dopravy.

Systém má v tomto ohledu pomoci se správou všech linek a informací o vybraných cestách, které pak budou součástí itineráře k setkání.

6. **Rozšíření o přidání vlastních položek**

   Osoby nebo organizace mají fixní počet atributů, které může uživatel použít k uchování informací. V nové verzi systému by mělo být možné nadefinovat vlastní atributy k vybraným třídám dynamicky bez nutnosti změny kódu.

7. **Ubytování účastníků**

   Přípravný tým se snaží nalézt co nejvíce míst k ubytování účastníků v rodinách. To ale není možné zajistit ve všech městech z důvodu velké návštěvnosti a nebo neochoty místních lidí spolupracovat. Když nastane tato situace je nutné dohodnout spolupráci se školami. Počet míst v rodinách je třeba evidovat stejně jako hromadné ubytování. Proto je nutné navrnout část aplikace pro snadnou správu ubytovacích míst.

8. **Itinerář setkání**

   Účastnící setkání jsou, jak už jsem se zmínil, rozděleni do několika částí města. Je důležité, aby každý z nich, měl informace o celkovém programu mýtinku, podle toho, kam byl rozdělen. Systém musí být schopen shromáždit veškeré informace a vygenerovat výsledné pdf soubory, které je možné vytisknout před vítacím dnem.

Kapitola 2

Analýza

2.1 Současný stav

Aplikace 2m3, která je výsledkem mé předchozí bakalářské práce byla napsána v jazyce PHP ve verzi 5.6, s použitím populárního frameworku Symfony, v dnes již zastaralé verzi 2.3.[2] K ukládání dat je použita MySQL databáze.

Symfony je soubor PHP komponent, též použitelných mimo kontext frameworku. V dnešní době je na těchto knihovnách závislá celá řada jiných projektů.[3] Symfony je také kompletní webový framework, který je vyvíjen již od roku 2005. Z počátku byl napsán pro potřeby společnosti Sensio Labs. Postupem času si ovšem začal budovat místo v PHP komunitě, která byla hlavním motorem dalšího vývoje.[4] Na podzim roku 2017 byla vydána již čtvrtá verze, na které se podílelo více než 300 přispěvovatelů.[5]

Již v úvodu jsem se zmínil, že návrh aplikace byl ovlivněn nedostatkem informací o evropském setkání a celkovém fungování komunity. Nekladl se tedy velký důraz na znovupoužití a flexibilitu. Byl vytvořen systém, který není snadné nastavit podle požadavků přípravného týmu. Mezi další problémy aplikace patří:

- Velká část logiky se nalézá přímo ve třídách kontroleru. Dochází tedy k míšení business logiky aplikace s ostatními částmi. Řadič má navíc přístup přímo k dependency injection kontejneru, tudíž má veškeré prostředky na to, aby vyřešil jakýkoliv problém. To ztěžuje rozšíření z několika důvodů:
  - Je velice težké otestovat aplikaci jednotkovými testy.
  - Opakující se business logika napříč řadiči.
  - Aplikace je nepřehledná a težko se v ní orientuje.[6]
- V době, kdy byla aplikace napsána, neexistovalo mnoho osvědčených nástrojů pro práci s CSS a JavaScriptovými knihovnami. Proto byla

použita PHP knihovna assetic, vyvíjená týmem Symfony. S pomocí přídavných knihoven dokázala minimalizovat a zoptimalizovat assety, které je pak možné přidat do šablony pomocí tagů šablonovacího systému TWIG. Bohužel toho neuměla více. Chyběla například podpora JavaScriptových modulů nebo podpora kompilace skriptů, které jsou překladatelné staršími webovými prohlížeči. Na vývoji knihovny assetic se přestalo aktivně pracovat, především z důvodu velkého rozvoje jazyka JavaScript a jeho nástrojů.[7]

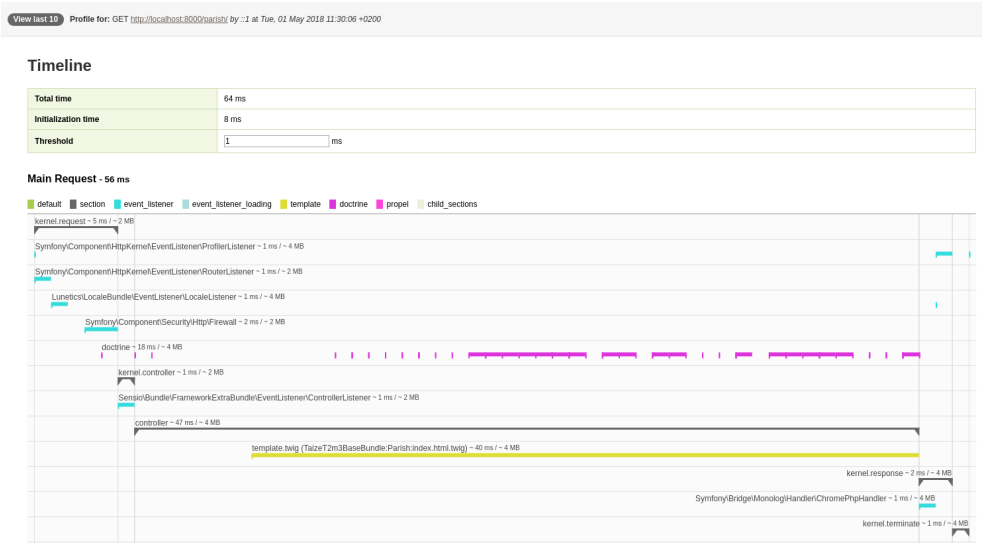
- Pro práci s databází MySQL byla použita knihovna Doctrine 2. Jedná se o ORM knihovnu, která mapuje řádky databázových tabulek na PHP objekty. Toto řešení přináší mnoho výhod. Mezi některé z nich patří:
  - Aplikace není závislá na implementaci relační databáze.
  - Pro psaní dotazů nad databází má Doctrine jazyk DQL, který je obnoží OQL. Je ovlivněn jazyky Hibernate Query Language nebo Java Persistence Query Language. Výhodou DQL oproti SQL je jeho čitelnost a vztah k objektům modelu aplikace.[8]
  - Řádky databáze jsou namapovány na objekty, tudíž je možné přistupovat k jiným objektům skrz jejich metody. V případě, kdy ještě není objekt v paměti, doctrine automaticky pošle dotaz do databáze a nahraje jej.

  Při vývoji aplikace se však zapomnělo na stinné stránky ORM, které při špatném použití dokáží rapidně ovlivnit rychlost systému. Díky tzv. línému nahrávání je možné snadno získat objekty, které jsou v relaci s jinými. Toto řešení na každý pokus o získání objektu, vyvolá dotaz do databáze a tak může dojít k velkému nahrávání objektů do paměti.[9] Na obrázku 21 je vidět zpomalení při načítání seznamu farností a počet spuštěných dotazů. Pro každý objekt třídy `parish` se též načítá objekt třídy `organization`.

2.1.1 Base bundle

Base bundle je modulem aplikace, který měl fungovat jako knihovna pro znovuopakující se požadavky ve vývoji aplikací komunity Taizé. S podporou této knihovny byl naimplementován systém 2m3. Tento modul měl zajistit rozhraní pro správu uživatelů, rozhraní pro správu modulů, služby pro práci s uživateli a abstraktní třídy, které měly umožnit snazší vývoj. Byl velice ovlivněn frameworkem Symfony 1 a tak některé věci řešil zastaralým způsobem.

- Obsahoval skupinu skriptů napsaných v bashi pro práci s databázovými migracemi, které byly generovány pro každý modul zvlášť. To přineslo spoustu problémů napříč aplikacemi. Migrace byly závislé na konkrétní databázi, tudíž její aktualizace přinášely problémy s kompatibilitou.



Obrázek 21: Doctrine lazy loading

- Base Bundle obsahuje několik tříd, které mají přístup ke globální proměnné, obsahující jádro aplikace a tak mají přístup k celému DI kontejneru.
- Dále modul obsahoval CSS a JavaScript knihovny, které se liší podle požadavků každého projektu.

Autor se snažil vytvořit knihovnu, která svým obsahem rapidně zasahuje do architektury celé aplikace a tak nebylo možné v jejím pokračování.

2.2 Architektura aplikace

Architektura aplikace je silně ovlivněna vybraným frameworkem Symfony. Je rozdělena do 3 vrstev. Vrstvami jsou model, řadič a pohled.

- **Model**

  Model se stará o chování a manipulaci dat domény aplikace. Odpovídá na požadavky o stavu informací (typicky z vrstvy pohledu) a reaguje na instrukce o změně stavu (požadavky pocházející z řadiče).

- **Řadič**

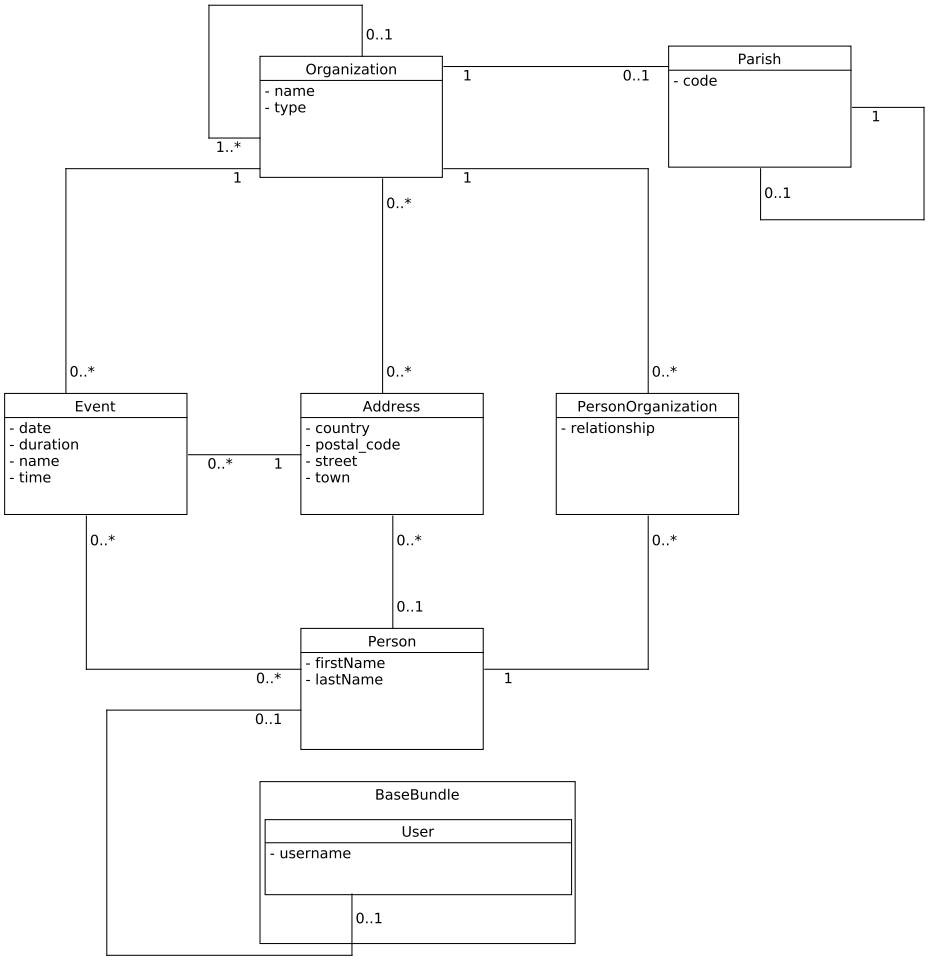
  Řadič je vrstvou, která reaguje na vstup uživatele, informuje model a aktualizuje pohled, který je výstupem pro uživatele.

- **Pohled** Tato vrstva se stará pouze o zobrazení informací. [10]

Ovšem v některých částech aplikace se jedná spíše o architekturu se dvěma vrstvami. Model a řadič je spojen do jedné vrstvy. Hlavním důvodem je použití modulu Base Bundle.

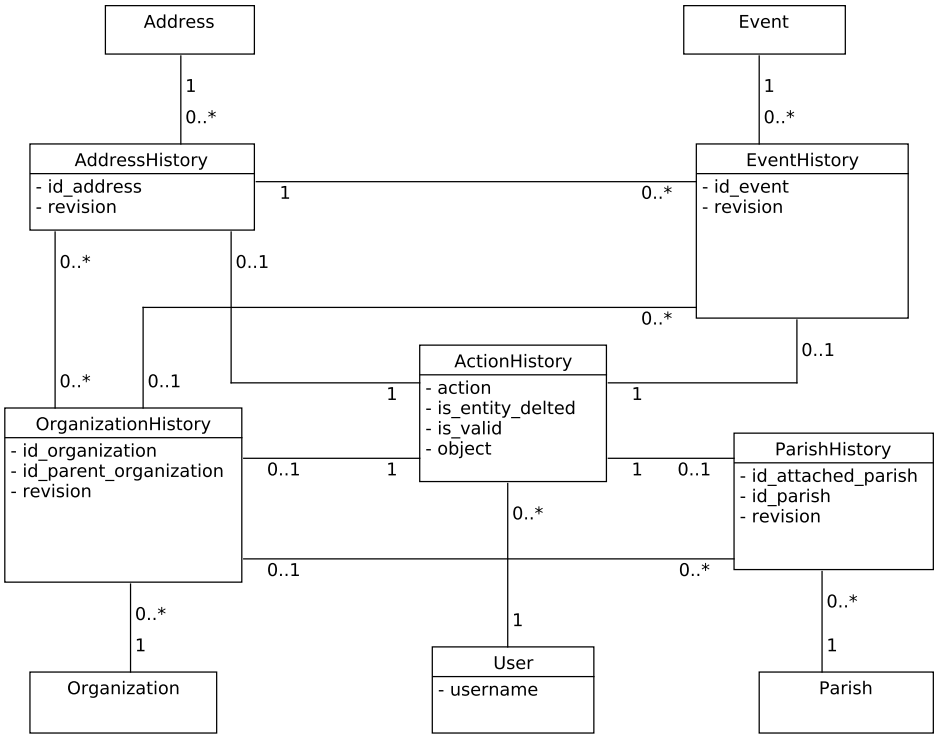
2.3 Doménový model

Doménový model jsem pro přehlednost rozdělil do dvou částí. Na obrázku 22 je znázorněn model jádra aplikace a na obrázku 23 je zobrazen model týkající se historie změn v systému.



Obrázek 22: Doménový model

Před rozšířením aplikace je nutné změnit celkový model historie změn. Z obrázku 23 je patrné, že jakýkoliv zásah do modelu jádra je i zásahem do



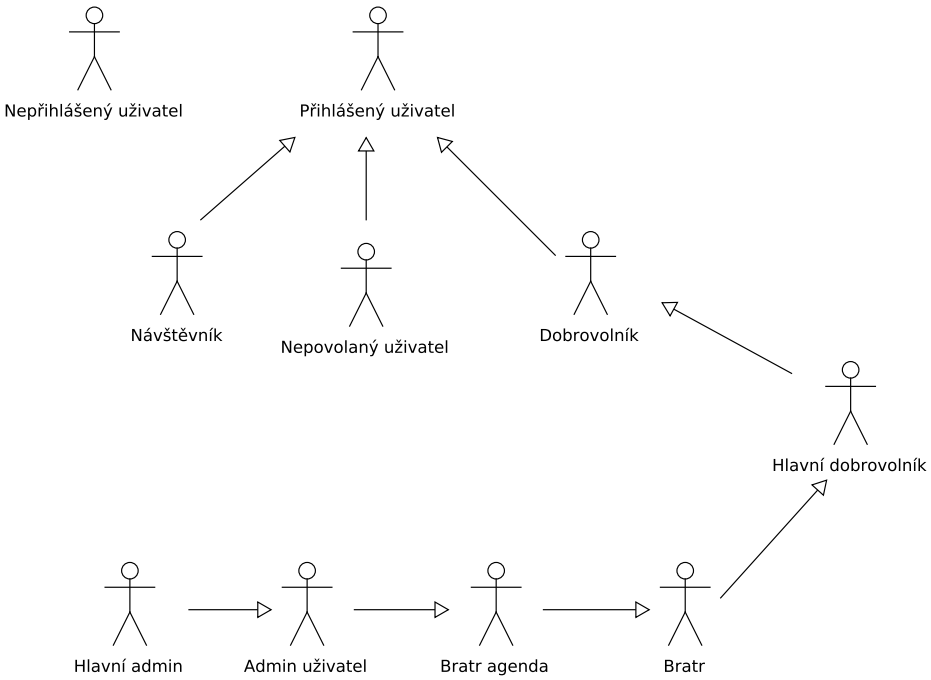
Obrázek 23: Doménový model historie

historické části. Při jakékoliv změně tříd je nutné dbát na to, aby historické třídy byly v synchronizaci. K vytvoření nové třídy je třeba vytvořit třídu historickou.

Dalším problémem, který nelze jednoduše vyřešit bez změny modelu je logování změn asociací. Při změně adresy události se vytvoří historický objekt, který je třeba spojit s historickým objektem události. Při přiřazení jiné adresy k události je pak nutné dbát na to aby historické objekty byly aktualizovány. Celková rekonstrukce a udržení konzistence těchto změn je velice složité.

2.4 Model rolí

Na obrázku 24 je zobrazen model rolí předchozí verze. Tento model vyšel z požadavků evropského setkání v Praze, při kterém byl prvně systém nasazen. Na dalším setkání byly požadavky změněny a tak se pravomoce některých rolí měly silně překrývat. Změna požadavků znamená velký zásah do tohoto modelu.



Obrázek 24: Model rolí

2.5 Funkční požadavky

1. Správa uživatelů
   - Rozšířit registraci uživatelů o možnost definování osobních dat a možnost přiřazení registrace ke konkrétní organizaci.
   - Přidat rozhraní pro potvrzení a odmítnutí registrace uživatelů.
   - Přidat možnost zobrazení profilu přihlášeného uživatele.
   - Přidat rozhraní pro správu uživatelů, správu jejich rolí a oprávnění.
   - Přidat rozhraní pro správu uživatelských skupin.
   - Přidat funkci, která umožní přihlášení do systému jako jiný uživatel pomocí přihlašovacích dat administrátora.
2. Farnosti
   - Rozšířit farnost o možnost evidence extra atributů.
   - Rozšířit farnost o možnost evidence územních celků.
   - Rozšířit farnost o možnost definování adres a telefonních čísel k vítacím dnům.

- Přidat rozhraní pro správu účasti konkrétní farnosti na setkání.
- Přidat rozhraní pro správu dobrovolníků a bratrů zodpovědných za danou farnost.
- Přidat rozhraní pro správu událostí pro konkrétní farnost.
- Přidat rozhraní pro správu hromadných ubytování v konkrétní farnosti.
- Přidat rozhraní pro správu nalezených míst k ubytování v rodinách.
- Přidat rozhraní pro správu možných cest do farnosti.
- Přidat rozhraní pro správu map důležitých k tisknutým itinerářům.
- Přidat možnost generování přehledu farnosti ve formátu pdf.
- Přidat přehled farnosti obsahující veškerá důležitá data. Tyto data jsou základní atributy farnosti, adresy a extra atributy obsahující telefonní čísla, vybrané atributy (adresy a telefonní čísla) k vítacímu dni, data o možnosti spolupráce, kontaktní osoby ve farnosti, nalezená místa ve farnosti a itineráře dopravy do farnosti.
- Přidat možnost vyhledat farnost pomocí jména, kódu a adres.
- Přidat rozhraní pro potvrzení změn, uvedených do systému dobrovolníky.
- Přidat statistiky o nalezených místech v rodinách s možností seskupení pomocí dobrovolníků nebo farností.
- Přidat přehled o přípravě a připravenosti každé farnosti. Tento přehled musí obsahovat informace o spolupráci, počtu nalezených míst, informace o tom zda je farnost připravena na vítací den vyhrazený pro dobrovlníky, informace o událostech, které by se měli konat v každé farnosti a informace o tom zda farnost má přiřazenou adresu, telefon, hlavní kontakt a vytvořený účet v systému.
- Přidat přehled o změnách nalezených míst. Informace musí obsahovat kód farnosti, seznam dobrovolníků zodpovědných za tuto farnost, datum změny, starou a novou hodnotu, uživatelské jméno uživatele, který danou změnu vykonal a rozdíl mezi starou a novou hodnotou. Tento přehled musí být možné filtrovat pomocí některých z předešlích atributů.
- Přidat přehled s informacemi, kdo je zodpovědný za danou farnost. Dále pak možnost tyto informace změnit.

3. Hromadné ubytování

- Přidat seznam všech hromadných ubytování.
- Přidat možnost vyhledat ubytování pomocí jména.

- Přidat rozhraní pro správu kontaktních osob i dobrovolníků.
- Přidat rozhraní pro editaci atributů hromadného ubytování.
- Přidat rozhraní pro výběr hlavního kontaktu.

4. Lidé

- Rozšířit rozhraní pro editaci osoby o další atributy včetně dynamických atributů.
- Přidat rozhraní pro správu organizací spojených s konkrétní osobou.
- Přidat rozhraní pro správu událostí spojených s konkrétní osobou.
- Přidat rozhraní pro potvrzení změn, uvedených do systému dobrovolníky.
- Přidat možnost rozlišit lidi pomocí skupin.
- Přidat možnost zobrazení duplikací osob.
- Přidat rozhraní pro sjednocení duplikací.
- Přidat možnost vyhledat osoby pomocí jejich jmen a dalších atributů.

5. Události

- Rozšířit rozhraní pro editaci událostí o další atributy. Tyto atributy jsou typ, priorita, nutnost překladatele, čas konce události.
- Přidat možnost spojit událost s existující organizací a vybrat jednu z adres této organizace.
- Přidat možnost definovat kontaktní osobu události.
- Přidat informaci o dostupnosti osob v čase události.
- Přidat možnost vyhledat události pomocí jejich jmen nebo organizací, ve kterých události probíhají.
- Přidat možnost filtrovat události podle účastníků nebo dobrovolníků.
- Přidat možnost generování souborů ve formátu pdf obsahující informace o událostech konaných v zadaném termínu.
- Přidat možnost generování souboru ve formátu pdf obsahující informace o vybrané události.
- Přidat přehled o počtu událostí v každém dnu od počátku přípravy.
- Přidat přehled o dostupnosti dobrovolníků a bratrů ve vybraném termínu.
- Přidat možnost změnit dostupnost přípravného týmu. Tyto lidé jsou označeni, tedy neměli by být přiřazeny k žádné události.

6. Hromadná doprava
    - Přidat rozhraní pro správu zastávek.
    - Přidat rozhraní pro správu linek.
    - Přidat rozhraní pro správu spojení.
    - Přidat rozhraní pro správu přestupů ve vybraných itinerářích.
    - Přidat možnost generování souborů ve formátu pdf obsahující informace o vybraném itineráři.
    - Přidat přehled o všech itinerářích a možnost jejich filtrování pomocí atributů itineráře. Dále pak možnost tyto informace vygenerovat do pdf souboru.
    - Přidat statistiky o předpokládaném zatížení linek pomocí informací zadaných v itinerářích farností.
7. Administrace
    - Přidat přehled o všech změnách v systému.
    - Vytvořit rozhraní pro zobrazaní jednotlivých změn v systému s možností potvrdit tyto změny, jedná-li se o změnu vytvořenou dobrovolníkem v organizaci.
    - Přidat rozhraní pro správu překladů textů v systému.
8. Ostatní
    - Přidat možnost generování itinerářů s informacemi o průběhu setkání. Tyto informace pak obdrží každý účastník.
    - Přidat možnost generování souboru ve formátu pdf obsahující informace nutné k hlavnímu přípravnému setkání ve farnosti.
    - Přidat možnost vyhledat osoby, organizace a farnosti dle jejich atributů.
    - Přidat rozhraní pro správu vlastních SQL dotazů nad databází.

2.6 Nefunkční požadavky

- Dostupnost aplikace ze všech moderních prohlížečů.
- Navrhnout nové požadavky tak, aby byla aplikace lehce rozšiřitelná.
- Přepsat aplikaci tak, aby části kódu, nezávislé na systému, byly možné použít v jiném projektu.
- Aktualizovat aplikaci tak, aby byla kompatibilní s jazykem PHP verze 7.2 a frameworkem Symfony verze 4.1.

- Vytvořit přehledné a intuitivní grafické rozhraní.
- Podpora více jazykových verzí.
- Bezpečnost.
- Spolehlivost aplikace a ošetření nesprávných vstupů.

Kapitola 3

Návrh

Při návrhu informačního systému jsem se snažil co nejvíce odprostit od předchozí verze, která nebyla snadno rozšiřitelná z důvodů popsaných v druhé kapitole. Tento požadavek považuji za stěžejní, neboť přípravný tým se každým rokem mění a tak se mění i kladené požadavky.

3.1 Diagram modulů

Na obrázku 31 je vidět propojení modulů se zbytkem systému. Aplikaci jsem rozdělil do několika částí tak, aby nedocházelo k duplikacím kódu napříč projekty. Každý modul by měl být použitelný i mimo kontext této aplikace.

3.1.1 Modul logování změn

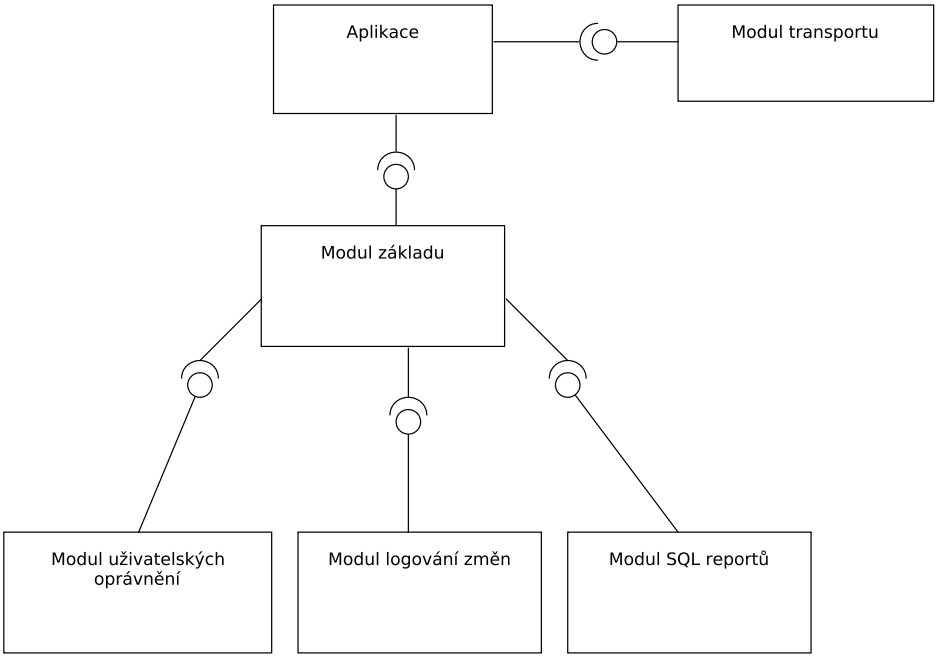
Hlavní zodpovědností tohoto modulu je ukládání změn databáze provedených uživatelem zpět do úložiště. Dále pak možnost vrácení změn do předchozího stavu, kontrola dat vložených uživatelem s minimální sadou práv a jednoduché uživatelské rozhraní pro přehled a správu historických změn.

3.1.2 Modul základu

Modul základu se stará o snadnou integraci dalších balíčků popsaných v této kapitole. V případě, že některý z nich není v projektu vyžadován, jeho funkce nejsou aktivovány. Dále obsahuje abstraktní třídy a služby, které slouží k redukci kódu a duplikací v aplikaci.

3.1.3 Modul hromadné dopravy

Tato komponenta má za cíl rozšířit stávající aplikaci o správu hromadné dopravy. Dobrovolník zadá do systému typicky 2 nebo 3 trasy, které vedou do



Obrázek 31: Diagram modulů

dané farnosti. Na přípravném týmu a dopravním podniku města pak leží zodpovědnost vybrat trasu tak, aby nedošlo ke kolapsu hromadné dopravy. Vybraná trasa je dále zobrazena v itineráři, který obdrží účastník mířící do dané farnosti.

3.1.4 Modul SQL reportů

Modul reportů rozšiřuje systém o možnost definice vlastních dotazů nad databází. Výsledek je pak možné sdílet s libovolnou uživatelskou skupinou. Znalý uživatel (admin) tedy není závislí pouze na vývojáři k získání zajímavých reportů.

3.1.5 Modul uživatelských oprávnění

V předchozí verzi systému byl přístup uživatele omezen dle jeho role. Pomocí tohoto modulu je možné definovat oprávnění k určité části systému bez nutnosti jakékoliv další konfigurace. Oprávnění lze definovat v konfiguraci aplikace nebo konfiguraci konkrétního modulu. Tyto oprávnění jsou pak nahrány do databáze a při spuštění systému do cache.

3.2 Databázový model

Vzhledem k tomu, že systém je určen pro práci s velkým množstvím dat, považuji databázi za ztěžejní část. Nevhodně navržená struktura může mít v konečném důsledku kritický dopad na rychlost odezvy u většiny dostupných funkcí. Při jejím návrhu jsem se snažil klást velký důraz na možnost budoucího rozšíření nebo změny.

Databázový model jsem pro přehlednost rozdělil do několika sekcí podle modulů. Vzhledem k tomu, že veškerý kód je napsán v angličtině, budu jména tabulek a atributů psát v tomto jazyce.

3.2.1 Modul logování změn

Na obrázku 32 je zobrazen model databáze komponenty logování změn zachycující tabulky a relace popsané níže.

3.2.1.1 Revision

Tato tabulka uchovává záznamy o tom, kdo a kdy vykonal změnu v systému. `Revision` má vazbu 1:n na tabulku `log_entry` a napovinnou vazbu 1:n na tabulku `additional_log_entry`. Tyto vazby reprezentují počet řádků tabulek, které byly změněny v rámci jedné interakce uživatelem.

3.2.1.2 Log_entry

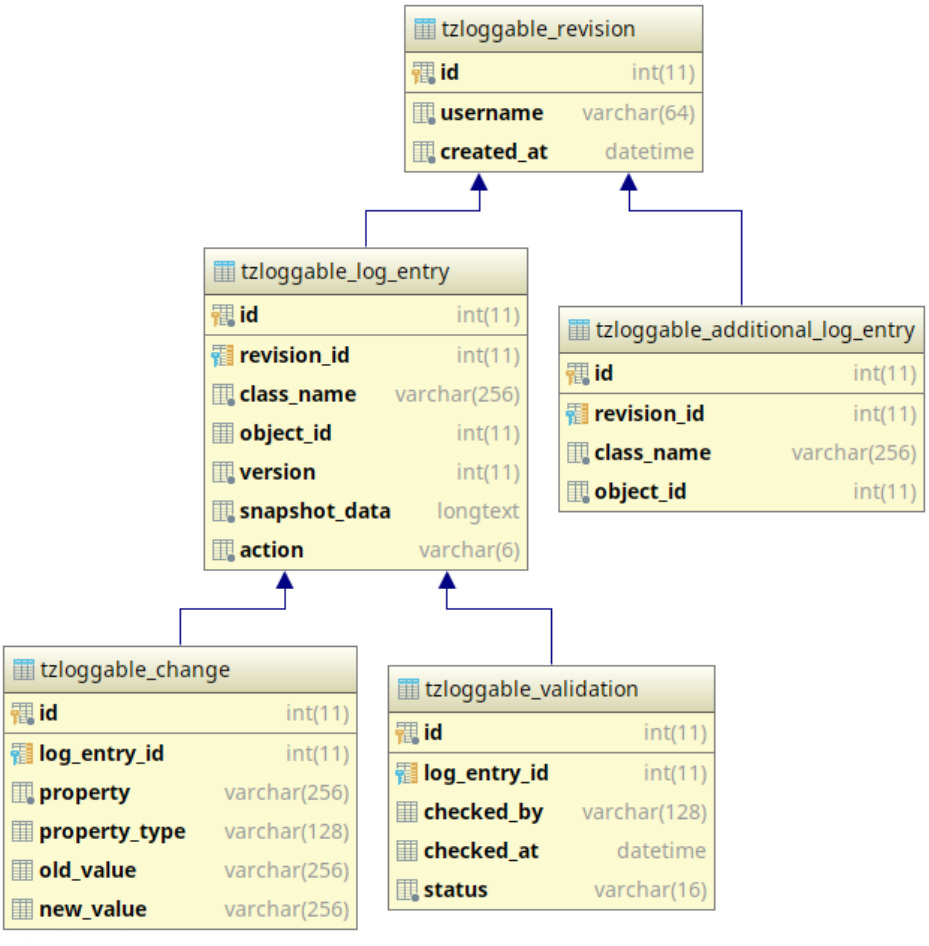
Tabulka `log_entry` uchovává informaci o tabulce, jejíž řádek byl změněn a informaci o konkrétní modifikaci. Obsahuje identifikátor konkrétního řádku, verzi změny, pole obsahující nově zadaná data a typ změny. Tento typ může nabývat hodnot: `insert`, `update` nebo `delete`. `Log_entry` má vazbu 1:n s tabulkou `change` a nepovinnou vazbu 1:1 s tabulkou `validation`. Tyto vazby přidávají rozšiřující informace o daných modifikacích.

3.2.1.3 Additional_log_entry

Tato tabulka uchovává informace o řádcích tabulek, u kterých v procesu změny nedošlo k modifikaci, a které jsou spojeny s řádky jiných tabulek obsahujících změnu. Příkladem může být změna dat adresy. V tabulce `log_entry` je uchována modifikace, ale nemáme informaci o tom, že adresa náleží konkrétní organizaci. Tedy není možné zobrazit seznam všech změn organizace. V tomto případě se vytvoří řádek v tabulce `additional_log_entry`.

3.2.1.4 Change

Tabulka `change` uchovává informace o změně atributu jedné položky. `Property` obsahuje název atributu, `property_type` obsahuje typ atributu, `old_value` a `new_value` uchovává informaci o staré a nové hodnotě.



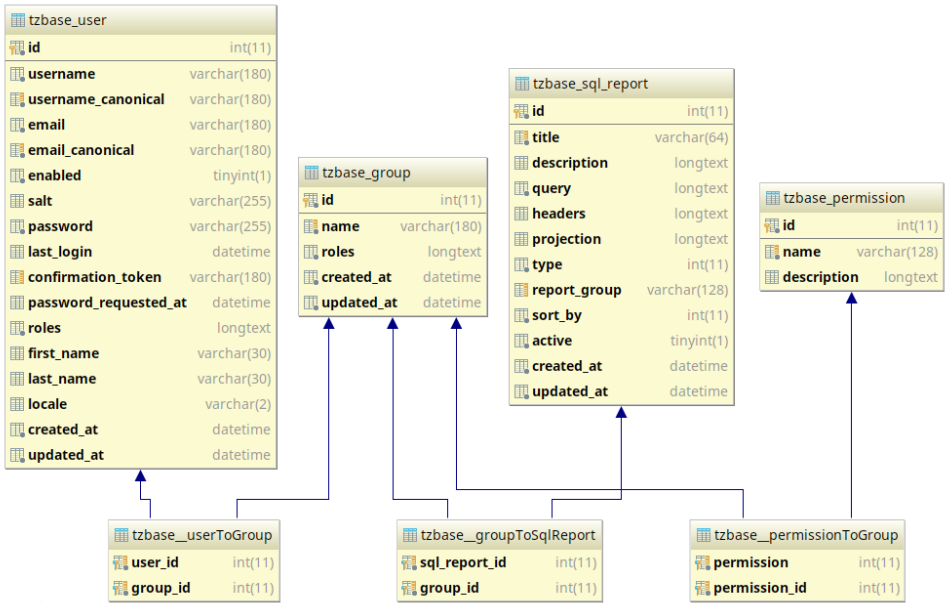
Obrázek 32: Databázový model modulu logování změn

3.2.1.5 Validation

Tato tabulka uchovává informaci o tom, byla-li změna provedena pověřenou osobou nebo dobrovolníkem s menší sadou práv. Těmito informacemi jsou jméno uživatele, a čas kdy potvrdil nebo zamítl modifikaci. Dále pak status, zda-li byla změna potvrzena, odmítnuta a nebo se čeká na rozhodnutí.

3.2.2 Modul základu

Na obrázku 33 je znázorněn databázový model modulu základu. V tomto modelu jsou zobrazené i tabulky definované v komponentách popsaných v sekci 3.1.



Obrázek 33: Databázový model základního modulu

3.2.2.1 User

Tato tabulka uchovává informace o uživatelích. Počet atributů je ovlivněn použitou knihovnou usnadňující registraci a správu uživatelského účtu. Dalšími atributy jsou `locale`, `first_name`, `last_name`, `created_at` a `updated_at`. `User` má nepovinnou vazbu n:m s tabulkou `group`.

Atribut	Volitelnost	Popis
locale	Nepovinné	Nastavení jazyka prostředí.
first_name	Nepovinné	Křestní jméno.
last_name	Nepovinné	Příjmení.
created_at	Povinné	Čas a datum vytvoření uživatele.
updated_at	Povinné	Čas a datum změny uživatele.

Tabulka 31: Atributy tabulky `user`.

3.2.2.2 Permission

Definice této tabulky se nalézá v modulu uživatelských oprávnění. Uchovává informace o oprávněních, které jsou definovány atributy `name` a `description`. Má nepovinnou vazbu n:m na tabulku `group`. Tato vazba reprezentuje sadu oprávněních jednotlivých skupin.

Atribut	Volitelnost	Popis
name	Povinné	Název oprávnění.
description	Nepovinné	Popis oprávnění.

Tabulka 32: Atributy tabulky `permission`.

3.2.2.3 Sql_report

Definice `sql_report` tabulky se nalézá v modulu SQL reportů. Tato tabulka uchovává informace o dotazech nad databází. Zahrnuje vazbu n:m s tabulkou `group`. Pomocí této vazby lze definovat, která skupina má oprávnění zobrazit daný report.

Atribut	Volitelnost	Popis
title	Povinné	Titulek reportu.
description	Nepovinné	Popis reportu.
query	Povinné	Dotaz ve formátu SQL nebo DQL.
headers	Nepovinné	Hlavička tabulky obsahující vybrané atributy.
type	Povinné	Typ dotazu. Nabývá hodnot SQL nebo DQL.
report_group	Nepovinné	Sekce do které report patří.
active	Nepovinné	Rozhoduje o viditelnosti reportu.
created_at	Povinné	Čas a datum vytvoření reportu.
updated_at	Povinné	Čas a datum změny reportu.

Tabulka 33: Atributy tabulky `sql_report`.

3.2.2.4 Group

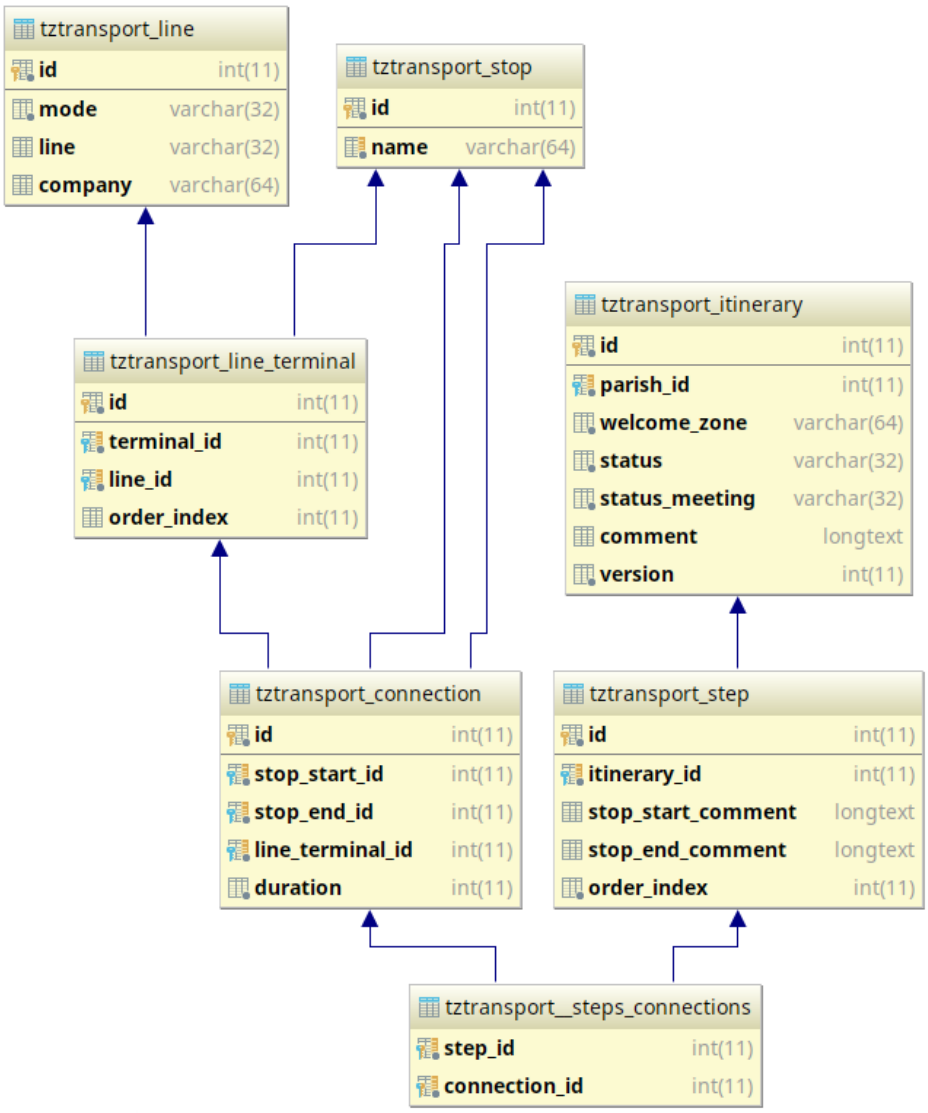
Tato tabulka ukládá informace o rozdělení uživatelů do skupin. Lze tedy definovat oprávnění a role v rámci skupiny.

Atribut	Volitelnost	Popis
name	Povinné	Název skupiny.
roles	Nepovinné	Pole obsahující názvy rolí.
created_at	Povinné	Čas a datum vytvoření skupiny.
updated_at	Povinné	Čas a datum změny skupiny.

Tabulka 34: Atributy tabulky `group`.

3.2.3 Modul hromadné dopravy

Na obrázku 34 je znázorněn databázový model této komponenty.



Obrázek 34: Databázový model rozšíření aplikace o hromadnou dopravu

3.2.3.1 `Stop`

Tato tabulka uchovává informaci o zastávkách. `Stop` je definována pouze jedním povinným atributem `name` a primarním klíčem `ID`.

3.2.3.2 `Line`

`Line` je tabulkou linek hromadné dopravy.

Atribut	Volitelnost	Popis
mode	Povinné	Dopravní prostředek. Př. tram.
line	Nepovinné	Identifikace linky. Př. 22.
company	Nepovinné	Společnost zajišťující dopravní prostředek.

Tabulka 35: Atributy tabulky `line`.

3.2.3.3 `Line_terminal`

`Line_terminal` je tabulkou, která uchovává informace o lince a konkrétním směru této linky. Směr linky je určen vazbou n:1 s tabulkou `stop`. Linka je definována vazbou n:1 s tabulkou `line`.

3.2.3.4 `Connection`

Tabulka spojů. Obsahuje informace o délce spoje, startovní stanici a stanici výstupní. Tyto stanice jsou definované dvěmi vazbami n:1 s tabulkou `stop`. Spojení n:1 s tabulkou `line_terminal` určuje směr a linku spoje.

3.2.3.5 `Step`

Tabulka kroků, nebo-li přestupů, nutných k přepravě z počáteční stanice do stanice požadované. Pomocí vazby n:m je možné definovat více spojů mezi stanicem.

Atribut	Volitelnost	Popis
stop_start_comment	Nepovinné	Komentář k počáteční stanici.
stop_end_comment	Nepovinné	Komentář ke konečné stanici.
order_index	Povinné	Index, který definuje pořadí kroku v itineráři.

Tabulka 36: Atributy tabulky `step`.

3.2.3.6 `Itinerary`

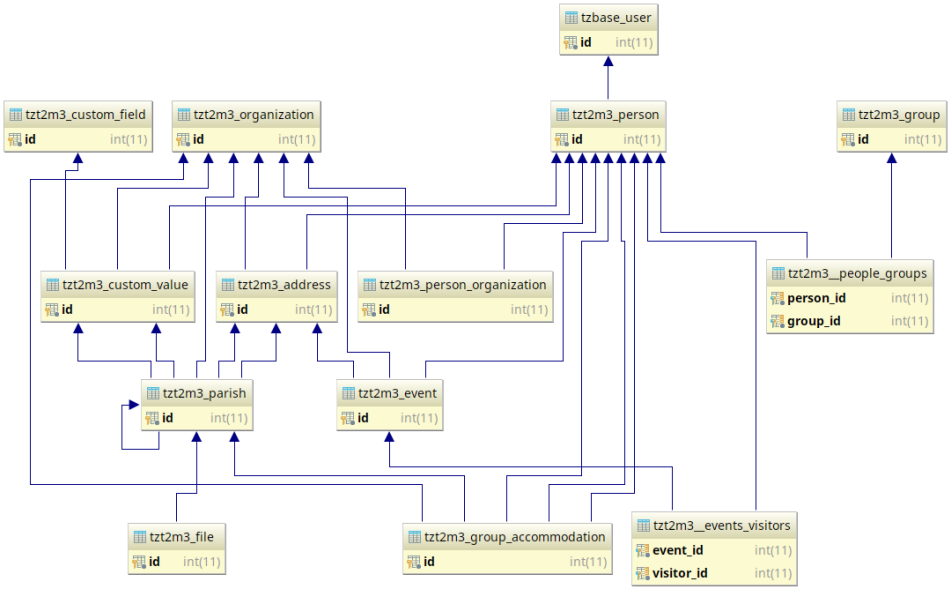
Tabulka itinerářů obsahuje informace o možných variantách dopravy mezi farností a bodem shromáždění. Je spojena s tabulkou `parish` vztahem n:1 a pomocí vazby 1:n také s tabulkou `step`.

Atribut	Volitelnost	Popis
welcome_zone	Povinné	Místo shromáždění, kam kroky v itineráři směřují.
status	Povinné	Informace o tom, zda byl itinerář vybrán přípravným týmem.
status_meeting	Povinné	Informace o tom, zda bude místo shromáždění aktivní pro vybranou farnost.
comment	Nepovinné	Komentář specifikující další informace o itineráři.

Tabulka 37: Atributy tabulky `itinerary`.

3.2.4 Databázový model rozšíření aplikace

Na obrázku 35 je zobrazen databázový model jádra aplikace. Atributů tabulek je opravdu mnoho a proto jsem je v modelu nezobrazil pro přehlednost. Jelikož jsem zde vycházel z návrhu staré verze informačního systému, budu popisovat jen tabulky, atributy a vazby, které jsou součástí této práce.



Obrázek 35: Databázový model rozšíření aplikace

3.2.4.1 `Custom_value`

Tabulka extra atributů. Obsahuje hodnotu atributu a má povinnou vazbu n:1 s tabulkou `custom_field`. Touto vazbou je definován typ atributu.

3.2.4.2 `Custom_field`

Tabulka rozšiřuje extra atributy o jejich typ.

3.2.4.3 `File`

Tabulka `file` uchovává informaci o souborech uložených v aplikaci. Má nepovinnou vazbu 1:n s tabulkou `parish`.

Má-li řádek tabulky `file` definovaný cizí klíč, jedná se o soubor typu PNG s mapou vítacího místa farnosti. V dalších případech se jedná o jakýkoliv soubor typu PNG nebo PDF.

Atribut	Volitelnost	Popis
hash	Povinné	Automaticky generovaný klíč, který je jménem souboru v souborovém systému.
name	Povinné	Jméno souboru definové uživatelem.
note	Nepovinné	Poznámka k souboru.

Tabulka 38: Atributy tabulky `file`.

3.2.4.4 `Group_accommodation`

Tabulka uchovávající informace o hromadném ubytování. Vazbou 1:1 s tabulkou `organization` je typem organizace. Dále má nepovinnou vazbu n:1 s tabulkou `parish`. Ve farnosti může být více než 1 hromadné ubytování. `Group_accommodation` musí mít 2 kontaktní osoby pro 2 vítací dny a kontakní osobu během setkání. Tyto informace jsou zajištěny 3 vazbami n:1 s tabulkou `person`. V tabulce 39 jsou zobrazeny nejdůležitější atributy s jejich popisem.

3.2.4.5 `Group`

Tato tabulka uchovává informace o skupinách lidí.

3.3 Architektura aplikace

Architektura aplikace je silně ovlivněna vybraným PHP frameworkem Symfony, který je založen na architektuře MVC. Při návrhu tedy nezacházím do příliš velkých detailů.

3.3.1 Základní princip

Architektura je postavena na použití jednoho centrálního PHP skriptu (tzv. front controller), který se volá při každém načtení libovolné stránky. Tento skript spouští jádro aplikace, které pak extrahuje specifické informace dotazu

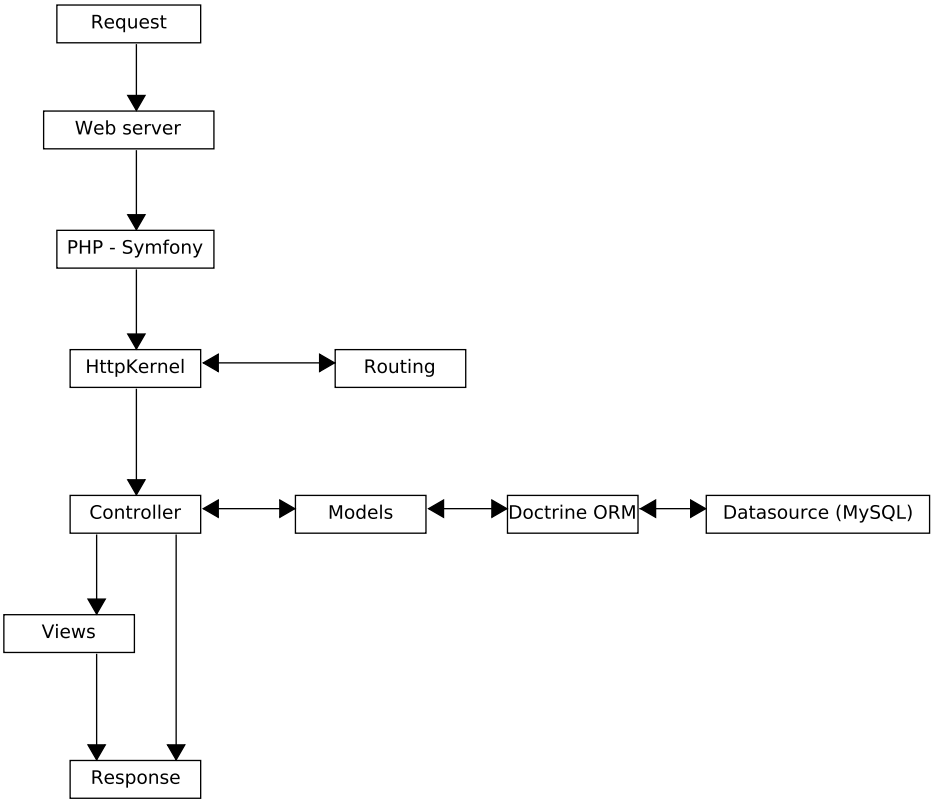
Atribut	Volitelnost	Popis
part_status part_prep_status part_information_meeting	Povinné	Atributy indikující je-li hromadné ubytování schopno spolupracovat v dané aktivitě.
program_morning program_night_prayer program_night_feast program_workshop	Povinné	Atributy indikují zda-li se bude vybraný program konat v hromadném ubytování.
service_guard service_hot_water service_cafeteria	Povinné	Atributy uchovávají informaci o dostupnosti dané služby
class_rooms_number	Povinné	Počet místností.
gyms_number	Povinné	Počet tělocvičen.
sleeping_places_number	Povinné	Počet míst na spaní.
toilets_number	Povinné	Počet toalet.
sinks_number	Povinné	Počet umyvadel.
showers_number	Povinné	Počet sprch.
boys_number	Povinné	Počet míst, které jsou k dispozici pro muže.
girls_number	Povinné	Počet míst, které jsou k dispozici pro ženy.
total_places_number	Povinné	Celkový počet míst.

Tabulka 39: Atributy tabulky `group_accommodation`.

Atribut	Volitelnost	Popis
code	Nepovinné	Kód skupiny. Tento atribut je použit v případě, kdy nahráváme data skupin, migracemi.
name	Povinné	Jméno skupiny.
comment	Nepovinné	Komentář ke skupině.

Tabulka 310: Atributy tabulky `group`.

(jako např. identifikátor načítané stránky) a dle jejich obsahu dále řídí volání příslušných skriptů. Ty poté zpracovávají přijatou informaci a generují s pomocí vrstvy modelu odpověď. Na obrázku 36 je zobrazen workflow aplikace Symfony. [11]



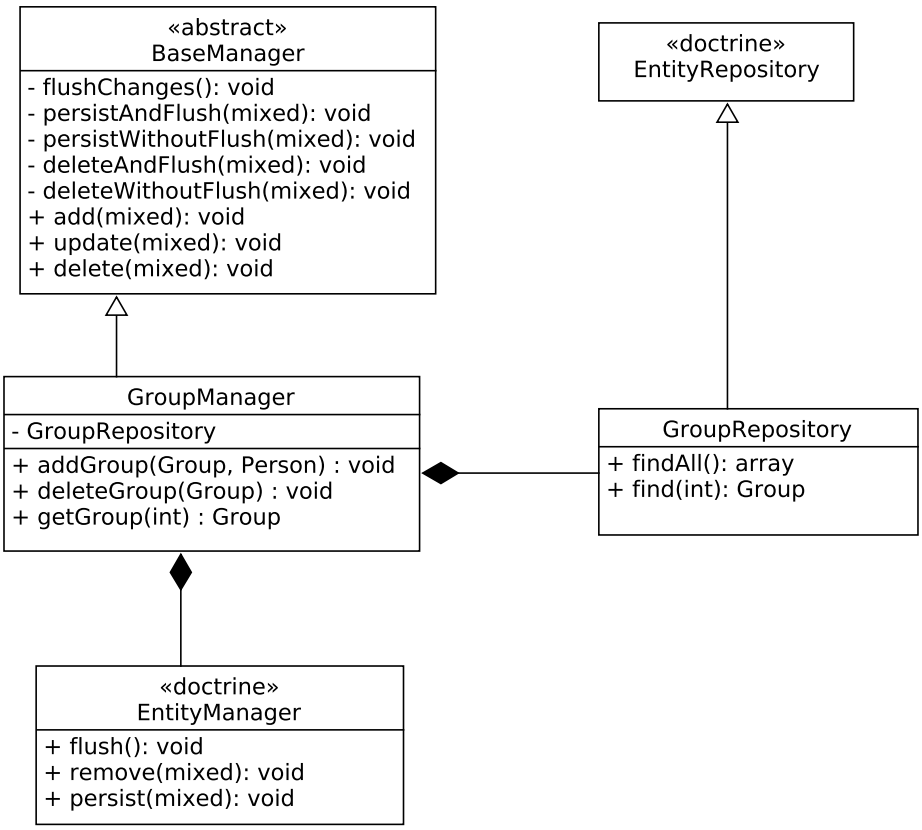
Obrázek 36: Workflow aplikace Symfony

3.3.2 Vrstva Model

Třídy této vrsty představují jednotlivá rozhraní pro práci s databází. Pro každou databázovou tabulku pak existuje třída, která zná její strukturu a je schopna její data extrahovat za použití knihovny Doctrine a definicí tabulek v konfiguračních souborech.

Na obrázku 37 je vidět diagram tříd tohoto modelu. Jako příklad jsem uvedl `GroupManager` a `GroupRepository` pro práci s objekty třídy `group`.

Klíčovou třídou je abstraktní třída `BaseManager`, která se nalézá v komponentě základu. Třída, která dědí od této třídy je zodpovědná za řízení přístupu k databázi a manipulaci s objekty modelu. Jednotlivé požadavky deleguje do objektů třídy `EntityManager` a podtřídy `EntityRepository`, které jsou součástí knihovny Doctrine.



Obrázek 37: Diagram tříd modelu

3.3.3 Vrstva View

Části této vrstvy jsou reprezentovány formou TWIG šablon. Tyto šablony jsou překládány do nativního kódu a uloženy v cache. Jednou z výhod je možnost kombinace a dědění šablon a možnost definice vlastních filtrů a funkcí.

Základní šablonou je `baseHtml`, která definuje kostru html stránky. Od této šablony dědí dále šablona `base`, která udává strukturu v tagu `body` a dále importuje šablony obsahující cesty k CSS a JS souborům.

Další výhodou šablonovacího systému TWIG je možnost přepsání šablony některého z modulů. Příkladem použití je výpis všech modifikací v systému z komponenty logování změn. Šablony tohoto modulu mají podobnou strukturu proto je snadné definovat, nadřazenou šablonu v konkrétním projektu.

3.3.4 Vrstva Controller

Tato vrstva je řízena třídou `HttpKernel`, která pomocí komponenty Routing vyvolá příslušný Controller. To je zajištěno anotacemi v metodách tříd Controllerů. Těmito anotacemi lze definovat pravidla pro routing, povolené http metody a překlad dat dotazu do objektů.

Tato vrstva je zodpovědná pouze za příjetí dotazu, jeho zpracování (delegace do vrstvy modelu) a vytvoření odpovědi.

Kapitola 4

Realizace

4.1 Symfony bundle

Bundle je název pro modul v kontextu Symfony, který slouží k rozšíření aplikace o další funkce. Pomocí Symfony je možné s těmito moduly velice snadno manipulovat. K vytvoření bundle je třeba dodržet předepsanou strukturu. Dále je nutné jej aktivovat v konfiguračním souboru.

4.2 Loggable Bundle

Hlavním úkolem tohoto modulu je logování změn v systému tak, aby bylo možné jednoduše definovat, které entity je nutné logovat bez jakékoliv vazby na konkrétní systém. Toto je zajištěno konfiguračním souborem, který obsahuje sekci allowed_classes. Ta se skládá z pole tříd, jejichž objekty chceme logovat.

Logování všech změn je zajištěno ve třídě `LoggableListener`, která naslouchá na události `onFlush`, `prePersist` a `postPersist`, komponenty doctrine. Ta ukládá veškeré změny od poslední práce s databází v objektu třídy `UnitOfWork`. V momentě, kdy je spuštěna událost `onFlush`, modul zjistí veškeré změny provedené v systému a vytvoří objekty, které reprezentují danou změnu. Po této akci informuje knihovnu Doctrine, že došlo k dalším změnám a tak zajistí opětovnou kalkulaci.

Dále jsem naimplementoval několik veřejných tříd, pro práci s daty historie, popsaných níže.

4.2.1 LogEntryManager a RevisionManager

Tyto třídy jsou veřejným rozhraním pro práci s entitami historie. K dispozici jsou tyto metody:

- `getRealObject(LogEntryInterface): ?LoggableInterface` je metoda, která získá objekt, jehož změna je předána parametrem.
- `approve(LogEntryInterface) : void` je metoda pro akceptování změny konkrétního objektu.
- `refuse(LogEntryInterface) : void` je metoda pro odmítnutí změny konkrétního objektu.
- `hasEntityNotApprovedChanges(LoggableInterface) : bool` je metoda, která vrací `true`, má-li předaná entita změny, které je stále nutné akceptovat.
- `findSubmissionCountFor(LogEntryInterface) : int` nalezne počet změn, které je nutné akceptovat pro entitu předanou parametrem.
- `findLastApprovedValues(LogEntryInterface) : array` vrací pole hodnot atributů konkrétní entity, které jsou akceptovány.

4.2.2 LoggableInterface

Tento interface musí implementovat třída, která je obrazem databázové tabulky, aby bylo možné logovat modifikace v této tabulce. Obsahuje 3 metody popsané níže, které je nutné implementovat.

- `getId(): int` je metoda, která vrací id entity.
- `setCreateLog(bool) : void` nastavuje příznak pro další uchování historických dat. Je-li příznak `false`, nebude příští změna brána v potaz.
- `isCreateLog() : bool` vrací nastavení příznaku.

4.2.3 GUI rozhraní

Přidal jsem velice jednoduché rozhraní pro správu změn v systému. Šablony jsem vytvořil minimální, aby je bylo možné rozšířit v konkrétní aplikaci. K použití je nutné importovat routing definice a přepsat již zmíněné šablony.

4.3 Base Bundle

Prvním důvodem pro vytvoření tohoto modulu bylo vytvoření správy uživatelů. Pro tento požadavek jsem použil modul FOSUserBundle, který přidává podporu pro práci s uživateli, uživatelskými skupinami a dále dokáže přidat do aplikace funkce registrace a přihlášení uživatele. Obsahuje také definice databázových tabulek a abstraktní třídy, které jsou obrazem těchto tabulek. Tyto definice jsem rozšířil o další atributy a implementoval třídy, které dědí od již zmíněných abstraktních tříd.

EasyAdminBundle je modul, který do systému přidává administrátorské uživatelské rozhraní. Tento modul jsem použil pro správu rolí, uživatelů, skupin a oprávnění. Práce s ním je velice jednoduchá a tak stačilo definovat jména tříd a CRUD akce, které jsou vyžadovány. Konfigurační soubor je snadné rozšířit v konkrétním projektu.

Další funkcí této komponenty je integrace modulů LoggableBundle, PermissionBundle a SqlReportBundle, tak aby každý modul byl schopen použít funkce ze zbývajících. Příkladem je komponenta PermissionBundle. Je-li aktivována, chceme použít oprávnění i v komponentě LoggableBundle. Vytvořil jsem proto několik konfiguračních souborů rozdělených podle modulů. Při kompilaci DI kontejneru pak komponenta zjišťuje zda-li požadovaný modul je aktivní a podle toho nahrává tyto konfigurační soubory. Při použití jen jedné komponenty nedochází k problémům ani zbytečnému nahrávání nepotřebných funkcí.

4.4 Permission Bundle

Komponenta rozšiřuje autorizační pravidla frameworku Symfony o třídu oprávnění. Tyto oprávnění je pak možné definovat v konfiguračním souboru aplikace nebo konfiguraci konkrétního modulu. Pomocí třídy `LoadPermissionDataCommand` lze tyto oprávnění nahrát do databáze.

`PermissionManager` je implementací interface `PermissionManagerInterface`, která přidává veřejné rozhraní pro práci s databází pomocí knihovny Docrine. Při použití jakékoliv jiné knihovny nebo úložiště je nutné implementovat již zmíněné rozhraní.

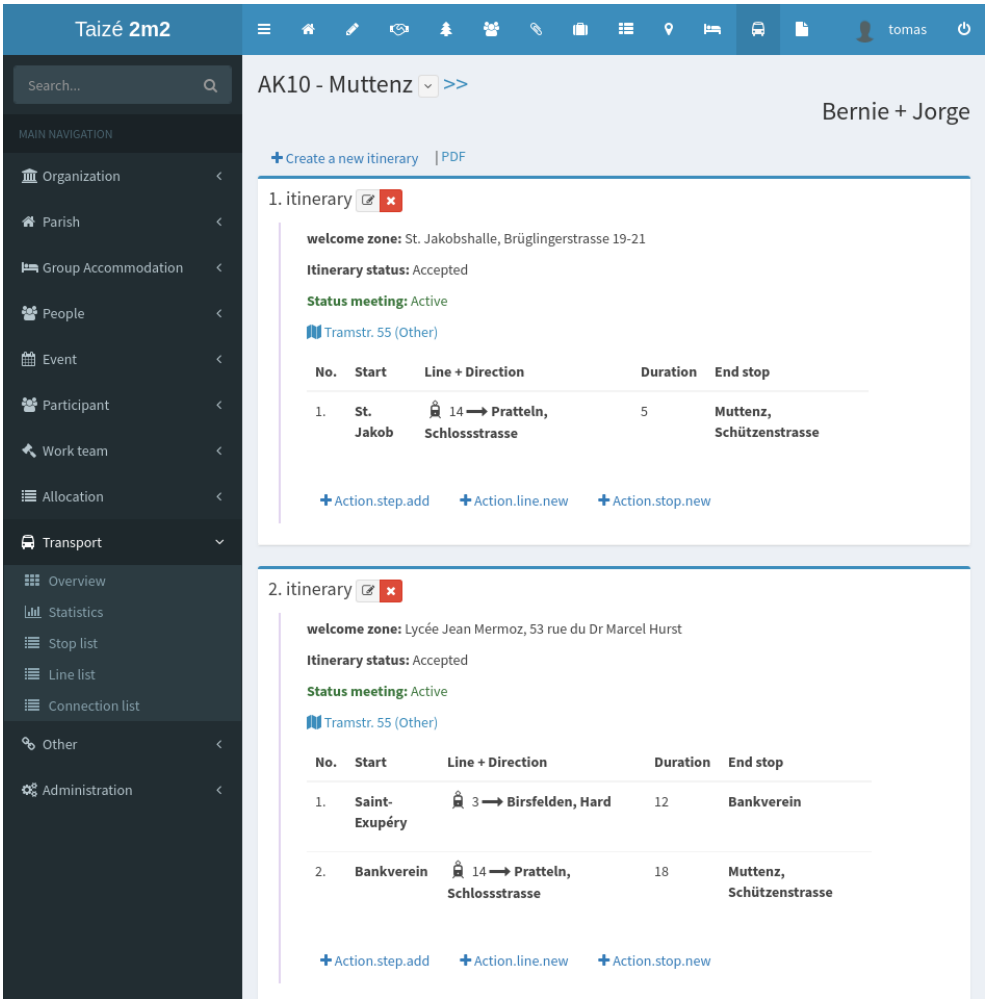
`CachedUserManager` je třídou, která přidává veřejné rozhraní pro práci s oprávněním uživatele. Oprávnění ukládá do cache a tím redukuje počet dotazů nad databází.

4.5 Uživatelské grafické rozhraní

Od roku 2013 se mnoho nástrojů pro práci s webovým GUI změnilo vzhledem k rozšíření jazyka JavaScript. Musel jsem tedy provést několik změn.

První změnou bylo odstranění zastaralé PHP knihovny Assetic a nahrazení knihovnou webpack. Pro snadnější práci používám nadstavbu Encore vyvíjenou týmem Symfony.

Druhou výraznou změnou bylo použití nadstavby AdminLTE frameworku bootstrap. Prostředí je tak přehlednější a více intuitivní. Rozhraní jsem rozdělil do několika částí (viz. obrázek 41). Na levé straně se nalézá menu aplikace s globálním políčkem pro vyhledávání důležitějších informací. V horní části se nalézá lokální menu, které se mění dle generované stránky. Poslední sekcí je prostor pro zobrazení konkrétní stránky.



Obrázek 41: Grafické znázornění rozhraní aplikace

4.6 Aktualizace aplikace

Aplikaci jsem aktualizoval na verzi Symfony 4.1 a PHP 7.2. Tato aktualizace byla poměrně pracná, neboť ve verzi 4.0 došlo k velkým změnám struktury kódu a tak jsem byl nucen měnit namespace všech tříd v aplikaci.

4.7 Itineráře k setkání

Generování itinerářů je jedna z nejdůležitějších funkcí systému, neboť itineráře obsahují veškeré informace, které účastník potřebuje. Každou stránku jsem navrhl v TWIG šabloně, která se pak převádí do souboru formátu PDF pomocí aplikace wkhtmltopdf. Dále jsem vytvořil příkazy ke generování a manipulaci

těchto stránek. Tyto příkazy je možné také spustit z webového prostředí.

4.8 Testování

V předchozí verzi aplikace chyběl jakýkoliv test. Bylo to také z důvodu použití knihoven, které velmi zasahovaly do architektury systému. V této bakalářské práci jsem se snažil situaci vylepšit.

Tříd v systému je velké množství, proto jsem nedbal na 100% pokrytí jednotkovými testy. Testoval jsem převážně entity. Dalším krokem by mělo být přidání více testů a tím se vyvarovat narušení již fungujících částí systému při případném rozšíření.

Mnohem více jsem se věnoval smoke a funkcionálním testům. Tyto testy prochází aplikaci a testují zobrazení dané stránky, interakci s formuláři i návratové kódy. Dokáží tedy rychle odhalit případnou chybu.

Závěr

Na aplikaci 2m3 pracuji již od roku 2014, kdy jsem poprvé obhájil svou bakalářskou práci. Za tuto dobu jsem se velmi dobře seznámil s problematikou přípravy evropského setkání. Zpočátku nebylo jednoduché identifikovat veškeré požadavky a případy užití, neboť jsem s přípravným týmem neměl žádný kontakt. To se však změnilo po evropském setkání v Praze, kdy jsem měl možnost být součástí tohoto týmu.

V první části jsem se věnoval analýze stávajícího řešení. Z této analýzy je patrné, že aplikace trpěla několika nedostatky. Ty jsem se snažil odstranit ještě před návrhem nových požadavků. Dále jsem provedl sběr požadavků, které se od roku 2014 velmi změnily a mění se každým rokem. Došel jsem k závěru, že je třeba navrhnout systém tak, aby dokázal pružně reagovat na tyto změny. V poslední části jsem vytvořil návrh a provedl jeho implementaci.

Vznikl plně funkční informační systém, který je nasazen ve starší verzi již od roku 2015 na adrese `https://2m2.taize.fr`, jako kompletní náhrada dřívějších manuálních scénářů. Veškeré požadavky byly tedy splněny a řádně otestovány uživateli předchozích setkání.

I přesto, že se podařilo rozšířit systém o chybějící funkce, není vývoj aplikace u konce. Ve většině případů bude docházet k údržbě a rozšíření kvůli novým požadavkům. Je tedy nutné doplnit testovací sadu o další jednotkové a funkcionální testy, tak aby nedocházelo k regresním chybám.

Prací na tomto projektu jsem získal řadu cenných zkušeností. Zkusil jsem si komunikaci se zadavatelem v cizím jazyce. Rozšířil znalosti ve vývoji webových aplikací a jejich nasazení v reálném prostředí. Ověřil jsem si důležitost analýzy. A blíže jsem se seznámil s nástroji jazyka PHP, verzování kódu a především s frameworkem Symfony.

Literatura

[1] et Presses de Taizé, A.: European Meeting in Basel. [ONLINE]. Dostupné z: https://www.taize.fr/en_rubrique3161.html

[2] Ryan, F.: Language Framework Popularity: A Look at PHP. [cit. 2016-11-01]. Dostupné z: http://redmonk.com/fryan/2016/11/01/language-framework-popularity-a-look-at-php/

[3] Symfony™: Projects using Symfony. [ONLINE]. Dostupné z: https://symfony.com/projects

[4] Symfony™: Symfony, 9 years of history, rewards its top 150 contributors. [cit. 2014-10-21]. Dostupné z: https://symfony.com/blog/symfony-9-years-of-history-rewards-its-top-150-contributors

[5] Symfony™: Hello Symfony 4! [cit. 2017-11-30]. Dostupné z: https://symfony.com/blog/hello-symfony-4

[6] Peipman, G.: Why to avoid fat controllers. [cit. 2015-05-31]. Dostupné z: http://gunnarpeipman.com/2015/05/why-to-avoid-fat-controllers/

[7] Symfony™: Introducing Webpack Encore for Asset Management. [cit. 2017-06-13]. Dostupné z: https://symfony.com/blog/introducing-webpack-encore-for-asset-management

[8] project, D.: *Doctrine Query Language*. [ONLINE]. Dostupné z: https://www.doctrine-project.org/projects/doctrine-orm/en/2.6/reference/dql-doctrine-query-language.html

[9] Fowler, M.: Catalog of Patterns of Enterprise Application Architecture. [ONLINE]. Dostupné z: https://martinfowler.com/eaaCatalog/lazyLoad.html

[10] Microsoft: Model-View-Controller. [cit. 2014-03-17]. Dostupné z: https://msdn.microsoft.com/en-us/library/ff649643.aspx

[11] Point, T.: Symfony - Architecture. [ONLINE]. Dostupné z: https://www.tutorialspoint.com/symfony/symfony_architecture.htm

Příloha **A**

Seznam použitých zkratek

CRUD Create, read, update and delete.

CSS Cascading Style Sheets.

DI Dependency injection.

DQL Doctrine query language.

JS JavaScript.

MVC Model-View-Controller.

OQL Object query language.

ORM Object relational mapper.

PDF Portable Document Format.

PHP PHP: Hypertext Preprocessor.

PNG Portable Network Graphics.

SQL Structured Query Language.

Příloha B

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `exe` ..... adresář se spustitelnou formou implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF