VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

REZERVAČNÍ SYSTÉM PRO AKCE NEZISKOVÝCH ORGANIZACÍ

RESERVATION SYSTEM FOR EVENTS OF NON-PROFIT ORGANIZATIONS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **PATRIK ČERNÝ**
AUTHOR

VEDOUCÍ PRÁCE **Ing. JAROSLAV DYTRYCH, Ph.D.**
SUPERVISOR

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2021/2022

Zadání bakalářské práce

Student: **Černý Patrik**
Program: Informační technologie
Název: **Rezervační systém pro akce neziskových organizací**
Reservation System for Events of Non-profit Organizations
Kategorie: Informační systémy

Zadání:

1. Seznamte se s jazyky a prostředky pro tvorbu webových informačních systémů.
2. Proveďte analýzu požadavků uživatelů na rezervační systém na akce spolku ČFN pořádané za pomoci dobrovolníků a prostudujte dostupné řešení pro evidenci členů a členských příspěvků spolku.
3. Navrhněte nový informační systém pro pořádání akcí členy spolku, přihlašování dobrovolníků na aktivity v jejich programu a rezervaci míst na akcích a platby záloh účastníky. Nový systém současně poskytne evidenci členů spolku a plateb členských příspěvků.
4. Implementujte navržené řešení.
5. Zhodnoťte dosažené výsledky a vytvořte stručný plakát prezentující výsledky práce.

Literatura:

- Dle doporučení vedoucího

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 - 3.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Dytrych Jaroslav, Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 29. července 2022
Datum schválení: 1. listopadu 2021

Abstrakt

Tato práce se zabývá návrhem rezervačního systému pro spolek Českomoravská federace naturistů (ČMFN). Jejím cílem je navrhnout, implementovat a zdokumentovat nově vytvořený systém, který by měl umožňovat evidenci členských příspěvků a pořádání akcí. Dále tato práce popisuje, jak by se v novém systému měli uživatelé přihlašovat na akce či aktivity a platit za ně zálohy.

Abstract

This thesis deals with designing the reservation system for „Českomoravská federace naturistů“ association. Its purpose is to design, implement and document a newly created system which should be able to record membership fees and organizing of events. This thesis also describes how members should sign up for events or activities and pay deposits for them.

Klíčová slova

rezervační systém, informační systém, akce, aktivity, platby, bankovní API, Nette Framework, PHP, MySQL, časová pásma, rodinní příslušníci

Keywords

reservation system, information system, events, activities, payments, bank API, Nette Framework, PHP, MySQL, time zones, family members

Citace

ČERNÝ, Patrik. *Rezervační systém pro akce neziskových organizací*. Brno, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jaroslav Dytrych, Ph.D.

Rezervační systém pro akce neziskových organizací

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jaroslava Dytrycha, Ph.D. Další informace mi poskytli členové výboru Českomoravské federace naturistů. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Patrik Černý
29. července 2022

Poděkování

Chtěl bych poděkovat svému vedoucímu práce panu Ing. Jaroslavovi Dytrychovi, Ph.D. za ochotu se mnou konzultovat různé nejasnosti i v časech, kdy jsme zrovna neměli naplánovanou konzultaci. Rovněž bych rád poděkoval za odborné vedení práce a velmi přínosné připomínky a doporučení k vypracování. Dále bych chtěl poděkovat spolku Českomoravská federace naturistů za dodání potřebných materiálů a za možnost navrhnout a vytvořit jejich nový informační systém.

Obsah

Kapitola 1

Úvod

V dnešní době řada spolků potřebuje evidenci informací o členských příspěvcích a svých pořádaných akcích. Tento systém by měl řešit zmiňovanou evidenci a navíc umožňovat uživatelům automatické strhávání členského příspěvku. Pomocí kreditního systému, na který lze dobíjet libovolnou částku, se pak případným uživatelům sníží potřeba využívat internetové bankovnictví a budou moci platit za akce přímo ze systému jedním kliknutím.

U neziskových organizací, které pořádají různé akce a nemají k dispozici vybraný systém, může dojít k problémům, kdy není předem znám počet zájemců o danou akci. To pak může znamenat nižší účast na akci, což donutí organizátora akce, aby musel doplácet akci ze svých zdrojů. Může nastat i opačný případ, kdy má akce úspěch, ale není zajištěna dostatečná kapacita.

Cílem práce je tedy navrhnout a implementovat systém, který by dokázal evidovat členské příspěvky za pomoci kreditního systému. Navíc by měl umět vytvářet akce, kde organizátoři uvedou všechny potřebné náležitosti. Uživatelé systému by pak měli mít možnost se na akci přihlásit a stvrdit svůj zájem zaplacením zálohy použitím zmíněného kreditního systému. Rovněž je časté, že mnoho zájemců o danou akci chce prokázat vlastní iniciativu a navrhují různé doplňkové aktivity. V současné době však není možné členům zajistit takové možnosti bez systému, který by se o zakládání těchto akcí staral.

Příkladem spolku, který tento systém potřebuje, je Českomoravská federace naturistů, spolek (ČMFN).

Tato práce se tedy zabývá navržením vhodného způsobu, který by zmíněné požadavky mohl zrealizovat. Nejprve je analyzován současný stav, což je popsáno v kapitole 2. Následně byla v kapitole 3 provedena analýza požadavků na vytvářený systém, kdy bylo potřeba zjistit, co vše by měl systém umět a jaká by z toho mohla plynout omezení či problémy v další fázi, kterou je samotný návrh. Návrh, uvedený v kapitole 5, se již zabývá detailnějším popisem vybraných částí a snaží se je uvést do přesnější a srozumitelnější podoby. Dokumentace postupu při navrhování tohoto rozsáhlého systémů je pak následována kapitolou 6 zabývající se implementací, kde jsou zmíněny vybrané problémy a jejich konkrétní řešení. Po této kapitole následuje popis testování v průběhu implementace, jehož výsledky jsou uvedeny v kapitole 7. Dosažené výsledky a možná rozšíření jsou zmíněna na úplném závěru v kapitole 8.

Kapitola 2

Popis spolku a dosavadního stavu systému

Neziskové organizace často pořádají akce pro své členy. Tyto akce zajišťují buď samotní členové dané neziskové organizace, nebo členové jejího výboru. K uskutečnění akce je však potřeba jistý finanční základ, jelikož rezervace lokality a případného ubytování či stravování se musí téměř vždy zaplatit předem.

Mezi takové neziskové organizace patří například spolky, přičemž spolek, kterým se tato kapitola a zbytek práce bude zabývat, se označuje názvem `Českomoravská federace naturistů, spolek` (zkratka `ČMFN`) a je členskou federací mezinárodní organizace International Naturist Federation (INF-FNI).

2.1 Činnosti spolku

Výše zmíněný spolek má za úkol zejména:

- evidovat nové a existující členy,
- vybírat členské příspěvky, vydávat členské průkazy a známky INF-FNI,
- organizovat akce a k nim přidružené aktivity,
- vybírat zálohy od zájemců o danou akci.

Těmito a dalšími činnostmi se bude tato podkapitola zabývat v příslušných sekcích.

Členové spolku

Aby se osoba mohla stát členem ČMFN, musí se zaregistrovat v informačním systému a zaplatit členský příspěvek. Pokud uživatel nezaplatí členský příspěvek, nebere se jako člen spolku, ale jako pouhý návštěvník webu, který má zřízený účet. Je tomu tak proto, že se musí ověřit osobní údaje daného člena, zejména věk, neboť členem může být pouze zletilá osoba. Dále jsou tyto údaje nutné pro případné slevy na akce pro studenty či seniory a osoby se zdravotním postižením.

Osoba, která se registruje na webu spolku, navíc musí souhlasit se stanovami spolku[1]. Jakmile se uživatel stane členem, může organizovat akce a aktivity nebo se vytvořených

[1] Stanovy spolku jsou dostupné na tomto odkazu: https://or.justice.cz/ias/ui/vypis-sl-detail?dokument=39738846&subjektId=103101&spis=992591.

akcí účastnit. Mezi další důležité členy patří volení členové výboru, kteří rozhodují o výši členských příspěvků a schvalují navržené akce. Členové výboru se dělí na základě funkce:

- Člen výboru – podílí se na rozhodování o členských příspěvcích, schvaluje a organizuje akce či aktivity, má přístup k seznamu všech členů,
- Předseda – má stejné pravomoce jako člen výboru, ale navíc je zodpovědný za chod spolku a má za spolek podpisové právo,
- Místopředseda – zastává funkci předsedy, pokud předseda nemůže vykonávat svou funkci,
- Pokladník – eviduje veškeré platby, zajišťuje vydávání členských průkazů pro nové členy a zasílání známek existujícím členům na nový kalendářní rok.

Každý uživatel systému si může přidávat rodinné příslušníky, kterým může také zaplatit členství. V takovém případě je částka za rodinného příslušníka nižší, než kdyby ji platila daná osoba sama. Doposud všichni přidaní rodinní příslušníci v systému nemohli mít vlastní účet, což bylo velkým omezením pro řadu členů. Kdyby se totiž zaregistrovali pod svým účtem, museli by zaplatit plnou částku, aby se v systému jevili jako řádní členové. Také se sami nemohli přihlašovat na akce či aktivity dané akce. Veškeré přihlašování řešil člen, který měl rodinné příslušníky přidané v systému.

Členské příspěvky

V současném stavu se členský příspěvek platí zasláním příslušné částky na zřízený transparentní účet spolku. Nastavení statusu řádného člena se provádí ručně členem výboru, který musí sledovat platby v internetovém bankovnictví. Členské příspěvky se platí jednou ročně nejpozději na konci ledna daného kalendářního roku, pro který má být členství zaplaceno. Pokud má člen rodinné příslušníky, může jejich členství zaplatit společně se svým, přičemž se cena členského příspěvku za rodinného příslušníka sníží z plné částky 500 Kč na pouhých 100 Kč dle aktuálních stanov spolku. Za děti do 15 let se členské příspěvky neplatí.

Životní cyklus akce

Každá akce začíná návrhem, kde se bude konat a jaká bude její náplň. Po schválení akce členem výboru se mohou na akci přihlašovat zájemci až do určeného termínu. Po zjištění přibližného počtu zájemců pořadatel akce domluví lokalitu, zaplatí zálohu za rezervaci daného místa a případně rovněž zaplatí zálohu za ubytování a za jiné aktivity pro zájemce o akci. Následně akce proběhne, přičemž každý účastník zaplatí vstupné v den konání na místě události. Akce i po skončení může sloužit jako příklad podařené události na webu spolku a upoutat možné nové zájemce pro příští akci.

Akci může pořádat nejen člen výboru, ale i řádný člen. To dává volnost členům při realizaci vlastních nápadů a jejich členství není omezeno pouze na slevy. Avšak z toho plynou rizika popsaná níže.

Aktivity a pořadatelé

Akce má nějaký program, což jsou různé aktivity na místě akce (např. hra šipky). Ovšem pro uskutečnění některých aktivit je nutné dodat další prostředky (např. pro hru šipky je

nutné dodat terč a samotné šipky na házení). Takové aktivity jsou většinou dobrovolné a záleží pouze na zájemcích, zda dodají požadované prostředky. V současném systému není možné zjistit seznam potřebných prostředků a program akce je uveden pouze jako seznam odrážek v popisu akce, tudíž se často některé aktivity neuskuteční právě z důvodu nedostatku požadovaných prostředků. Organizátor totiž často nemá rozpočet na to, aby veškeré věci zorganizoval a dodal sám, je tedy nutný zájem ze strany účastníků akce.

Další problém v současném systému nastává v tom, že když se někdo rozhodne dodat prostředky pro uskutečnění aktivity a sdělí svůj zájem pořadateli akce, tak pořadatel nemá naprostou jistotu, že to daná osoba skutečně zajistí. Pořadatel tedy vypíše do popisu akce, že se bude konat daná aktivita, kvůli které se přihlásí zájemci, ale v průběhu konání akce se tato aktivita neuskuteční, protože nebudou dodány prostředky, což vyústí v nespokojenost účastníků a menší pravděpodobnost přihlášení těchto osob na jinou akci v budoucnu.

Akce může mít více pořadatelů a zároveň aktivity může organizovat někdo, kdo neorganizuje samotnou akci. Jak již bylo zmíněno výše, akce i aktivity může organizovat i řádný člen a nejen člen výboru.

Přihlašování členů na akce a aktivity

Doposud zájemci nemuseli platit zálohu předem, ale zaplatili vstupné až na místě akce. To znamenalo nejistotu v počtu zájemců. V minulosti se tedy často stávalo, že se členové sice na akci přihlásili, s jejich účastí se počítalo a uhradily se potřebné náklady pro uskutečnění akce, ale v den konání se tito členové nedostavili a veškeré náklady pak musel doplácet organizátor akce z vlastních zdrojů. Proto by měl nový systém vyžadovat placení zálohy za akci předem, aby se tyto situace nestávaly.

Dalším možným problémem je přihlašování na program akce, mezi který může patřit například ubytování či plná penze (strava). V minulosti se všechny tyto aktivity připravovaly předem pro všechny zúčastněné, ale v den konání ne každý vyžadoval ubytování či stravu a neměl potřebu platit za něco, co sám nevyužil. Současný systém však neumožňuje určit, kdo o danou aktivitu má zájem a kdo ne, a proto se tyto zásadní části akce rezervovaly pro všechny. S novým systémem by však každý zájemce měl volnost a mohl by stvrdit svůj zájem o daný program akce svým přihlášením a zaplacením zálohy. V případě, že by pak nevyužil to, na co se přihlásil, pořadatel programu by alespoň mohl využít, v tomto stavu již nevratnou, zaplacenou zálohu k pokrytí svých nákladů.

Z akcí je samozřejmě možné se odhlásit. Nicméně současný systém nemá žádnou evidenci přihlášených členů a když někdo věděl, že se nemůže akce zúčastnit, musel napsat přímo pořadateli dané akce. Na to mnoho lidí zapomínalo, a tak docházelo k nepříjemným situacím popsaným výše (s přihlášenými členy se počítalo, ale nedostavili se a pořadatel akce či aktivity musel za tyto členy doplácet).

V nově navrženém systému by tedy členové měli mít možnost se do zadaného termínu odhlásit, přičemž se jim vrátí zaplacená záloha. Pokud by se však do tohoto termínu neodhlásili, s přihlášenými zájemci se bude počítat a záloha je v takovém případě nevratná.

2.2 Současná databáze spolku

Současná databáze, kterou využívá informační systém spolku ČMFN, je velmi špatně navržená ze dvou hlavních důvodů. Informace o uživatelích ukládá v některých entitách v PHP serializovaném formátu[2], čímž dochází k mnoha problémům (redundance dat, neaktuálnost dat, větší nároky na zpracování, ukládání a dohledávání). Dalším velkým problémem je to, že jsou hesla ukládána pomocí hashovacího algoritmu **MD5**. Tento hashovací algoritmus je nyní nebezpečný na použití ze tří hlavních důvodů [13]:

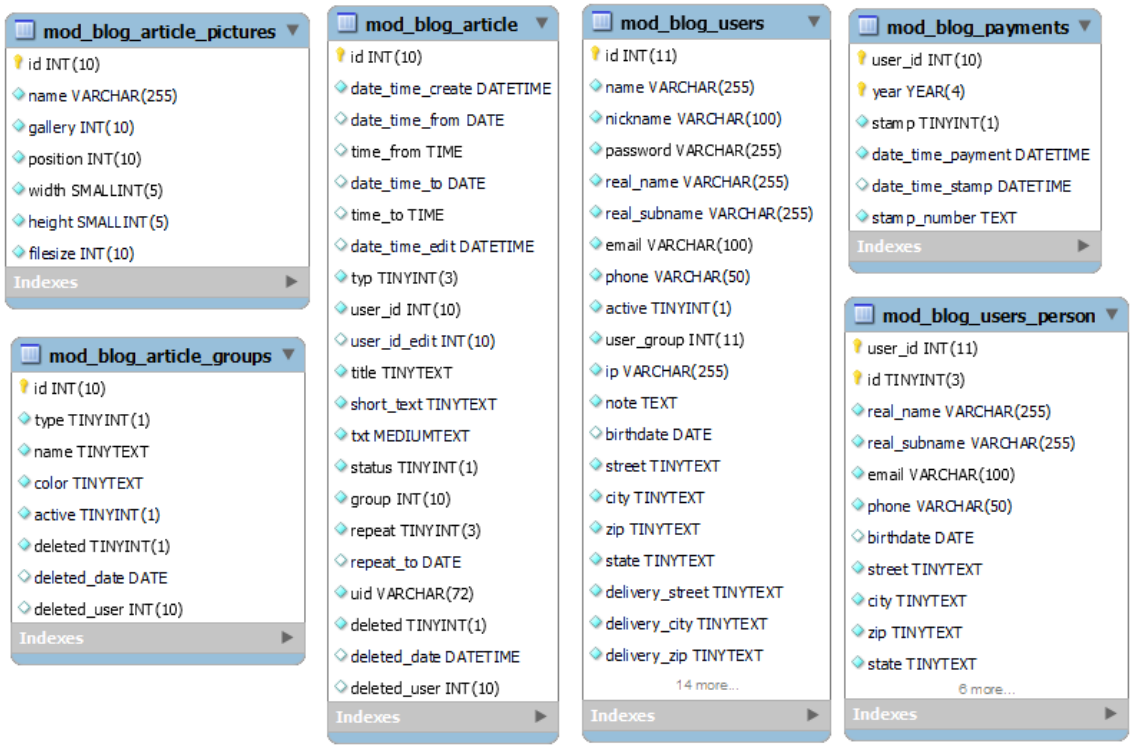
- Útoky typu *brute force*[3] jsou v současné době moderních procesorů a výkonných grafických karet velmi rychlé, jelikož je algoritmus MD5 rychlý na použití (vygenerování hashe z textu je rychlé a tudíž i zkoušení různých hesel lze provést daleko rychleji než v minulosti).
- Existuje slovník MD5, kde pro každý hash je určeno původní heslo. V dnešní době je takových slovníků mnoho a zabírají vysoké spektrum možných hesel. Například heslo „abc123“ lze s velkou pravděpodobností v takovém slovníku najít během pár sekund.
- Algoritmus MD5 má nízkou odolnost proti kolizím. To znamená, že můžou vzniknout dva stejné hashe pro dvě různá slova (hesla). Jedná se o velmi špatnou vlastnost hashovacích algoritmů, poněvadž útočník pak může hádat spoustu odvozených slov.

Za další nedostatek lze považovat použití engine[4] **MyISAM** místo **InnoDB** při použití databáze **MySQL**. Tento engine nepodporuje cizí klíče, což pak znamená vyšší režii při odstraňování záznamů z entit, které mají závislosti na jiné entity. S cizími klíči by bylo možné použít kaskádové smazání, což smaže všechny závislé záznamy, a nebylo by nutné psát doplňující SQL dotazy pro vymazání závislých záznamů z jiných entit. Takových entit může být mnoho a pak se z jednoduchého dotazu pro smazání několika záznamů stává komplikovaný dotaz náročnější na provedení. Současně úložiště **MyISAM** nepodporuje databázové transakce, což je velmi nebezpečné např. při ukládání platebních výpisů.

[2] https://www.php.net/manual/en/function.serialize.php

[3] Jedná se o útok na hesla uživatelů, kdy se zkouší všechny kombinace písmen, čísel a speciálních znaků.

[4] Engine je databázová komponenta zodpovědná za to, jak jsou data ukládána a jak je k datům přistupováno. Detailnější popis podporovaných úložišť databáze MySQL je dostupný na stránce https://www.w3resource.com/mysql/mysql-storage-engines.php.



Obrázek 2.1: Entity z ER diagramu původní databáze

Na obrázku 2.1 lze vidět příklad některých entit ze starého systému. Například entita s názvem `mod_blog_payments` ukládala známky INF-FNI na nový kalendářní rok pro členy a jejich rodinné příslušníky v serializované podobě v atributu `stamp_number`, který má jako datový typ nastaven neomezený textový řetězec. Datový typ je rovněž nastaven špatně, jelikož z dokumentace funkce `serialize()` jazyka PHP je možné se dočíst, že vhodný datový typ by byl v takovém případě `BLOB` a nikoliv `TEXT`. Místo použití další tabulky pro evidenci známek, ve které by byl odkaz na uživatele, se ukládaly potřebné osobní údaje a záznamy o rodinných příslušnících v textové podobě. V momentě, kdy by si uživatel upravil své osobní údaje na profilové stránce, přestaly by tyto údaje být aktuální a člen výboru, jenž by z nich četl, by poté mohl udělat velké chyby při zajišťování známek.

Další dvě entity se týkají uživatelů a rodinných příslušníků. Entita `mod_blog_users` eviduje zaregistrované uživatele v systému. Entita `mod_blog_users_person` pak eviduje rodinné příslušníky zaregistrovaného uživatele. Takto navržená databáze neumožňuje, aby byl rodinný příslušník plnohodnotným uživatelem, který má možnost se přihlašovat a používat funkce systému, a zároveň aby byl dohledatelný jako rodinný příslušník (např. při placení členských příspěvků). Atribut `user_id` z tabulky `mod_blog_users_person` zaznamenává ID uživatele, který rodinného příslušníka přidal (reference na atribut `id` z tabulky `mod_blog_users`), zatímco atribut `id` je identifikátor rodinného příslušníka. Oba tyto atributy tvoří složený primární klíč. Je tedy zřejmé, že by bylo problematické v současném návrhu zařídit evidenci rodinných příslušníků tak, aby zároveň mohli být samostatnými uživateli bez zbytečného ukládání duplicitních dat.

Kapitola 3

Analýza požadavků na nový rezervační systém

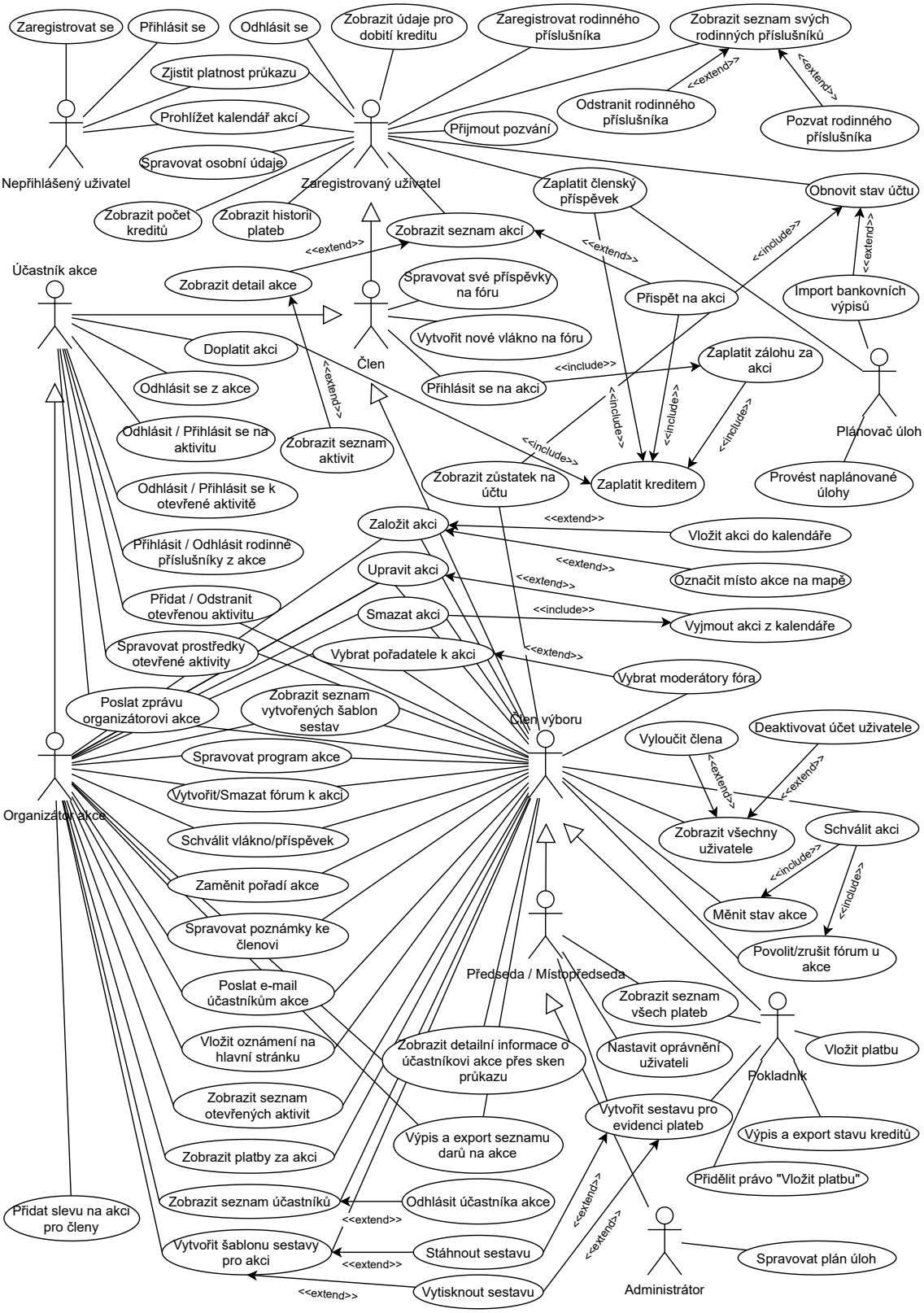
Tato kapitola se zabývá zjištěním požadavků, které by mělo splňovat nově navržené řešení, a detailnějším popisem některých stavů v systému.

3.1 Diagram případů užití

Tento typ diagramu pochází z rodiny behaviorálních diagramů jazyka UML (zkratka pro Unified Modeling Language). Slouží k modelování funkčních požadavků systému. Funkční požadavky znamenají, co za možnosti (funkce) by měl systém umět. Je vhodný zejména pro zobrazení interakcí mezi uživateli a systémem bez nutnosti znalosti vnitřní implementace navrhovaného systému. Diagram případů užití se skládá ze tří hlavních komponent [1]:

- Aktéři – znázorňují uživatele, kteří pracují se systémem, nebo jiný vnější systém, jenž požadovaných funkcí využívá (např. čas),
- Případy užití – jednotlivé akce systému, které aktéři mohou provádět,
- Vztahy – propojení mezi aktérem a případem užití vyjadřující, že daný aktér může provést danou akci. Kromě standardního vztahu existují navíc dva speciální vztahy [19]:
  - Rozšiřující vztah (`«extend»`): Případ užití rozšiřuje hlavní případ užití, ke kterému aktér přistupuje, o novou funkcionalitu, která se však nemusí vždy vykonat.
  - Zahrnující vztah (`«include»`): Při provedení hlavního případu užití se bezpodmínečně provedou také případy užití, které jsou k hlavnímu případu užití tímto vztahem připojeny.

V rámci sezení s členy výboru ČMFN vznikl diagram případů užití znázorněný níže:



Obrázek 3.1: Diagram případů užití

Následuje popis jednotlivých aktérů a nejdůležitějších případů užití z diagramu na obrázku 3.1.

Seznam aktérů:

- *Plánovač úloh* – Nejedná se o osobu, nýbrž o *CRON* službu[1], která má na starost spouštění naplánovaných úloh (např. import bankovních výpisů).
- *Nepřihlášený uživatel* – Uživatel, který buď nemá vytvořený účet (není registrovaný), nebo není přihlášen.
- *Zaregistrovaný uživatel* – Uživatel je zaregistrován a přihlášen, ale nemá platné členství. Na člena, který má vypršené členství, je rovněž pohlíženo jako na zaregistrovaného uživatele.
- *Člen* – Člen je zaregistrovaný uživatel, který zaplatil členství a má možnost se přihlašovat na akce či je vytvářet. Rovněž má k dispozici slevy na vstupném u některých akcí.
- *Účastník akce* – Tento typ uživatele není v systému implementačně uveden jako samostatná role, naopak se jedná stále o člena. V diagramu je však uveden proto, aby bylo možné rozlišit, kdy se člen účastní akce a jaké z toho pro uživatele vyplývají další dostupné funkce. Účastníkem akce se člen stává po přihlášení na akci.
- *Organizátor akce* – Člen se stává organizátorem akce v momentě, kdy založí minimálně jednu akci. Opět se nejedná o nový typ člena, ale v diagramu je znázorněn ze stejného důvodu, jako bylo zmíněno výše.
- *Člen výboru* – Zde se již jedná o nový typ člena. Tento člen má oprávnění ke správě a schvalování všech akcí. Zároveň si může zobrazit seznam uživatelů a rozhodovat o jejich vyloučení.
- *Pokladník* – Tento člen výboru je zodpovědný za všechny platby v systému, které jsou buď přidány automaticky plánovačem úloh, nebo vloženy ručně pokladníkem (označuje případ užití *„Vložit platbu“*). Rovněž může přidělit právo vložit platbu i jinému členovi výboru, ale stále je jeho odpovědnost, komu dané právo přiřadí a jak s ním daná osoba naloží.
- *Předseda / Místopředseda* – Obě tyto role mají stejná oprávnění a jedná se o nejvyšší roli v systému mimo administrátora. Má na starosti oprávnění uživatelů a seznam plateb.
- *Administrátor* – Administrátor má všechna práva jako předseda či místopředseda, ale navíc může spravovat plán úloh, který slouží k automatickému spouštění skriptů. Tyto automaticky spouštěné skripty se využijí v případě kontroly expirace některých odkazů nebo členství uživatelů.

Případ užití s názvem *„Zaregistrovat rodinného příslušníka“* znamená, že zaregistrovaný uživatel má možnost vytvořit účet rodinnému příslušníkovi bez přihlašovacích údajů se základními osobními údaji. Takový uživatel se sice nemůže přihlásit do systému, ale uživatel

[1] Jedná se o automatické spouštění úloh v operačních systémech UNIX/Linux na základě dodaného souboru s intervaly a cestami ke spustitelným souborům, které se mají v určeném čase spouštět.

mu může poslat pozvánku k vyplnění přihlašovacích údajů na zadanou e-mailovou adresu, což reprezentuje případ užití *„Pozvat rodinného příslušníka“*. Tento případ užití mimo jiné umožňuje zaslat pozvánku již existujícímu uživateli v informačním systému. V takovém případě se uživatel do rodiny pouze připojí, ale nemusí zadávat žádné doplňující údaje, které jsou v systému povinné. Pozvání musí daná osoba přijmout, což znázorňuje případ užití *„Přijmout pozvání“*.

Akce *„Odstranit rodinného příslušníka“* pak znamená čistě odstranění rodinného příslušníka z rodiny, ale je nutné rozlišovat rodinné příslušníky registrované uživatelem, kteří nemají přístup ke svému účtu, a pozvané rodinné příslušníky, kteří již svůj účet mají a nelze je tedy odstranit z databáze uživatelů. Více detailů k rodinným příslušníkům je popsáno v sekci Rodinní příslušníci.

Mezi další důležité případy užití, které je vhodné zmínit, patří *„Zaplatit členský příspěvek“*. Tuto akci může uživatel vykonat sám nebo se provádí automaticky pomocí aktéra *„Plánovač úloh“*, který pravidelně stahuje bankovní výpisy, což zobrazuje akce *„Import bankovních výpisů“*. Při této akci se strhává částka za členský příspěvek, pokud má daný uživatel dostatek kreditů v systému. Kreditový systém je popsán v podkapitole 3.4. Případ užití *„Obnovit stav účtu“* aktualizuje počet kreditů všem uživatelům v systému na základě evidence plateb v databázi a pokud je nutné stáhnout nové platební transakce z banky, provede akci pro import bankovních výpisů z bankovního účtu.

Případ užití *„Zaplatit kreditem“* provede zaplacení dané částky z kreditu daného uživatele. Neprovádí se tedy žádný bankovní příkaz, ale jedná se o placení pouze v rámci daného informačního systému.

Další případy užití se týkají akcí a aktivit. Akci lze vytvořit, upravit a smazat. Smazat akci je však možné pouze v době přípravy akce, více viz podkapitola 3.2. Po vytvoření akce se mohou vybrat další pořadatelé akce, kteří pomáhají s realizací dané akce. To je znázorněno případem užití *„Vybrat pořadatele k akci“*, ve kterém lze zároveň vybrat moderátory případného fóra k akci. Není to však nutné, proto je případ užití *„Vybrat moderátory fóra“* připojen k hlavnímu případu užití jako rozšiřující. Pořadatelem může být kterýkoliv člen s platným členstvím. Moderátorem fóra může být pouze vybraný pořadatel či hlavní organizátor akce, který akci založil. V momentě, kdy se člen stane pořadatelem akce, v systému se z něj stává aktér *„Organizátor akce“*.

Po vytvoření akce se člen stává organizátorem akce, z čehož vyplývají další případy užití jako *„Spravovat program akce“*. Tento případ užití znamená veškerou správu aktivit k dané akci (vytvoření, úprava a smazání). Aktivity se dělí na zajištěné a dobrovolné (otevřené). Zajištěné aktivity jsou organizované v rámci akce a nelze do nich zasahovat účastníky dané akce. Naopak dobrovolné aktivity vyžadují pomoc od účastníků akce, aby dodali potřebné prostředky k jejímu uskutečnění. Dobrovolné aktivity může vytvářet i účastník akce, jak je znázorněno případem užití *„Přidat / Odstranit otevřenou aktivitu“*. Je tomu tak proto, aby účastníci akce měli možnost vylepšit program akce. Např. když si hodně lidí přeje zahrát šipky, ale organizátoři akce na to nemají prostředky a rozpočet, můžou to zajistit zájemci (dobrovolníci) o tuto aktivitu. Seznam prostředků k dobrovolné aktivitě je vypsán pomocí případu užití *„Spravovat prostředky otevřené aktivity“*.

Všechny akce je nutné schválit členem výboru, což popisuje činnost *„Schválit akci“*, která zahrnuje případy užití *„Měnit stav akce“* a *„Povolit/zrušit fórum u akce“*. U schválení akce je totiž nutné uvést, zda může mít organizátor fórum ke své akci. Fórum slouží pro členy, kteří se chtějí na něco zeptat nebo popsat svou spokojenost s programem akce apod. (případy užití *„Vytvořit nové vlákno na fóru“* a *„Spravovat své příspěvky na fóru‘“*).

Pokud je fórum povoleno, tak se navíc uvádí režim fóra, který určuje, zda je možné na fóru přispívat bez schválení daného vlákna či příspěvku. Pokud je nutné příspěvek či vlákno nejdřív schválit, zařizují to vybraní moderátoři nebo člen výboru případem užití *„Schválit vlákno/příspěvek“*.

Případ užití *„Měnit stav akce“*, který je samostatným případem užití a zároveň je zahrnut v případě užití *„Schválit akci“*, obecně představuje všechny možné změny stavu akce, jak popisuje podkapitola 3.2.

Poslední případ užití, který by se měl zmínit, je *„Vytvořit šablonu sestavy pro akci“*. Tento případ užití znamená, že lze vytvořit sestavu účastníků akce, sestavu zaplacených záloh či všech plateb, sestavu pro program akce a jiné možné sestavy. O jakou sestavu se jedná určuje právě šablona, do které může organizátor zadat informace, které by si potřeboval zobrazit. Tento případ užití byl do systému přidán proto, jelikož mnoho pořadatelů si potřebuje na místě akce zobrazit seznam účastníků, aby mohli ověřit, zda je daná osoba na akci opravdu přihlášená či která osoba musí doplatit vstupné. Organizátor akce s největší pravděpodobností nebude mít na místě akce notebook, ale jen mobilní telefon, na kterém je zobrazení seznamu v podobě tabulky s mnoha sloupci velmi nepraktické, protože šířka telefonu nedokáže pojmout všechny tyto údaje. Mnohem výhodnější je mít papír s daným seznamem, kam se všechny požadované údaje vejdou a navíc si zde může zapisovat poznámky. Proto po vytvoření sestavy lze tuto sestavu stáhnout či rovnou vytisknout na papír, což znázorňují rozšiřující případy užití *„Stáhnout sestavu“* a *„Vytisknout sestavu“*. Stažení sestavy by mělo být možné v různých formátech jako např. XLSX (MS Excel) nebo PDF.

3.2 Stav akce

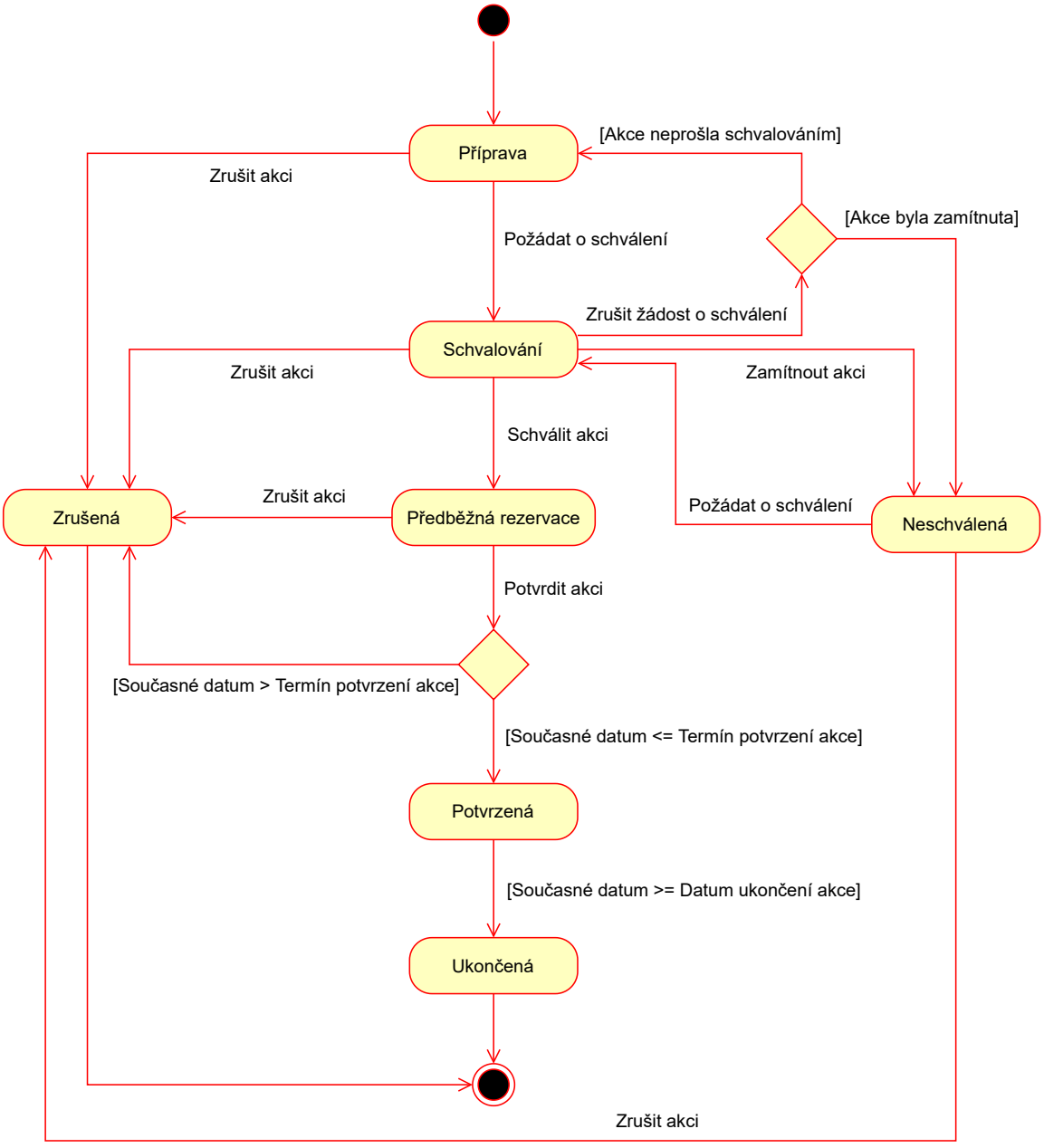
V této podkapitole budou probrány jednotlivé stavy, kterých akce může nabývat. Rovněž bude zmíněno, za jakých stavů lze provádět určité úkony.

Stavový diagram akce

Stavový diagram opět pochází z rodiny behaviorálních diagramů jazyka UML. Slouží k popisu stavů vybraného objektu v systému, kterých může nabývat během svého životního cyklu v reakci na různé události. Životní cyklus lze chápat jako období mezi vznikem a zánikem objektu.

Stavový diagram obsahuje mnoho komponent, kterými lze stavy objektu popsat. Tyto doplňující informace si lze dočíst z literatury [25].

Na obrázku 3.2 je znázorněn diagram stavů pro akci. Každý stav akce bude následně detailněji popsán.



Obrázek 3.2: Stavový diagram akce

Popis jednotlivých stavů:

- *Příprava* – V tomto stavu je akce ihned po jejím založení organizátorem. Lze provádět úpravy, vybírat pořadatele a upravovat program akce. Když je organizátor spokojený se svou akcí, požádá o schválení, jak je znázorněno v diagramu popisem přechodu *„Požádat o schválení“*. Po této události je akce ve stavu *„Schvalování“*.
- *Schvalování* – Tento stav říká schvalovatelům, že existuje akce, kterou je třeba buď schválit, nebo zamítnout. Pořadatelé akce mohou akci v tomto stavu stále upravovat.

- *Předběžná rezervace* – Po schválení akce členem výboru je akce v tomto stavu. Akci je stále možné upravovat, přičemž není nutné žádat o nové schválení akce. Je to z toho důvodu, že jakmile má organizátor jednou schválenou akci, získala tato osoba určitou důvěru, že případné změny nebudou v rozporu s všeobecnými pravidly. Pokud by se však opravdu stalo, že se akce upraví na nevhodnou podobu, člen výboru ji může zrušit. Předběžná rezervace znamená, že se členové mohou začít přihlašovat na akci.

- *Potvrzená* – Potvrzení akce znamená, že je akce jistá a skutečně proběhne. Akci je nutné potvrdit před určitým datem, které je zpravidla 24 – 48 hodin před konáním akce. Tento termín, dokdy je nutné akci potvrdit, nastavuje administrátor globálně v systému. Pokud organizátor nepotvrdí akci do stanoveného termínu, akce bude automaticky zrušena. Potvrzenou akci již nemůžou pořadatelé upravovat (platí i pro program akce) ani zrušit.

- *Ukončená* – Jakmile akce skončí, je automaticky uvedena do tohoto stavu. Stejně jako v případě potvrzené či zrušené akce, nelze tuto akci nijak upravovat.

- *Zrušená* – Zrušená akce zůstává v systému, ale nelze ji obnovit. Obyčejným členům se nezobrazuje v seznamu akcí, pokud na tuto akci nebyli přihlášeni. V takovém případě jsou členům vráceny peníze za akci a případné aktivity.

- *Neschválená* – Pokud člen výboru akci zamítne, bude v tomto stavu. Organizátor může akci upravit a znovu požádat o schválení, pokud si myslí, že již jeho akce splňuje potřebné požadavky. Rovněž může akci zrušit. Z tohoto stavu se nelze dostat do stavu předběžné rezervace.

Možnosti akce na základě stavu

Smazat akci lze pouze ve fázi *„Příprava“*. Smazáním se akce kompletně odebere z databáze systému. Akci nelze smazat v jiných stavech, jelikož je nutné evidovat přihlášené zájemce, platby za zálohy nebo to, že existuje nějaká akce ve stavu *„Schvalování“*, kterou si člen výboru musí prohlédnout a případně schválit či zamítnout. Pokud by se pořadatel rozhodl, že udělal chybu a chtěl by akci smazat, ale akce by již byla ve stavu *„Schvalování“* a nebyla by předtím zamítnuta, může tak stále učinit zrušením žádosti, čímž se akce přesune zpátky do stavu *„Příprava“* a v tomto stavu již lze akci smazat, jak bylo zmíněno na začátku odstavce.

Alternativou smazání je zrušení akce, přičemž bude akce stále existovat v databázi a člen výboru si tak bude moci dohledat případné informace.

Další důležitou součástí schvalovacího procesu je povolení či zrušení fóra. Povolení fóra se provádí při schvalování akce členem výboru, přičemž se zde vybírá i režim fóra, který byl popsán v podkapitole 3.1. Povolit nebo zrušit fórum však lze i po schválení akce, kdy je ve stavech *„Předběžná rezervace“*, *„Potvrzená“*, *„Ukončená“*. Dodatečné povolení fóra je možné proto, aby například po ukončení akce mohli zúčastnění popsat svou spokojenost či naopak možná vylepšení do budoucna. V ostatních stavech nelze fórum povolit nebo zrušit. Výjimkou je zrušení fóra u zrušené akce. Jakmile se akce zruší a fórum bylo povoleno (např. v době předběžné rezervace nebo během schvalování), tak se fórum automaticky nezruší, aby se mohli členové doptat na důvod zrušení či jiné záležitosti. Opětovné povolení již možné není.

Automatické rušení fóra při zrušení akce se provádí jen tehdy, když byla akce zamítnuta. Při schvalování akce je možné povolit fórum i v případě, kdy se akce následně zamítne.

Fórum se sice nemůže vytvořit, dokud nebude akce schválena, ale schvalovatelům to dává informaci, že fórum může být u této akce vytvořeno v případě schválení celé akce.

Členové výboru mohou upravovat akci ve všech stavech. Za úpravu akce se počítá i výběr pořadatelů nebo správa programu.

3.3 Rodinní příslušníci

Existuje více způsobů, jak přidružit rodinné příslušníky k účtu uživatele. Tato podkapitola se jednotlivými způsoby bude zabývat, popíše, jaké podmínky je nutné splnit a jiné požadavky.

V zásadě lze rozdělit přidání rodinného příslušníka na dva způsoby: registrace rodinného příslušníka a pozvání rodinného příslušníka. U registrace nejsou vyplněny přihlašovací údaje, takže se pod takovým účtem nemůže nikdo přihlásit. U pozvání se očekává již existující účet, pod kterým se přihlásit lze. Pokud ovšem uživatel dodá při registraci rodinného příslušníka jeho e-mailovou adresu, tak tato osoba může dokončit registraci zadáním přihlašovacích údajů a následně již bude možné se účtem přihlašovat. Detaily k přidání rodinného příslušníka jsou popsány tabulkou 3.1.

Podmínky a omezení	**Registrace**	**Pozvánka**
Zadaná e-mailová adresa	Ne	Ano
Lze se přihlašovat	Ne	Ano
Uživatel může měnit údaje[2]	Ano	Ne
Aktivní účet	Ne	Ano

Tabulka 3.1: Přidání rodinného příslušníka

Když uživatel zaregistruje rodinného příslušníka, může mu hned nebo i později zadat e-mailovou adresu. Pokud tak učiní, odešle se na tento e-mail pozvánka v podobě odkazu. Po kliknutí na odkaz může osoba dokončit registraci a stát se tak plnohodnotným uživatelem, který se může přihlašovat do systému.

Pokud se rodinný příslušník zaregistroval už dříve, může uživatel poslat pozvánku na jeho e-mailovou adresu. Poté jen stačí, aby rodinný příslušník pozvánku přijal a přidá se do rodiny. Nemusí vyplňovat žádné doplňující údaje. Podmínkou však je, aby daná osoba nebyla již v jiné rodině a měla aktivní účet.

Obě pozvánky mají nastavenou dobu expirace. Pozvánky by mělo být možné v systému prodloužit nebo zrušit. V případě, že se pozvánka zruší nebo vyprší doba platnosti, stává se pozvánka neplatnou a rodinný příslušník ji nemůže použít pro přidání se k rodině.

Všechny rodinné příslušníky si může uživatel zobrazit v podobě seznamu. Nerozlišuje se, kdo poslal pozvánku či zaregistroval některého rodinného příslušníka. Pokud existuje v rodině více uživatelů s platným účtem, mohou provádět stejné operace jako uživatel, který je pozval. To například znamená, že kdokoliv v rodině s platným účtem může zaplatit členství sobě a dalším rodinným příslušníkům, které může vybrat jednotlivě dle uvážení.

Měnit údaje rodinnému příslušníkovi lze pouze v případě, že nemá platný účet s přihlašovacími údaji (nemůže se sám přihlásit) nebo není zletilý. Odstranit rodinného příslušníka

[2]Uživatel může měnit údaje rodinnému příslušníkovi.

může uživatel kdykoliv. Záleží však na tom, zda má rodinný příslušník platný účet nebo ne. Pokud má rodinný příslušník platný účet s přihlašovacími údaji (nezávisle na věku), odstranění znamená pouze odebrání v rámci rodiny, přičemž jeho účet bude fungovat nadále. Pokud rodinný příslušník nemá platný účet s přihlašovacími údaji, má uživatel, který rodinného příslušníka z rodiny odebírá, dvě možnosti:

- Odstranění rodinného příslušníka s ponecháním jeho údajů
- Odstranění rodinného příslušníka s vymazáním jeho údajů

V případě, že se uživatel rozhodne ponechat údaje, tak se může tato osoba sama zaregistrovat, jestliže vyplní všechny osobní údaje správně včetně e-mailové adresy. Pokud je registrace úspěšná, nepřidá se nově registrovaný uživatel do rodiny jako tomu bylo v případě pozvánky. V případě, že se uživatel rozhodne vymazat údaje, tak tento způsob registrace není možný, jelikož se všechny osobní údaje rodinného příslušníka zneplatní a nebude možné je zpětně dohledat.

Rodinný příslušník může z rodiny odejít. Pokud tak učiní a jsou v rodině jiní rodinní příslušníci s platným účtem, rodina je zachována. Jestliže se však jedná o posledního uživatele s platným účtem, rodina je rozpuštěna, všechny pozvánky jsou zrušeny a rodinní příslušníci, kteří nemají vlastní účet, jsou automaticky smazáni bez možnosti se zaregistrovat.

3.4 Kreditový systém a členské příspěvky

Kreditový systém je systém, který registrovaným uživatelům umožňuje dobíjet si kredit zasláním peněžitého příspěvku na transparentní účet spolku. Tento kredit pak mohou využít pro placení členských příspěvků či záloh za akce. Veškeré položky, které lze kreditem platit, jsou uvedeny v seznamu níže. Výhodou tohoto systému je, že nemusí každý členský příspěvek či zálohu na akci platit bankovním převodem. Místo neustálé práce s bankou proto uživatelům stačí, aby si dobili částku v systému (kredit), jejíž výši uznají za vhodnou, a z té následně čerpali. V případě, že se některá akce či aktivita zruší, je uživatelům kredit automaticky vrácen a není třeba žádat o vrácení peněz. Tento vrácený kredit pak mohou použít na jiné akce.

Veškeré položky, které lze kreditem platit, jsou shrnuty zde:

- *Členské příspěvky* – Člen může platit členské příspěvky i za své rodinné příslušníky, přičemž to pak mají tito členové levnější. Výši členských příspěvků určuje administrátor v nastavení po rozhodnutí členů výboru.
- *Akce* – Lze platit zálohy za akci po přihlášení, aby tím člen stvrdil svůj zájem. Doplatek je pak vybrán na místě akce. Je ovšem možné zaplatit i celou cenu vstupného přímo v systému.
- *Aktivity* – Člen se může přihlásit na určité typy aktivit v rámci akce a pokud nejsou v ceně akce, je nutné za ně opět zaplatit zálohu či plnou cenu.
- *Příspěvky na akce a aktivity* – Člen může přispět na akci či aktivitu, pokud je malý rozpočet, čímž pomůže organizátorovi k realizaci.

Dále je možné implementovat automatické strhávání členských příspěvků od stanoveného data, což ve výsledku znamená, že uživatelé systému nemusí myslet na každoroční

placení svého členství. Příspěvek se automaticky strhne i v případě, kdy si kredit navýší později.

Jelikož se platby stahují z banky pravidelným importem, který se bude provádět např. jednou denně v noci, tak se uživatelům ne vždy zobrazí aktuální data (počet kreditů, zaplacený členský příspěvek). Proto mohou uživatelé provést obnovu účtu sami, což znamená, že se provede stejná akce jako v případě pravidelného importu. Musí se však nastavit časové omezení, aby se v případě, kdy se snaží tuto akci provést více uživatelů najednou, neposílalo mnoho stejných požadavků a systém se nezahltil. Navíc i samotná banka má nastavený minimální interval pro posílání dotazů. Administrátor tedy nastavuje globální interval, jenž určuje, jak často lze takto stahovat bankovní výpisy. Tento globální interval platí pro všechny, takže například pokud jeden uživatel zažádá o obnovu účtu a ta se provede, další uživatelé nemohou tuto akci provést, dokud neuběhne nastavený časový interval.

Kapitola 4

Využité technologie

V této kapitole bude popsán a zdůvodněn výběr některých technologií. Kromě technologií, které jsou zmíněny v jednotlivých podkapitolách, jsem rovněž použil základní technologie, a sice HTML[1] (HyperText Markup Language) pro vytvoření rozložení výsledných stránek, CSS[2] (Cascading Style Sheets) pro nastavení vzhledu všech stránek a nakonec deklarativní jazyk SQL (Structured Query Language) pro práci s databází MySQL.

4.1 PHP – PHP Hypertext Preprocessor

PHP je imperativní (procedurální)[3] interpretovaný[4] programovací jazyk, který je od verze 5 i objektově orientovaný. Autorem jazyka je **Rasmus Lerdorf**. Slouží k vytváření dynamických webových stránek, kdy se na serveru zpracuje zdrojový kód v PHP a výsledkem je odpověď protokolem HTTP[5] (nejčastěji ve formátu HTML), která se pošle klientovi.

Nový informační systém pro spolek ČMFN používá PHP verze 7.4, neboť se jedná o stabilní verzi s plnou podporou objektově orientovaného programování. Existují sice novější verze (např. PHP 8), ale webový hosting, na kterém se nachází informační systém spolku, podporuje PHP pouze do verze 7.4. V současné době existuje mnoho jiných programovacích jazyků, které umožňují dynamické vytváření webových stránek (např. C# (ASP.NET), Python (Flask/Django), JavaScript (Node.js) a další), ale PHP je stále velmi rozšířené a nové verze jazyka podporují téměř to stejné, co jeho konkurenti.

Při vypracovávání této práce jsem čerpal z knihy uvedené v literatuře [29] a z oficiální dokumentace k jazyku PHP [26]. Obsahuje mnoho užitečných informací, ať už se týkají práce s databází či popisu konceptů objektově orientovaného programování v jazyce PHP.

[1] https://developer.mozilla.org/en-US/docs/Web/HTML

[2] https://developer.mozilla.org/en-US/docs/Web/CSS

[3] Imperativní (procedurální) jazyk je takový jazyk, jenž má danou přesnou posloupnost příkazů, které se vykonají postupně v daném pořadí.

[4] Interpretovaný jazyk je takový jazyk, kdy se analýza zdrojového kódu provádí za běhu a vykonání kódu zařizuje interpret, což je program pro daný jazyk.

[5] Více informací k protokolu HTTP se lze dočíst na stránce https://datatracker.ietf.org/doc/html/rfc2616.

Frameworky

„Framework je sada knihoven, které v obecné podobě implementují časté programátorské konstrukce a usnadňují tak vývojářům rutinní práci. Bývají zaměřeny na určitou vymezenou oblast, např. tvorbu webových aplikací. Často jsou v nich aplikovány různé návrhové vzory a best practices z dané oblasti.“ [17]

Pro jazyk PHP existuje mnoho frameworků, které velmi usnadňují vytváření komplexních webových aplikací. Mezi nejpoužívanější frameworky se řadí Laravel, Symfony a Nette. Nette je český PHP framework, který je používán zejména v České republice. Byl rovněž zvolen jako vývojová podpora pro tento rezervační systém. Více informací o tomto frameworku je možné se dočíst v podkapitole 4.2.

4.2 Nette – Český PHP framework

Jedná se o PHP framework, který se skládá z několika samostatných částí. Tento framework jsem zvolil, protože má výbornou podporu pro formuláře a zpracování webových stránek pomocí *snippetů* s využitím technologie AJAX[6]. Snippety jsou označené úseky webové stránky pro překreslení. Zároveň umožňuje vytvářet a využívat komponenty, což jsou samostatné znovupoužitelné objekty, jež lze vkládat do stránek (např. formuláře).

Autorem tohoto frameworku je **David Grudl**. Současná verze je 4.0, ta však podporuje pouze PHP verze 8.0 a vyšší. Pro účely tohoto projektu jsem tedy vybral verzi 3.1, která je kompatibilní s PHP verze 7.4. Framework je udržován samotným autorem a komunitou **Nette Foundation**. Je šířen jako svobodný software pod licencemi New BSD nebo GNU General Public License verze 2 nebo 3 [15].

Nette disponuje řadou rozšíření, což je dalším důvodem zvolení tohoto frameworku. Použil jsem následující rozšíření:

Název	**Verze**	**Licence**	**Popis**
Contributte/Translation [9]	1.0	MIT	Knihovna použitá pro různou lokalizaci textů umístěných na webové stránce.
Contributte/Forms-Wizard [8]	3.1	MIT	Knihovna, která umožňuje vytvářet vícekrokové formuláře (využito u registrace uživatele).
Contributte/Webpack [10]	2.1	BSD-3-Clause	Tato knihovna slouží pro snadnou integraci technologie Webpack, která je popsaná v podkapitole 4.5, do projektů Nette.

Tabulka 4.1: Seznam použitých rozšíření pro Nette Framework

Nette má rovněž vlastní šablonovací systém zvaný Latte [16]. Díky tomu jsou všechny texty vložené na stránce ošetřeny proti útoku XSS (Cross-Site Scripting)[7]. Pomocí šablo-

[6] Asynchronous JavaScript and XML – Technologie, která umožňuje dotazovat se serveru bez nutnosti znovu načítat obsah stránky.

novacího systému lze navíc správně oddělit aplikační logiku od pohledů a také nebude kód v šabloně znečištěn zdrojovým textem v jazyce PHP.

Framework Nette mimo jiné umožňuje vytvářet si vlastní Latte makra, což jsem využil u vykreslování formulářových polí s chybovými hláškami [21].

4.3 JavaScript

Jedná se o interpretovaný, z části funkcionální, prototypovaný a objektově orientovaný skriptovací jazyk s dynamickým typováním. Dynamické typování znamená, že se typ proměnných určuje za běhu skriptu. Jeho autorem je **Brendan Eich** a vznikl v roce 1995. V roce 1998 byl standardizován pod názvem *ECMAScript*, což v průběhu dalších let zajistilo kompatibilitu mezi více prohlížeči webových stránek [5].

Tento jazyk se používá zejména pro tvorbu dynamických webových stránek na straně klienta. Umožňuje přistupovat k elementům jazyka HTML pomocí aplikačního rozhraní DOM (Document Object Model). Pro JavaScript existuje spousta knihoven, pomocí kterých si lze zjednodušit vytváření některých často používaných komponent (např. zobrazení modálních oken).

Pro vývoj na tomto projektu jsem zvolil následující knihovny:

Název	**Verze**	**Licence**	**Popis**
Tempus Dominus [23]	6.0.0	MIT	Umožňuje jednoduše zobrazit okno pro výběr data a času. To je velmi užitečné u vstupních polí, kde je nutné uvést datum, popřípadě čas.
Bootstrap Table [30]	1.20.2	MIT	Doplněk pro HTML tabulky, který přidává řazení, filtrování, stránkování a jiné možnosti. Lze jej použít i bez použití stylovacího frameworku Bootstrap.
Naja[8]	2.3.0	MIT	Knihovna pro AJAXové dotazy na straně klienta. Je speciálně navržena pro Nette Framework a umí tedy dynamicky zpracovávat a překreslovat snippety na stránce.
TinyMCE [27]	6.0.2	MIT	JavaScriptový textový editor s mnoha funkcemi pro formátování textu. Bude využit především u zadávání podrobných popisů pro akce a aktivity.

Tabulka 4.2: Seznam použitých knihoven pro JavaScript

[7] Útok, který funguje tak, že se do databáze vloží text se značkami `<script>`, které obsahují škodlivý kód v jazyce JavaScript, jenž se vykoná po vložení tohoto neošetřeného textu na stránku.

[8] Dostupné z: https://naja.js.org/ (autorem je **Jiří Pudil**)

4.4 jQuery

jQuery je optimalizovaná knihovna, která zjednodušuje práci s JavaScriptem. Poskytuje rychlé řešení pro získání elementů HTML pomocí selektoru jazyka CSS, dále podporuje zasílání asynchronních požadavků protokolem HTTP s využitím technologie AJAX. Výhodou je především kompatibilita nabízených metod napříč různými prohlížeči. Knihovna jQuery sama zjistí dostupnost některých vlastností implementace JavaScriptu v závislosti na použitém prohlížeči a zajistí v různých prohlížečích stejnou funkcionalitu.

Tvůrcem této knihovny je společnost **OpenJS Foundation** a nabízí popisovanou knihovnu pod licencí MIT s otevřeným zdrojovým kódem. V projektu je použitá verze 3.6.0, což je zároveň poslední dostupná a stabilní verze této knihovny [22].

Implementace některých klientských částí by se sice obešla bez této knihovny, kterou dnes nahrazují komplexnější frameworky jako například React, Angular či Vue.js, ale v projektu je pro front-end využíván framework *Semantic UI*, který použití knihovny jQuery vyžaduje. Více informací o *Semantic UI* se lze dočíst v příslušné podkapitole níže.

V předchozím odstavci byly zmíněny JavaScriptové frameworky typu React, které se zaměřují na klientskou webovou aplikaci, přičemž server pak slouží pouze jako REST API[9]. Tento přístup se označuje jako CSR (Client Side Rendering), kdy výsledná podoba stránky je sestavena přímo na klientském počítači. Projekt tohoto typu však nemá tak rozsáhlou klientskou část, aby se vyplatilo použít podobné řešení. Proto se využívá přístup SSR (Server Side Rendering), kdy je výsledná podoba stránky sestavena přímo na serveru pomocí připravených šablon a tento výsledek je poté vrácen ve formátu HTML klientovi.

4.5 Webpack

Webpack je nástroj napsaný v JavaScriptu, který slouží pro správu různých webových zdrojů (obrázky, fonty, styly, skripty v jazyce JavaScript apod.). V zásadě to znamená, že všechny soubory v jazyce JavaScript, které mohou mít různé závislosti na jiných skriptech či modulech, se spojí do jednoho minifikovaného souboru, což znamená, že se odstraní všechny bílé znaky, komentáře a díky tomu má takový soubor mnohem menší velikost. To je velmi důležité při přenosu těchto souborů po síti, jelikož soubor s menší velikostí se přenáší rychleji. Rovněž umožňuje zjistit závislosti mezi jednotlivými skripty a extrahovat je tak, aby se vložily na stránku jen jednou a nedocházelo ke zbytečné duplicitě a tedy větší velikosti souboru. Obdobně to platí pro soubory se styly (CSS).

Webpack je vydán pod licencí MIT organizací **JS Foundation**. Zakladatelem Webpacku je **Tobias Koppers**. Pro účely tohoto projektu jsem použil nejnovější vydanou verzi 5 [18].

4.6 Semantic UI

Jedná se o responzivní framework pro front-end používaný pro tvorbu uživatelsky přívětivých rozhraní. Obsahuje předdefinované třídy CSS, které lze použít přímo v kódu stránky v jazyce HTML. Mimo jiné řeší i různé dynamické části pomocí jQuery (např. zobrazování modálních oken). Tento framework se skládá z mnoha komponent, které je však možné použít samostatně bez nutnosti vkládání celého frameworku.

[9] Webová aplikace na serveru čeká na požadavky protokolem HTTP a poté vrací odpověď nejčastěji ve formátu JSON. Nezabývá se zobrazováním stránek pro uživatele.

To je vhodné zejména pro tento projekt, jelikož vzhled stránky v podobě stylů v jazyce CSS byl dodán spolkem ČMFN. Použitím jednotlivých komponent z tohoto frameworku nedojde k narušení již existujících stylů. Použil jsem pouze komponentu pro zobrazování modálních oken[10] a k tomu menší dodatečné komponenty, na kterých je tato komponenta závislá a bez kterých by zobrazení modálního okna nefungovalo korektně.

Celá knihovna předpřipravených uživatelských komponent je šířena pod licencí MIT s otevřeným zdrojovým kódem. Projekt je udržován zejména komunitou pomocí pull requestů v příslušném repozitáři na hostingu GitHub.

4.7 Font Awesome

Font Awesome je rozsáhlá knihovna ikon pro použití v dokumentech v jazyce HTML. Tuto knihovnu jsem použil jen z toho důvodu, že JavaScriptová komponenta pro výběr data a času (Tempus Dominus) používá ikony pro popis svých tlačítek. Zdarma dostupná varianta této knihovny je šířena pod dvěma licencemi: CC BY 4.0 se vztahuje na ikony a SIL OFL (Open Font License) 1.1 na použité fonty. Knihovna je distribuována společností **Fonticons, Inc.** Použil jsem poslední verzi 6.1.1[11].

[10] Dostupné z: https://semantic-ui.com/modules/modal.html
[11] Dostupné z: https://fontawesome.com/docs/web/setup/packages

Kapitola 5

Návrh rezervačního systému pro akce

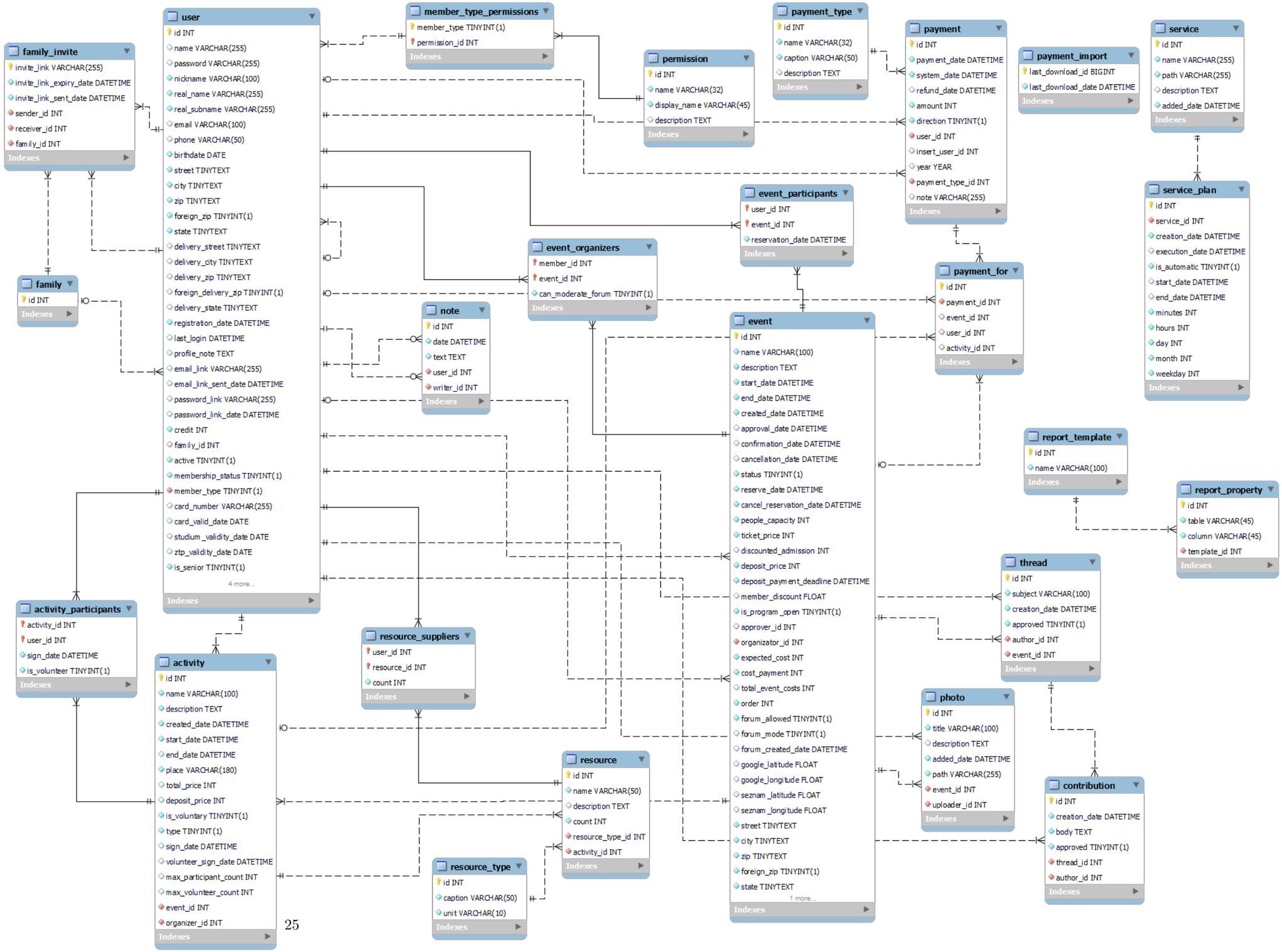
Po zjištění všech požadavků na nový systém je možné začít s návrhem databáze, uživatelského rozhraní a jiných implementačních detailů.

5.1 Nový ER diagram

Pro návrh relační databáze bude využit ER diagram (Entity Relationship Diagram). Tento diagram se skládá z **entitních množin** a **vztahových množin**. Následující vysvětlení je převzato a mírně upraveno z [31].

Entitní množina je množina entit stejného typu, které sdílí tytéž vlastnosti. Entita je objekt reálného světa rozlišitelný od ostatních objektů, o níž potřebujeme znát informace v databázi. **Vztahová množina** je množina vztahů téhož typu. Vztah je potom konkrétní spojení dvou nebo více entit a určuje, jaká je mezi těmito entitami závislost.

Následuje ER diagram návrhu nové databáze a poté popis nejdůležitějších entitních množin, jejich význam a odůvodnění daného řešení.



Obrázek 5.1: ER diagram nově navržené databáze

Uživatelé

Na obrázku 5.1 lze vidět nově navržený ER diagram. Základem je entitní množina `user`, která popisuje všechny zaregistrované uživatele. Oproti předchozímu návrhu na obrázku 2.1 zde již není tabulka pro uchování rodinných příslušníků, nýbrž se rodinní příslušníci přidávají do stejné tabulky jako nově zaregistrovaní uživatelé. To je také důvodem, proč atributy `name` a `password` jsou nepovinné. Když se přidává nový rodinný příslušník, tak za něj uživatel nemůže zadat přihlašovací jméno a heslo. Tyto údaje jsou vyžadovány až při případné registraci daného rodinného příslušníka. Proto je nutné umožnit těmto atributům nabývat hodnoty `NULL`. Při registraci jsou ovšem tyto údaje vyžadovány a tudíž se nestane, že by se někdo mohl zaregistrovat bez hesla a přihlašovacího jména.

Tato tabulka rovněž zahrnuje údaj o tom, kolik má uživatel kreditů. Kredit se pak při každém importu bankovních výpisů aktualizuje. Když uživatel zaplatí členský příspěvek, tak se nastaví atribut `membership_status`. Ten určuje, zda má uživatel platné nebo vypršené členství, případně že byl vyloučen. Údaj `card_valid_date` se po zaplacení členského příspěvku nastaví vždy na konec ledna dalšího kalendářního roku (dle nastavení) a určuje, dokdy má uživatel platné členství.

Registrovaný uživatel se může stát členem pouze pokud má aktivní účet. Bez aktivního účtu rovněž nelze přidávat rodinné příslušníky. Z toho důvodu se po registraci zasílá odkaz na zadanou e-mailovou adresu, který slouží k ověření této e-mailové adresy. Po úspěšném ověření e-mailové adresy má uživatel účet aktivován a získá přístup ke správě rodiny. Také se mu po aktivaci účtu aktivuje členství, jestliže si dobil dostatečnou částku na svůj kreditový účet.

Automatická aktivace členství je potřebná i z toho důvodu, že příjmem spolku jsou primárně členské příspěvky a na pořádané akce jsou následně vynaloženy výdaje hrazené z peněz vybraných za členské příspěvky. Pokud by uživatel nebyl členem a členský příspěvek nebyl započítán automaticky, platba vedoucí k získání kreditu by nemohla být brána za členský příspěvek a nakládání s kreditem by tak nemohlo být pouze interním mechanismem pro přerozdělení peněz na akce, ale muselo by být samostatnou činností spolku.

Atribut `member_type` vyjadřuje typ uživatele dle aktérů v diagramu případů užití (viz seznam aktérů v podkapitole 3.1).

Rodina a pozvánky

Jelikož v rodině může být více aktivních uživatelů a každý má stejná práva, nerozlišuje se, kdo pozval kterého rodinného příslušníka. Všichni tedy mohou platit za vybrané rodinné příslušníky nebo posílat pozvánky novým členům. V případě, že někdo z rodiny odejde, systém by neměl automaticky rozpustit rodinu, pokud v ní existuje někdo s aktivním účtem. Proto byla přidána tabulka `family`, která obsahuje pouze identifikátor `id`. Uživatelé, co patří do rodiny, mají nastavený atribut `family_id`, který se odkazuje právě na identifikátor z tabulky `family`. Díky tomu má každý stejná práva a nevadí, když někdo z rodiny odejde.

Příkladem může být rodina, kde jsou dva dospělí zaregistrovaní uživatelé a mají tam přidané své dítě, které však nemá přístup k účtu. Pokud se jeden z těchto rodičů rozhodne odejít, druhý rodič stále může spravovat rodinu a bude mít v seznamu své dítě, se kterým se může přihlašovat na akce.

Tabulka `family_invite` eviduje poslané pozvánky. Primárním klíčem je unikátní vygenerovaný token, který je poslán v odkazu na zadanou e-mailovou adresu. Tento odkaz s tokenem tedy slouží jako pozvánka do rodiny. Tabulka má rovněž atribut `family_id`. Důvodem je následující příklad: Pokud by uživatel v rodině poslal někomu pozvánku, poté z ro-

diny odešel a v tabulce `family_invite` by nebyla reference na rodinu (atribut `family_id`), tak by nebylo možné dohledat, do jaké rodiny pozvánka patří, jelikož atribut `family_id` z tabulky `user` se nastaví na `NULL` jako důsledek odchodu. Proto je nutné evidovat v tabulce `family_invite` referenci na rodinu, ze které pocházel uživatel, jenž pozvánku zaslal.

Nicméně uživatel, co z rodiny odchází a poslal nějaké pozvánky, má možnost své pozvánky zrušit a nechat vymazat z databáze.

Platby

Velmi důležitá tabulka je `payment_for`. Tato tabulka určuje, co bylo předmětem platby. Položky, které mohou být předmětem platby, jsou zmíněny v sekci 3.4. Pokud uživatel platí členský příspěvek sám za sebe, tabulka `payment_for` se nevyužije a nastaví se pouze atribut `year` z tabulky `payment` na rok, za který je členský příspěvek placen.

Pokud však uživatel platí i za své rodinné příslušníky, nastaví se nejen rok v tabulce `payment`, ale zároveň se přidají záznamy do tabulky `payment_for` s nastaveným atributem `user_id`, který vyjadřuje ID rodinného příslušníka, za kterého je placeno. Tento atribut se použije vždy, když uživatel platí za své rodinné příslušníky (např. když za ně platí zálohu na akci).

Další atribut z tabulky `payment_for` je `event_id`. Tento atribut se nastaví, když je předmětem platby záloha či plná cena vstupného za akci. Následující atribut `activity_id` odkazuje na aktivitu, za kterou byla zaplacena záloha či plná cena. Když se platí za aktivitu, není třeba uvádět do tabulky `payment_for` i identifikátor akce, ke které aktivita náleží, neboť primární klíč aktivity není složený z ID akce a ID aktivity, nýbrž primárním klíčem je pouze unikátní ID aktivity. To znamená, že každá aktivita bude mít své vlastní jedinečné ID nezávisle na tom, ke které akci patří.

V sekci 3.4 byl zmíněn dobrovolný příspěvek na akci. Atributy v tabulce `payment_for` se nastaví stejně, jako kdyby za sebe uživatel platil zálohu. O co konkrétně se jedná tedy rozlišuje atribut `payment_type_id`, jenž odkazuje na tabulku `payment_type`. Tato tabulka definuje jednotlivé typy plateb (např. příspěvek na akci, platba členského příspěvku apod.). Záznamy v této tabulce spravuje administrátor, přičemž webová aplikace spoléhá na to, že dané typy plateb v databázi existují. Podrobnější detaily jsou popsány v implementaci, viz kapitola 6. Tabulka obsahuje unikátní atribut `name`, který slouží jako klíč, pomocí kterého webová aplikace vkládá do databáze určitý typ platby nebo jej získává, přičemž atribut `caption` poté slouží jako uživatelsky přívětivý název druhu platby.

Platbu lze vložit ručně a může tak učinit pokladník nebo někdo, kdo má právo vložit platbu. Atribut `insert_user_id` proto eviduje, kdo platbu vložil. Pokud je hodnota atributu `NULL`, znamená to, že byla platba importována z banky, případně se jedná o platbu kreditem v rámci systému. S tím souvisí atribut `direction`, který udává „směr“ platby. V podstatě rozděluje platby na **příchozí**, které byly importovány z bankovního systému, a na **odchozí**, jež znamenají platbu kreditem. Atribut `system_date` popisuje, kdy byla platba importována do systému, a tudíž má význam jenom pro příchozí platby. Pro odchozí platby bude hodnota tohoto atributu vždy shodná s datem zaplacení (`payment_date`).

Šablony sestav

Jelikož sestavy by měly být univerzální a člen by měl mít možnost si zvolit, jaké informace chce zobrazit či vytisknout, byla vytvořena entitní množina `report_property`. Ta popisuje, jaký atribut z jaké tabulky zahrnout do výsledné sestavy. Vyžaduje tedy název tabulky (`table`), což může být název kterékoliv tabulky z databáze, u které má smysl vytvářet

sestavu. Atribut `column` poté vyjadřuje název atributu z dané tabulky, jehož hodnota se bude vypisovat do výsledné sestavy pro všechny záznamy existující v zadané tabulce.

Současný návrh avšak neumožňuje lokalizaci, tudíž názvy atributů by se musely překládat v samostatném souboru mimo databázi, jinak by se použil anglický název atributu z databáze (např. confirmation_date ⇒ Confirmation Date).

Tabulka `report_template` poté jen popisuje název vytvořené šablony a pomocí vztahu 1:M lze zjistit, jaké informace do sestavy zahrnout. Díky tomu, že se ukládají vytvořené šablony do databáze, je lze znovu použít i pro jiné akce, uživatele apod.

Plánovač úloh

Naplánované skripty pro automatické spouštění se ukládají do tabulky `service_plan`. Tato tabulka specifikuje periodu spouštění podobným způsobem, jako se to uvádí v souborech *crontab*[1]. Také umožňuje zadat, od kterého data se má daný skript začít spouštět (atribut `start_date`), případně kdy se má automatické spouštění skriptu zastavit (atribut `end_date`). Oba atributy jsou nepovinné, tudíž pokud není žádné datum specifikováno, bude se skript v zadaných intervalech spouštět napořád.

Lze vytvořit plán, kdy se daný skript spustí pouze jednou v předem určený den a čas bez dalšího opakování. To je možné nastavit zadáním požadovaného termínu do položky `execution_date`. Poslední atribut `is_automatic` je příznak, zda bylo spuštění služby automaticky naplánováno systémem (např. když uživatel požádá o obnovu účtu před vypršením globálního intervalu, přidá se obnova účtu do plánu úloh).

Atribut `service_id` odkazuje na službu z tabulky `service`, která se má spouštět. Tato tabulka popisuje cestu ke spouštěnému skriptu a jeho název pro zobrazení na webové stránce. Název by měl být unikátní stejně jako cesta k souboru. Je to z toho důvodu, aby se stejný skript nespouštěl vícekrát v různých nebo i totožných intervalech.

5.2 Oprávnění uživatelů

Oprávnění jsou navržena tak, že se musí vložit do databáze před spuštěním informačního systému. Informační systém totiž používá názvy oprávnění pro ověření, zda má uživatel autorizaci k provedení určité operace.

Jednotlivá práva jsou uložena v tabulce `permission`, přičemž atribut `name` označuje název oprávnění používaný ve zdrojovém kódu informačního systému. Naopak atribut `display_name` slouží pro uživatelský název daného oprávnění. Práva jsou přiřazena uživatelům na základě jejich typu, který uchovává vlastnost `member_type` v tabulce `user`. To znamená, že nelze přiřadit práva jednotlivým uživatelům, ale je možné přiřadit práva pouze celé skupině uživatelů.

Seznam oprávnění, která jsou přiřazená k danému typu uživatele, je uložen v tabulce `member_type_permissions`. Framework Nette sice obsahuje nativní podporu pro oprávnění uživatelů na základě prostředků a k nim přiřazeným operacím, ale tento způsob by nebyl příliš vhodný. Nastavovat oprávnění podle prostředků (mohly by být např. tabulky v databázi) by bylo zbytečně pracné na návrh v databázi a navíc je pravděpodobné, že by byl takový způsob složitý i pro členy výboru, kteří by tato práva museli nastavovat. Proto byl zvolen výše uvedený způsob.

[1] Formát zadávání plánu spouštění CRON služeb lze najít na tomto odkazu: http://www.nncron.ru/help/EN/working/cron-format.htm

5.3 Časová pásma

Systém by měl umožňovat vytvářet akce po celé Zemi. Problém však nastává při zadávání data konání a data konce akce. Čas na Zemi se v různých oblastech liší, a proto je třeba při zadávání časů uvažovat časové pásmo.

Časové pásmo vytvořené akce je ukládáno v položce `timezone` v tabulce `event`. Po vytvoření akce jsou však všechny časy převedeny do časové zóny nastavené v konfiguraci (výchozí je Europe/Prague), aby uložené časy v databázi byly konzistentní se zbylými daty v jiných tabulkách (např. `user`).

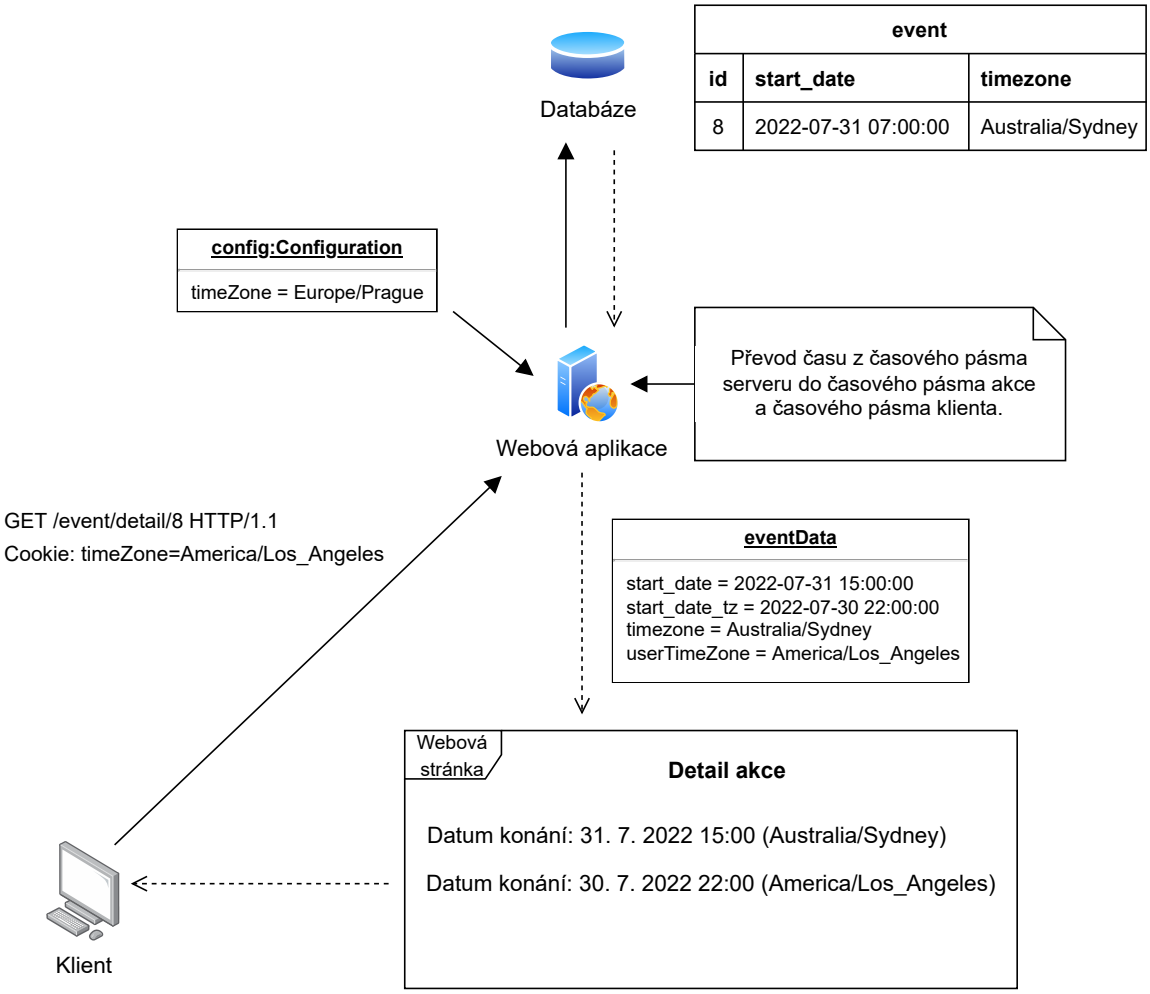
Jelikož uživatel může být při používaní informačního systému na různých místech Země, může se jeho časové pásmo lišit od časového pásma nastaveného v konfiguraci, případně se může lišit i od časového pásma akce. Proto se na serveru časy získané z databáze převádí do klientského časového pásma, které je získáno pomocí *cookies* obsažených v požadavku protokolem HTTP od klienta.

Pokud si uživatel prohlíží akci, která je rovněž v jiném časovém pásmu, než ve kterém je klient, tak se navíc časy převedou do časového pásma dané akce. Tím vznikají časy ve dvou různých pásmech: v časovém pásmu klienta a v časovém pásmu vytvořené akce. Názorná ukázka toho, jaké časy mohou vzniknout, když se budou lišit všechna zmíněná časová pásma (akce, klient a databáze), je vyobrazena v tabulce 5.1. Kompletní proces převodu časových pásem následně zobrazuje diagram na obrázku 5.2.

Při navrhování řešení jsem čerpal ze zdrojů [12] a [2].

	Časové pásmo	**Datum a čas**
Databáze	Europe/Prague (UTC+02:00)	31. 7. 2022 07:00
Klient	America/Los_Angeles (UTC-07:00)	30. 7. 2022 22:00
Akce	Australia/Sydney (UTC+10:00)	31. 7. 2022 15:00

Tabulka 5.1: Příklad uložení času v různých časových pásmech



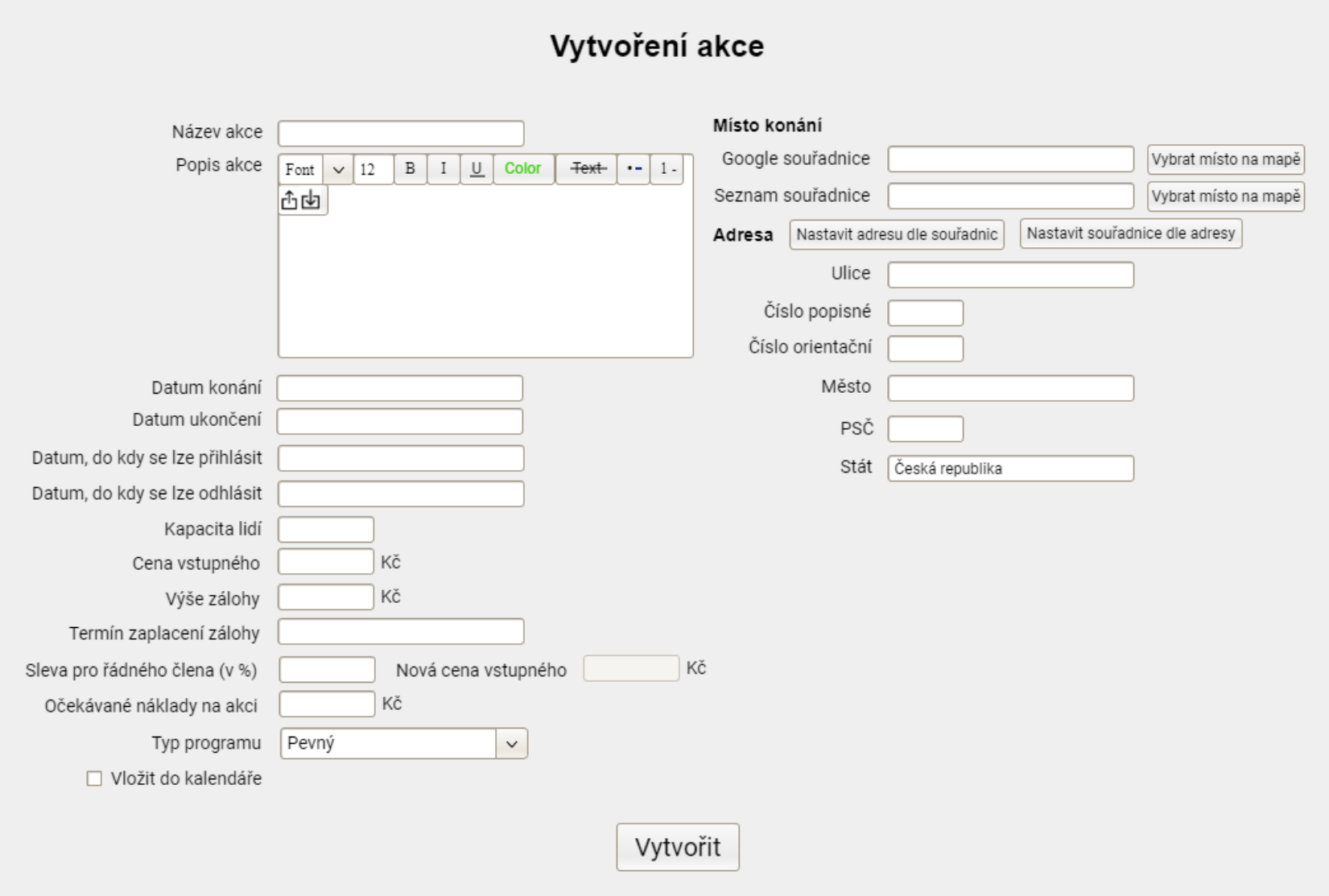
Obrázek 5.2: Postup převodu časového pásma

5.4 Návrh uživatelského rozhraní

V této podkapitole bude popsán návrh některých pohledů v systému. Většina pohledů se bude týkat především akcí a aktivit, jelikož návrh uživatelského rozhraní pro tyto části systému byl nejvíce kritický. Pokud má rezervační systém sloužit pro prezentaci různých událostí, měl by být uživatelsky přívětivý pro uživatele, jež tyto události budou vytvářet a spravovat.

Vytvoření akce

Prvním důležitým návrhem uživatelského rozhraní je právě vytvoření samotné akce. Podoba rozhraní je znázorněna na obrázku 5.3. Pole *„Popis akce“* je textový editor, který umožňuje formátovat text různými způsoby. Jako editor byl zvolen *TinyMCE* zmíněný v tabulce 4.2. Dále lze na obrázku vidět vstupní pole pro zadání data a času. Všechna tato pole využívají JavaScriptovou komponentu pro výběr data a času rovněž zmíněnou v tabulce 4.2.



Obrázek 5.3: Návrh pohledu pro vytvoření a úpravu akce

V této komponentě lze nastavit různá omezení pro datum a čas (např. nelze zadat datum ukončení před datem konání a naopak). Na základě toho je možné dynamicky omezit jednotlivá vstupní pole týkající se času. Validace by se samozřejmě měla provádět i na straně serveru, kdyby si uživatel daný skript vypnul nebo upravil.

Lze si povšimnout, že zde není uvedená časová zóna, jak bylo popisováno v podkapitole 5.3. Důvodem je to, že při návrhu této obrazovky uživatelského rozhraní jsem opomněl možnost vytvoření akce i na jiném místě než v České republice. Na tuto skutečnost se narazilo až při konzultaci návrhu s budoucími uživateli a bylo to zohledněno během implementace.

Velmi komplexní součástí je zadávání místa konání. Adresu lze totiž zadat několika způsoby. Prvním způsobem je zadání souřadnic z map od firem Google nebo Seznam.cz. K tomu slouží tlačítko *„Vybrat místo na mapě“*. Po kliknutí na dané tlačítko se otevře modální okno, ve kterém bude zobrazena buď mapa od firmy Google pomocí *Google Maps API*[2], nebo mapa od firmy Seznam.cz mapa pomocí *API Mapy.cz* [24]. Na zobrazené mapě je možné vybrat místo konání, jehož souřadnice lze následně vložit do příslušného textového pole.

Důvodem, proč lze vybrat souřadnice místa pomocí dvou různých API, je ten, že mapa od jedné firmy nemusí být na některých místech vždy přesná. Když uživatel hledá konkrétní místo, tak se může stát, že na jedné z těchto map nebude daný bod evidován stejně přesně jako na druhé mapě. Souřadnice se pak mohou mírně lišit a v momentě, kdy si někdo bude prohlížet, kde se daná akce koná, může být vyznačena nepřesná lokace. Uživatel, který danou lokalitu nezná, se pak bude snažit jet na místo, které však nenajde nebo bude zdánlivě jinde než ve skutečnosti. Pro snížení této nepřesnosti je přidána druhá mapa, kdy lze ověřit, že se jednotlivá místa opravdu nachází na daných souřadnicích. Jestliže uživatel uvidí, že na jedné mapě není možné dané místo přesně zobrazit, může zkusit druhou mapu. Také se může stát, že po zadání souřadnic do navigace budou souřadnice z jedné mapy směřovat na místo odlišné od správné lokace, nebo může navigovat jinou cestou, která ve skutečnosti není průjezdná, což může zmást uživatele neznalé v dané územní oblasti. K tomu rovněž slouží ověření v podobě souřadnic druhé mapy.

I přesto, že jsou zadány souřadnice, nastavení konkrétní adresy je stále povinné, aby se mohla tato adresa zobrazit v detailu akce a případní zájemci přibližně věděli, kde se daná akce koná. Tlačítko *„Nastavit adresu dle souřadnic“* umožňuje zjistit adresu podle souřadnic. Pokud se adresu podaří zjistit, budou zjištěné údaje vypsány do příslušných polí. Tato technologie se nazývá **reverzní geokódování**. Současný návrh rozhraní neumožňuje vybrat souřadnice (Google nebo Seznam), podle kterých by se měla zjistit adresa.

Obdobně funguje možnost *„Nastavit souřadnice dle adresy“*, která se pokusí najít souřadnice na mapě pomocí vyplněné adresy. Tato technologie se nazývá **geokódování**. Je také možné souřadnice vynechat úplně, přičemž se pak v detailu akce nebude zobrazovat mapa upřesňující dané místo a uživatelé se budou muset spolehnout na dodanou adresu.

[2] https://developers.google.com/maps/documentation/javascript

Vybírání moderátorů a pořadatelů

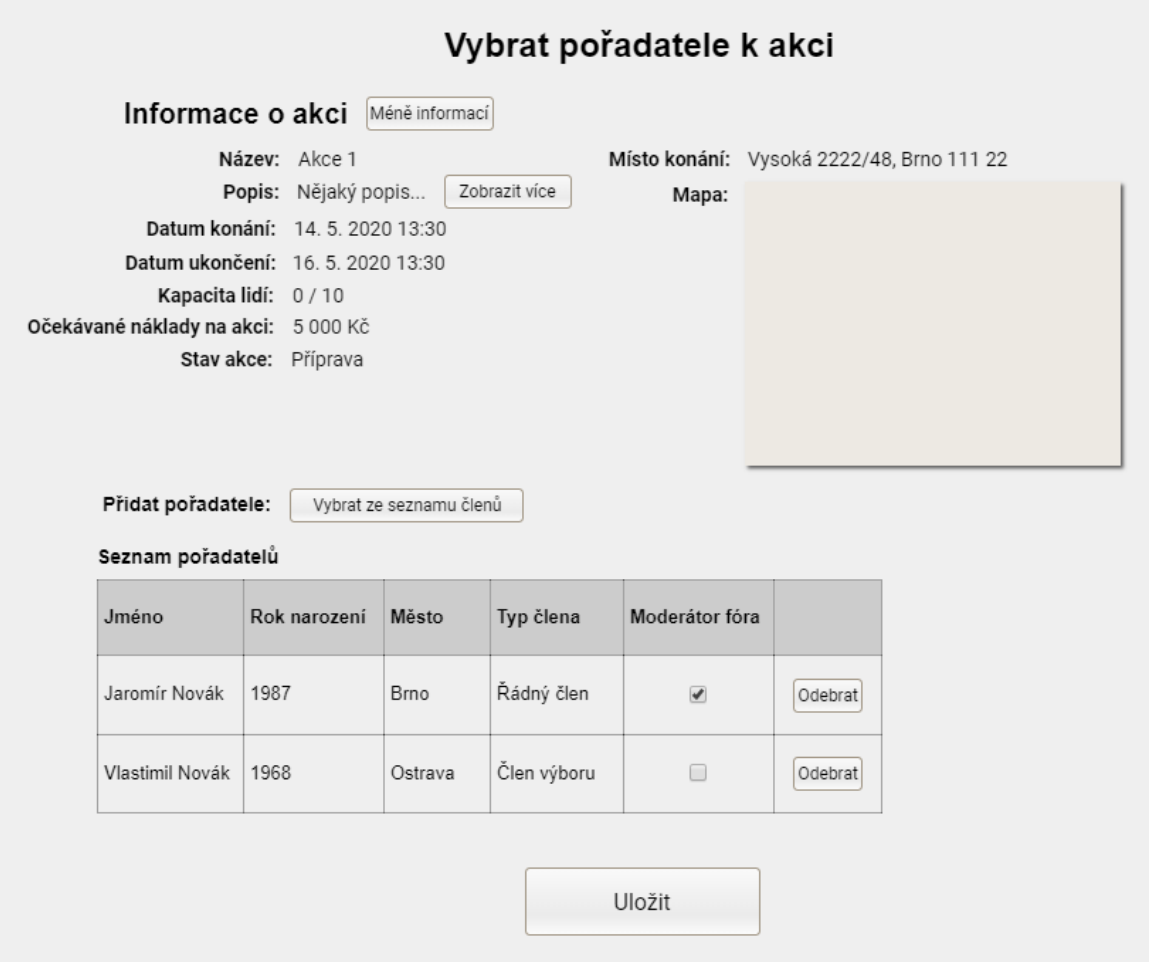
Na obrázku 5.4 je znázorněn pohled pro výběr pořadatelů. Po kliknutí na tlačítko *„Vybrat ze seznamu členů“* se zobrazí okno se seznamem členů, které lze vybrat jako pořadatele akce. Pořadatelem akce může být uživatel, který má platné členství. Jestliže organizátor akce není členem výboru, zobrazí se v seznamu pouze členové výboru a členové, kteří mají v profilu zatrženo, že chtějí být zobrazováni v seznamu pro výběr pořadatelů (příznak *show_as_organizator* v entitní množině *user*) – tedy že se nabízejí jako dobrovolníci k pořádání akcí. Jedná se o zabezpečení údajů členů, aby obyčejný člen neměl přístup k osobním údajům ostatních členů mimo členy výboru, neboť členové výboru jsou uvedeni veřejně na webu.

Kdy je možné vybírat moderátory a jaká jsou omezení na základě stavu akce shrnuje tabulka 5.2.

Typ člena	**Fórum povoleno**	**Příprava, Schvalování a Neschválená**	**Předběžná rezervace**
Pořadatel akce	Ne	Může vybírat i rušit výběr	Moderování je vypnuto
Schvalovatel	Ne	Může vybírat i rušit výběr	Může vybírat i rušit výběr
Pořadatel akce	Ano	Může vybírat i rušit výběr	Může pouze rušit výběr[3]
Schvalovatel	Ano	Může vybírat i rušit výběr	Může vybírat i rušit výběr

Tabulka 5.2: Vybírání moderátorů na základě stavu akce a typu člena

[3]Nově vybrané pořadatele nelze zvolit jako moderátory.



Obrázek 5.4: Návrh pohledu na výběr pořadatelů a moderátorů

Stránka pro schválení akce vychází z pohledu na obrázku 5.4. Pouze nelze vybírat nové pořadatele (je možné odebrat pořadatele) a rovněž není možné určovat nové moderátory (vybrané moderátory lze zrušit). Dále je na stránce pro schválení akce přidán formulář pro povolení či zamítnutí fóra a nastavení režimu fóra.

Pokud by chtěl člen výboru (schvalovatel) přidávat pořadatele a moderátory, může toto provést v rámci úpravy dané akce. Důvodem, proč to není nabízeno v rámci schvalování akce, je, že zatímco odebrání je považováno za nesouhlas s daným pořadatelem či moderátorem v rámci schvalovacího procesu, ke kterému může běžně docházet, přidání je neobvyklá operace, protože do týmu organizátorovi akce (který za akci bude zodpovědný) přidává další osobu, se kterou bude muset spolupracovat. Je proto vhodné, aby takováto neobvyklá operace nebyla ihned dostupná v rámci rutinního úkonu schvalování akce.

Přihlašování na položky programu

Na program akce se lze přihlásit buď jako pouhý účastník, nebo jako dobrovolník, který se zavazuje dodat určité prostředky. Pro přihlášení jako účastník je nutné zaškrtnout vybrané zaškrtávací políčko, jak znázorňuje pohled na obrázku 5.5. Pokud se chce člen přihlásit jako dobrovolník, musí kliknout na tlačítko *„Přihlásit jako dobrovolník“*. Po kliknutí na tohle tlačítko se objeví modální okno s detailem aktivity a seznamem požadovaných prostředků, jak znázorňuje obrázek 5.6. Po uložení programu se pak pod každým zaškrtávacím políčkem zobrazí, zda se uživatel přihlásil jako dobrovolník či jako účastník. Pokud se člen přihlásil jako dobrovolník a chce se odhlásit, stačí odškrtnout dané zaškrtávací políčko a uložit program.

Sloupec *„Organizace“* označuje, jak je aktivita organizována. Možnosti jsou následující:

- *Zajištěná organizátorem* – Aktivita je zajištěná v rámci akce, není potřeba dobrovolníků.
- *Zajištěná dobrovolníkem* – Aktivita vyžadovala účast dobrovolníků, přičemž bylo splněno dodání požadovaných prostředků.
- *„Hledáme dobrovolníky“* – Aktivita není zajištěná v rámci akce, ale vyžaduje účast dobrovolníků. Požadované prostředky zatím nebyly dodány a hledají se zájemci.

Každá aktivita navíc může být různého druhu:

- *Povinná* – Tento typ aktivity nemůže být zvolen pro dobrovolnou aktivitu. Jedná se o povinnou aktivitu, které se musí zúčastnit všichni. V seznamu pro přihlášení není pro tuto aktivitu zobrazena možnost přihlášení či odhlášení.
- *Společná s přihlášením* – Zájemci o tuto aktivitu se musí přihlásit.
- *Společná pro všechny* – Jedná se o nepovinnou aktivitu, na kterou však není nutné se přihlašovat. Na akci je zaručeno, že bude dostupná a kdo bude chtít, může se zúčastnit (např. běh).

Přihlašování na položky programu

Informace o akci

Méně informací

Název: Akce 1
Popis: Nějaký popis... Zobrazit více
Datum konání: 14. 5. 2020 13:30
Datum ukončení: 16. 5. 2020 13:30
Stav akce: Příprava
Cena vstupného: 1 100 Kč
Datum rezervace: 1. 5. 2020 18:22
Záloha zaplacena: Ano (500 Kč)
Stav kreditů: 2 500 Kč

Místo konání: Vysoká 2222/48, Brno 111 22
Mapa:

Program akce

Začátek aktivity	Konec aktivity	Název	Organizátor	Cena [Kč]	Záloha [Kč]	Organizace	Detail	Přihlášení
Pondělí 14. 5. 8:00	Středa 16. 5. 10:00	Aktivita 2	Vlastimil Novák	-	-	Zajištěná dobrovolníkem (4/10 zájemců) Přihlášen: Dodám výpomoc	Detail	☑ Přihlášen jako dobrovolník
Pondělí 14. 5. 13:30	Úterý 15. 5. 13:30	Aktivita 1	Vlastimil Novák	300	150 Zaplaceno	Zajištěná dobrovolníkem (4/6 zájemců)	Detail	☑ Přihlášen jako účastník
Pondělí 14. 5. 13:30	Středa 16. 5. 8:00	Ubytování	Petr Novák	1 000	300 Zaplatit	Zajištěná organizátorem	Detail	☑ Přihlášen jako účastník
Pondělí 14. 5. 13:30	Středa 16. 5. 13:30	Plná penze	Petr Novák	300	150	Zajištěná organizátorem (30/50 zájemců)	Detail	☐
Pondělí 14. 5. 18:00	-	Běh	Petr Novák	-	-	Zajištěná organizátorem	Detail	Společné pro všechny
Úterý 15. 5. 18:00	20:00	Šipky	Jan Novák	-	-	Hledáme dobrovolníky (0/10) Přihlásit jako dobrovolník	Detail	Přihlášení po zajištění organizace
Úterý 15. 5. 19:00	21:00	Golf	Jan Novák	-	-	Hledáme dobrovolníky (7/10) Přihlášen: Dodám golfové tyče	Detail	☑ Přihlášen jako dobrovolník

Rekapitulace

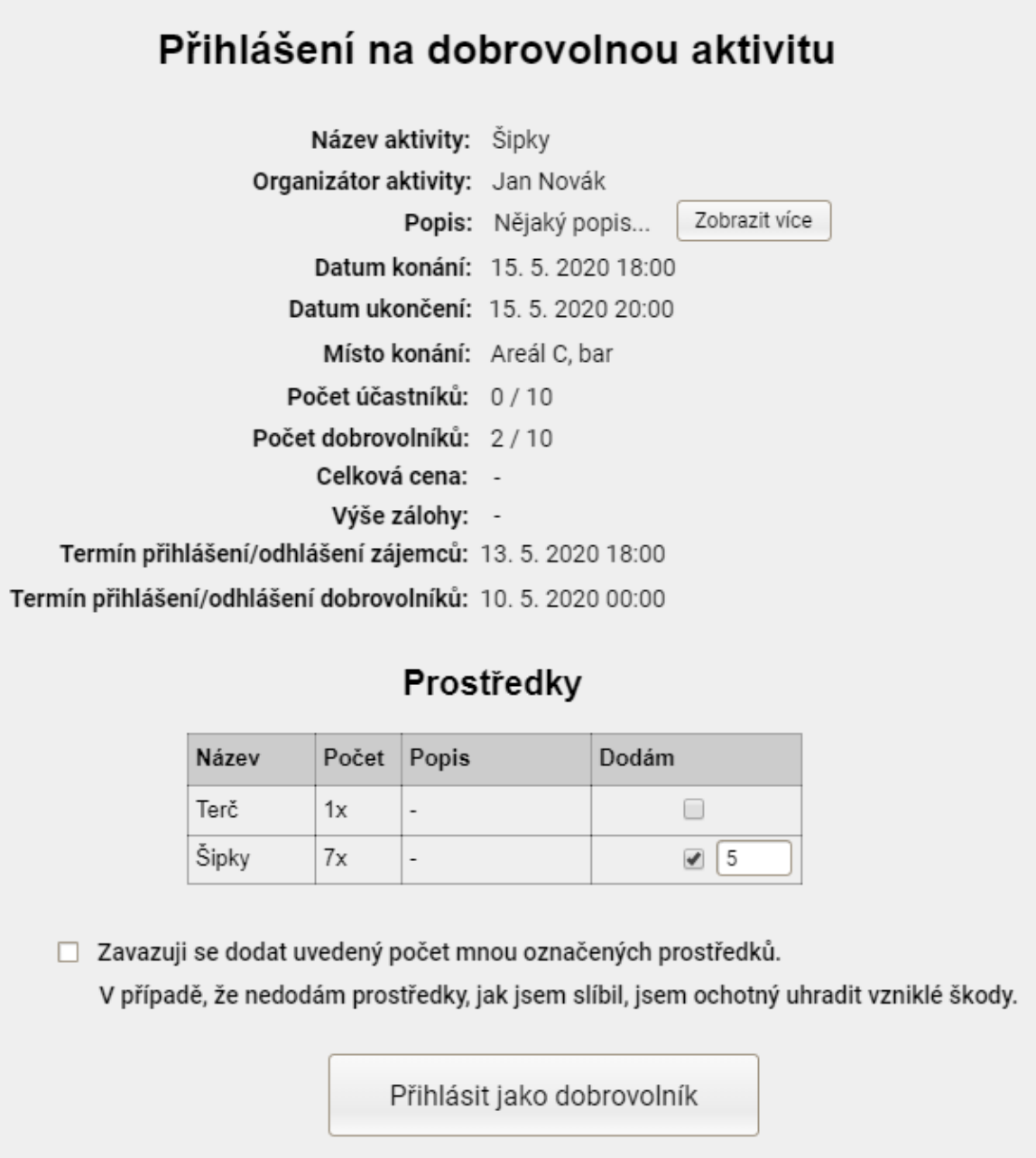
Počet přihlášených aktivit: 4 (z toho 2 jako dobrovolník)
Celková cena za přihlášené aktivity: 1 300 Kč
Celková výše zálohy za přihlášené aktivity: 450 Kč

Dodám prostředky: Golf - golfové tyče
Aktivita 2 - výpomoc

Uložit svůj program

Obrázek 5.5: Návrh pohledu pro přihlašování na aktivity

Jak lze vidět z obrázku 5.6, dobrovolník přihlášený na aktivitu se může odhlásit pouze do určeného termínu. Pokud tak neučiní a není schopen dodat prostředky, které se zavázal poskytnout, může ještě poslat zprávu organizátorovi aktivity nebo akce a vyhnout se tak postihu (vyloučení ze spolku, strhnutí kreditu, špatná reputace v poznámkách o uživatelích – viz tabulka `note` z ER diagramu).



Obrázek 5.6: Návrh pohledu pro přihlášení dobrovolníka

Pokud člen přidá otevřenou/dobrovolnou aktivitu, může ji odstranit pouze dokud nemá žádné přihlášené zájemce. Jestliže člen, který přidal otevřenou aktivitu, není schopen tuto aktivitu zajistit, může poslat zprávu organizátorovi akce a ten může aktivitu odstranit s tím,

že pošle zprávu o zrušení aktivity všem zájemcům a rovněž se přidá informace o zrušení této aktivity na hlavní stránku.

Přihlášení účastníci na aktivitu se ukládají do tabulky `activity_participants`. Zda se jedná o dobrovolníka rozlišuje příznak `is_volunteer`. Dobrovolníci, co poskytují různé prostředky, jsou však zároveň uloženi v tabulce `resource_suppliers`, která slouží k zaznamenání, kdo dodá jaké prostředky a případný počet daného prostředku. Samostatná vazební tabulka pro prostředky existuje proto, že v tabulce účastníků se eviduje údaj, kdy se daný člen na aktivitu přihlásil a také, zda se jedná o dobrovolníka či nikoliv. V případě, že by v této tabulce byla navíc informace o prostředcích (ID a počet), došlo by k duplicitě uvedených dat (datum přihlášení), neboť dobrovolník může poskytnout více prostředků než jen jeden (vztah M:N).

Lze si povšimnout, že v tabulce `resource_suppliers` není údaj o tom, k jaké aktivitě dobrovolník tyto prostředky poskytuje. Není to nutné, poněvadž ID aktivity lze zjistit z tabulky `resource`, na kterou vazební tabulka odkazuje, nebo také z tabulky `activity_participants` porovnáním uživatelského ID.

Za prostředek se považuje i výpomoc na aktivitě, přičemž se nejedná o hmotný předmět. V takovém případě pak dobrovolník nezadává počet prostředků, které dodá, ale dá se říct, že „poskytne sám sebe“.

Kapitola 6

Implementace

Rezervační systém pro neziskové organizace, který používá framework Nette pro snadnější implementaci, využívá architekturu MVP (Model-View-Presenter), což je obdobou MVC (Model-View-Controller). Tato architektura je specifická primárně pro framework Nette. Více informací se lze dočíst z literatury [14].

6.1 Rozložení projektu

Projekt je rozdělen tak, aby v maximální možné míře dodržoval zásady čistého kódu a zároveň aby byla oddělená vrstva dat od vrstvy aplikační logiky. Model aplikace, který zahrnuje aplikační logiku a databázovou vrstvu, je tedy rozdělen na repozitáře, fasády a služby.

Repozitáře jsou třídy, které se dotazují databáze MySQL a vrací výsledek dotazu. Fasády provádí aplikační logiku nad daty, která si získají použitím repozitářů předaných metodou *Dependency Injection* [15]. Některé fasády mohou volat služby, což jsou z hlediska tohoto projektu podpůrné třídy zajišťující určitou aplikační logiku, kterou lze provést nezávisle na fasádách. V tomto projektu se jedná zejména o službu automatického stahování bankovních výpisů. Více informací o návrhových vzorech v PHP je možné se dočíst z literatury [4].

Nakonec jsou zde presentery, které volají fasády na základě vstupu od uživatele. Vstup od uživatele je validován automaticky formuláři ve frameworku Nette. Zvalidovaný vstup je pak předán fasádě, která může provést dodatečnou validaci. Presentery jsou zodpovědné za ověřování práv uživatelů a zobrazování příslušných stránek naplněných požadovanými daty.

Velkým problémem při implementaci může být předávání dat z databáze do samotného pohledu. Existuje mnoho různých řešení, například mít doménový model pro všechny tabulky z databáze a data předávat pomocí business modelů, přičemž jednotlivé modely se mezi sebou mapují. Tento způsob umožňuje například framework Doctrine. Lze tím získat objektově orientovaný přístup za cenu výkonu, což může být u některých systémů značnou nevýhodou. Zvolil jsem proto hybridní přístup pomocí objektů **DTO** (Data Transfer Object). Tyto objekty neodrážejí databázové entity přesně, jak jsou v databázi, nýbrž se některé vlastnosti mohou lišit či dokonce nemusí vůbec existovat jako atribut v databázové entitě. Tyto objekty jsou užitečné pro přenos po síti, jak samotný název vypovídá. Jedná se tedy o objekt, který obsahuje data, jež se mají na stránce zobrazit.

V tomto projektu však nevyužívám DTO pro každou stránku, avšak volím kombinovaný přístup. Kde je to vhodné, využiji DTO, kde by to znamenalo zbytečné psaní kódu navíc, využiji klasický přístup využívaný v PHP, kdy se data přenáší v poli nebo v generalizovaném objektu (`object` nebo `stdClass`) [28].

6.2 Import bankovních výpisů pomocí bankovního API

Jak již bylo zmíněno, spolek ČMFN využívá transparentní bankovní účet zřízený u společnosti **Fio banka, a.s.** Tato banka nabízí vlastní API pro sledování všech plateb. Je k tomu však potřeba unikátní token, který mi poskytli členové výboru ze spolku ČMFN.

Popis bankovního klienta v systému

Stahovat bankovní výpisy lze zasláním požadavku typu `GET` protokolem `HTTP` na URL tohoto API z konfigurace systému. Pro zaslání takového dotazu používám volně dostupnou knihovnu **cURL** pro PHP[1]. Přesný formát dotazu je stanoven v dokumentaci bankovního API [11].

Jelikož dotazů může být více a každý má svou vlastní funkcionalitu na straně API, rozdělil jsem tyto dotazy na příkazy pomocí tříd jazyka PHP. Tyto třídy dědí z jednoho společného rozhraní, takže rozhraní klienta, který se stará o zasílání správných dotazů, nemusí znát přesnou definici příkazu.

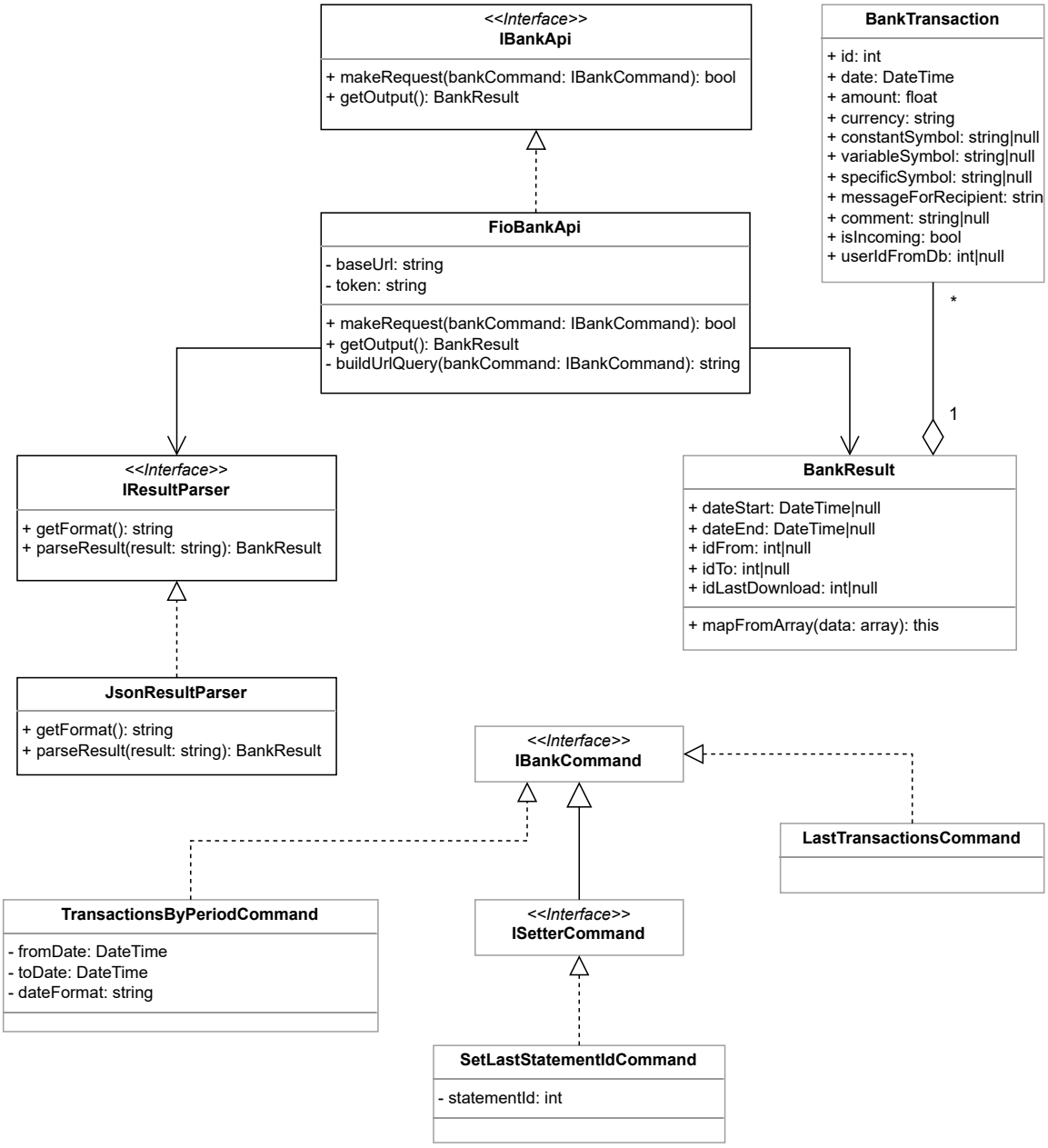
V budoucnu je možnost, že by si spolek ČMFN zřídil bankovní účet u jiné banky, která může mít odlišný způsob získávání potřebných dat. Proto jsem vytvořil rozhraní v jazyce PHP, které definuje metody, jenž by měla konkrétní implementace klienta umět. Tento způsob je vhodný z hlediska rozšiřitelnosti a jednoduché změny klienta v případě potřeby.

Data se z bankovního API stahují v určitém formátu. Fio bankovní API podporuje mnoho formátů pro import bankovních dat. Zvolil jsem formát JSON, jelikož tento formát je jednoduchý na parsování pomocí vestavěné funkce PHP `json_decode()`[2].

Uvažoval jsem však i případ, kdy by se mohla změnit implementace klienta, která by stahovala bankovní výpisy v jiném formátu. Proto jsem opět použil rozhraní, které slouží jako šablona dostupných metod, ale implementaci těchto metod zařizuje konkrétní třída. Aby byl výsledek přenositelný do různých implementací, vytvořil jsem třídu `BankResult`, která definuje položky, jež by se měly z bankovního API získat. Každá implementace rozhraní pro parsování dat tedy musí vracet instanci této třídy.

Výše uvedený postup znázorňuje diagram tříd na obrázku 6.1. Pro více informací o diagramu tříd jazyka UML včetně syntaxe a použití se lze dočíst z literatury [6] a [7].

[1] https://www.php.net/manual/en/book.curl.php
[2] https://www.php.net/manual/en/function.json-decode.php



Obrázek 6.1: Diagram tříd pro bankovního klienta v systému

Stažení bankovních transakcí a vložení do databáze

Potom, co je získán výsledek v přijatelné podobě, je na řadě zpracování těchto dat a jejich vložení do databázové tabulky `payment`. Volání metod bankovního klienta i zpracování a vložení plateb zařizuje třída `PaymentService`.

Obsahuje metodu, která se nejdříve podívá do tabulky `payment_import` na poslední stažený výpis. Pokud žádný není a tudíž se jedná o první import dat, pošle skrz bankovního klienta příkaz `TransactionsByPeriodCommand`, který je znázorněn v diagramu na obrázku 6.1. Jedná se tedy o příkaz, který vyžaduje období, za které se mají stáhnout

bankovní transakce. Počáteční datum je vybráno z konfigurace, zatímco koncové datum je nastaveno na současné. Poté se pošle příkaz.

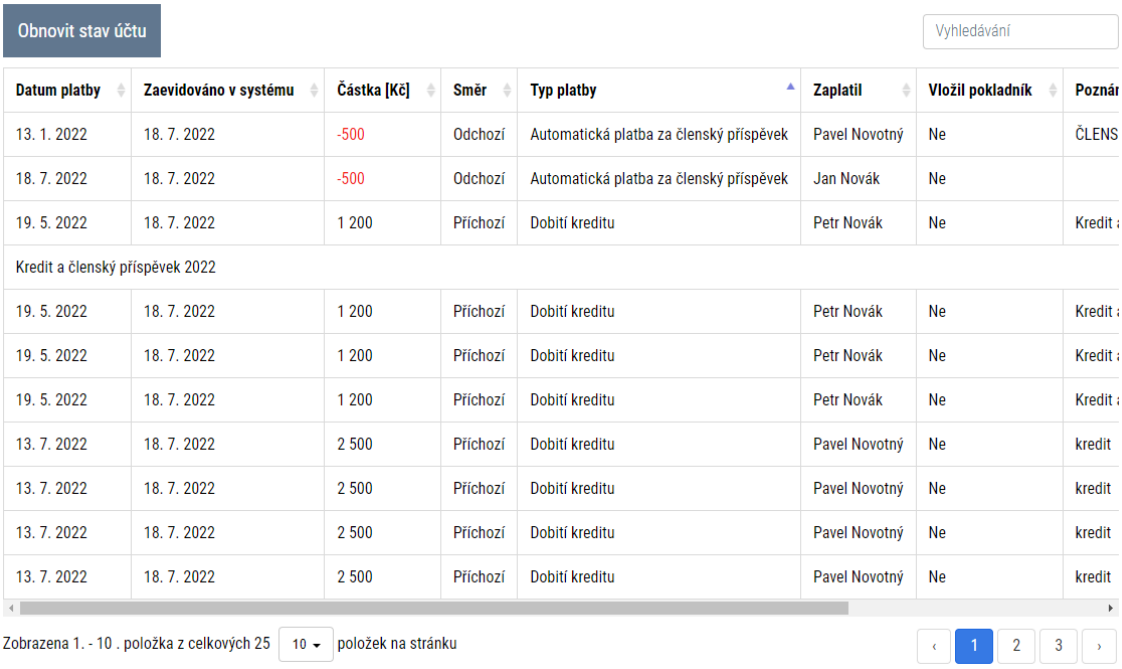
Pokud se však číslo posledního importovaného výpisu v tabulce najde, provede se příkaz `LastTransactionsCommand`, který stáhne výpisy od posledního stažení. Rovněž se zkontroluje, zda je dodržován globální interval dotazování pro případ, že se jedná o obnovu účtu iniciovanou uživatelem.

Následující operace musí být obaleny v transakčním bloku. Při importu plateb do databáze se nesmí stát, že by se některá platba ztratila. To znamená, že se používají databázové transakce pro zajištění, že pokud dojde k chybě, resetují se všechny provedené změny. Framework Nette při zjištění chyby vyhazuje výjimku, kterou lze odchytit. To je výhodné, jelikož nestačí jen resetovat změny v databázi. V případě, že dojde k chybě a platby nejsou úspěšně importovány do systému, musí se navíc nastavit zarážka na straně bankovního API, která určuje poslední úspěšně stažený výpis z banky. Tato zarážka slouží k tomu, aby se příkazem `LastTransactionsCommand` stahovaly transakce jen od této zarážky. Jedná se o velmi dobrou vlastnost daného bankovního API, jelikož se nemusí komplikovaně kontrolovat, zda už daná platba není v systému evidována.

Zarážka se nastavuje v bloku `catch` příkazem `SetLastStatementIdCommand` na poslední číslo úspěšně staženého výpisu, které je dostupné v tabulce `payment_import`.

Po vložení všech plateb se aktualizují kredity všech uživatelů (atribut `credit` z tabulky `user`) [20] a navíc se odečtou kredity za členské příspěvky, pokud se import spustil v době, kdy je nastaveno automatické strhávání příspěvků. Strhávání členského příspěvku je rovněž obaleno databázovými transakcemi. Stále se jedná o kritickou operaci.

Na obrázku 6.2 lze vidět ukázku seznamu importovaných plateb do databáze.



Obrázek 6.2: Seznam plateb po importu z bankovního API

Zpracování stažených dat

V předešlé sekci bylo popsáno, co vše je nutné udělat pro stažení a vložení plateb. Nebylo však zmíněno, jak se ze stažených dat pozná, o co se jedná.

Tohle zajišťuje privátní metoda `getMemberPayments()` třídy `PaymentService`. V zásadě projde všechny přijaté transakce v objektu `BankResult`. V diagramu 6.1 si lze povšimnout, že třída `BankTransaction` obsahuje položku `isIncoming`. Tento příznak určuje, zda se jedná o příjem či výdej v rámci bankovního účtu (nikoliv v rámci tabulky `payment`). Hodnota příznaku se zjišťuje velmi jednoduše: o **příjem** se jedná, když je částka **kladná**, naopak o **výdej** se jedná, když je částka **záporná**.

V rámci evidence uživatelských plateb se neukládají výdaje spolku. Proto se v první fázi zpracování bankovních transakcí odfiltrují všechny platby, jež nejsou příjmem. Jak již bylo zmíněno v popisu spolku, většina příjmů se týká členských příspěvků. Pro odlišení členských příspěvků od zbytku plateb se používá předčíslí **910** následované identifikátorem uživatele z databáze (atribut `id` z tabulky `user`) ve variabilním symbolu platby.

Algoritmus tedy zkontroluje všechny variabilní symboly, zda obsahují dané předčíslí. Pokud nelze najít definované předčíslí ve variabilním symbolu platby, regulárním výrazem se otestuje, zda náhodou není tohle předčíslí ve zprávě pro příjemce. Provádí se to proto, poněvadž někteří uživatelé, co platili za členský příspěvek, uvedli svůj variabilní symbol do zprávy pro příjemce místo do odpovídajícího variabilního symbolu. Jestliže se nenajde předčíslí ani ve zprávě pro příjemce, přidá se bankovní transakce do seznamu problémových plateb.

Problémové platby jsou příchozí platby, které systém neumí spárovat s uživatelem. Tyto platby jsou proto automaticky poslány pokladníkovi na jeho e-mailovou adresu, aby je zkontroloval a případně ručně vložil do systému.

Všechny nalezené bankovní transakce se stanoveným předčíslím se rovněž ukládají do seznamu, aby se pak pomocí jednoho dotazu SQL zjistilo, kteří uživatelé v systému opravdu existují a které identifikátory získané z platby jsou nevalidní. Platby s neexistujícím ID uživatele jsou rovněž přidány do seznamu problémových plateb.

Platby, které mají ID uživatele shodné s tím z databáze, jsou nakonec metodou vráceny a lze je adekvátně vložit do databáze. Tím je proces zpracování bankovních transakcí dokončen.

6.3 Akce

Akce byly z hlediska implementace zajímavé zejména tím, že se muselo řešit více větších celků. Například pro vytvoření akce bylo nutné zajistit možnost zvolení časového pásma, aby bylo založení akce možné kdekoliv po světě. Dalším zajímavým problémem bylo řazení akcí.

Přihlašování na akci bylo implementováno dle popsaného návrhu a analýzy požadavků. Prostředky týkající se dobrovolných aktivit zatím nebyly z časových důvodů implementovány, jelikož se zároveň jedná o rozšíření oproti původnímu zadání.

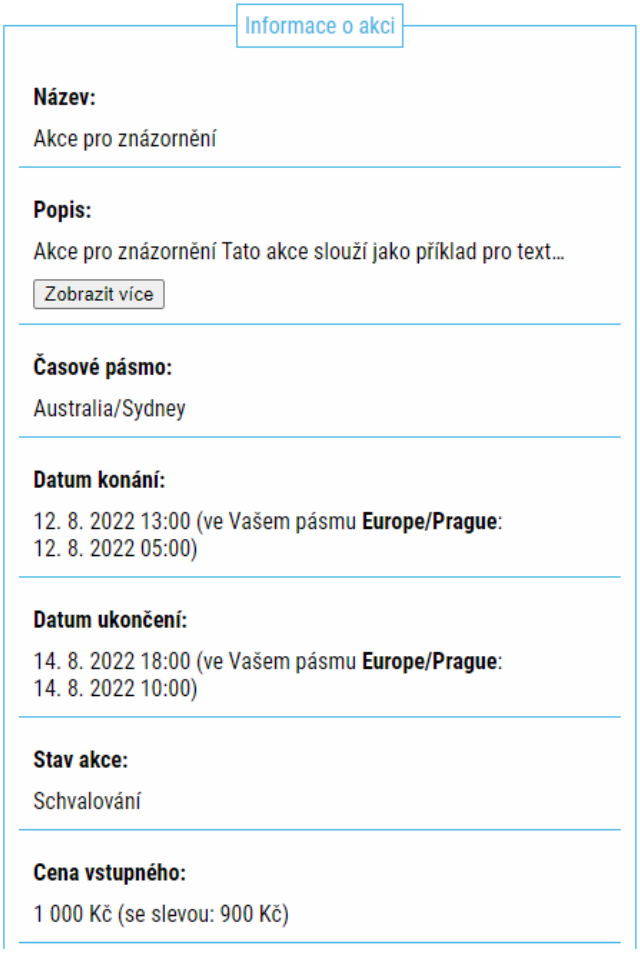
Zobrazení časových pásem

Jak již bylo popsáno v podkapitole 5.3, časy v databázi jsou uloženy v jiném časovém pásmu než v jakém časovém pásmu akce skutečně je. Zároveň i klient může být v jiné časové zóně, než je samotná akce.

Implementace řešení spočívá v zaslání klientského časového pásma pomocí *cookies* obsažených v požadavku protokolem HTTP. Jelikož se jedná o klientskou stranu, musí se tyto cookies nastavit pomocí JavaScriptu. K tomu jsem využil již vytvořené funkce [3]. Po nastavení cookies a odeslání požadavku protokolem HTTP na server se přepočítají veškeré časy získané z databáze tak, aby jejich časové pásmo odpovídalo časovému pásmu klienta. Tyto převedené časy se poté zobrazí klientovi na stránce.

Problém nastává u akcí, které mohou být v jiném časovém pásmu, než v jakém je klient. Už nestačí převést časy jen do klientského časového pásma, nýbrž je nutné zobrazit i správný čas na daném místě. Původní řešení spočívalo v zobrazení popisu klientského času po najetí myši na daný čas. Tohle řešení se však neosvědčilo v praxi, kdy na web přistupovalo mnoho uživatelů z mobilního zařízení, které tyto popisy nezobrazí. Použilo se tedy jiné řešení.

Po najetí myši na čas se stále zobrazí čas v klientském časovém pásmu, ale zároveň bylo přidáno tlačítko na zobrazení nebo skrytí klientského času. Tento popis i tlačítko se uživateli zobrazí jen tehdy, kdy je jeho časová zóna odlišná od časové zóny akce. Příklad mobilního zobrazení detailu akce, která je v jiném časovém pásmu než klient, lze vidět na obrázku 6.3.

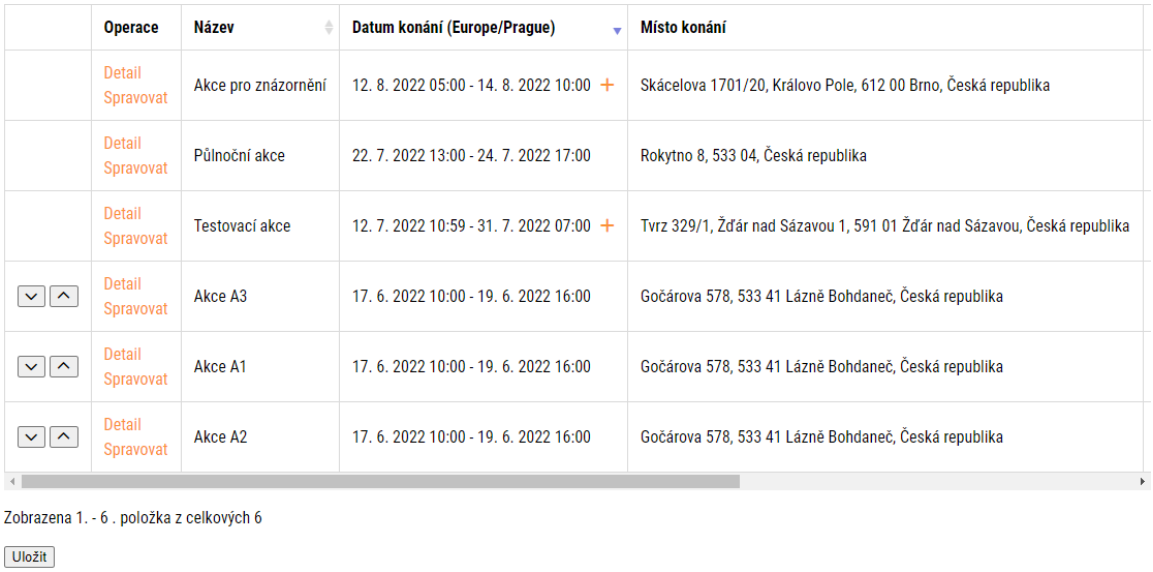


Obrázek 6.3: Zobrazení detailu akce na mobilním zařízení

Řazení akcí

Podle specifikace by se měly akce řadit nejdříve podle data konání a poté podle data ukončení. Jakmile však existují akce, které mají stejné datum konání i datum ukončení, musí se zavést jiný způsob řazení.

K tomu slouží položka `order` z tabulky `event`. Tento atribut uvádí pořadí akce pomocí čísla. Aplikuje se pouze tehdy, kdy nelze akce seřadit podle data konání nebo ukončení. Je však nutné zjistit, která akce má mít vyšší pořadí oproti jiné akci. Jednotlivé pořadí akcí určuje člen, jenž k tomu má příslušné právo. Ukázka možného řazení akcí je znázorněna na obrázku 6.4.



	Operace	Název	Datum konání (Europe/Prague)	Místo konání
	Detail Spravovat	Akce pro znázornění	12. 8. 2022 05:00 - 14. 8. 2022 10:00 +	Skácelova 1701/20, Královo Pole, 612 00 Brno, Česká republika
	Detail Spravovat	Půlnoční akce	22. 7. 2022 13:00 - 24. 7. 2022 17:00	Rokytno 8, 533 04, Česká republika
	Detail Spravovat	Testovací akce	12. 7. 2022 10:59 - 31. 7. 2022 07:00 +	Tvrz 329/1, Žďár nad Sázavou 1, 591 01 Žďár nad Sázavou, Česká republika
	Detail Spravovat	Akce A3	17. 6. 2022 10:00 - 19. 6. 2022 16:00	Gočárova 578, 533 41 Lázně Bohdaneč, Česká republika
	Detail Spravovat	Akce A1	17. 6. 2022 10:00 - 19. 6. 2022 16:00	Gočárova 578, 533 41 Lázně Bohdaneč, Česká republika
	Detail Spravovat	Akce A2	17. 6. 2022 10:00 - 19. 6. 2022 16:00	Gočárova 578, 533 41 Lázně Bohdaneč, Česká republika

Obrázek 6.4: Seznam akcí s možností řazení

Lze si povšimnout šipek u akcí se stejným datem konání. Jak bylo zmíněno v sekci výše, každá akce může být v jiném časovém pásmu. Takže například pokud by existovala akce s datem konání *17. 6. 2022 10:00* v časovém pásmu *Europe/Prague* a dále akce s datem konání *17. 6. 2022 10:00* v časovém pásmu *Australia/Sydney*, řazení podle pořadí by zde nebylo možné, jelikož po převodu do stejného časového pásma by se oba časy lišily (např. po převodu do *Europe/Prague* by měla druhá zmíněná akce datum konání *17. 6. 2022 02:00*). Zde se tedy aplikuje řazení podle data konání, případně data ukončení.

Ovšem u akcí se stejným časovým pásmem i datem konání by se již možnost řazení objevila, jak je možno vidět na obrázku 6.4 u akcí „Akce A3“, „Akce A1“ a „Akce A2“. Dále je nutné brát v potaz, že můžou existovat různé skupiny akcí, které mají stejné datum konání (v závislosti na časovém pásmu). Pro lepší porozumění lze uvést příklad dvou akcí, které se konají ve stejný den a čas a pak dalších dvou akcí, které se sice konají v jiný den a čas než předešlé dvě akce, ale současně by měly tyto dvě akce rovněž stejné datum konání. Vznikají tedy skupiny akcí na základě stejného data konání.

Měnit pořadí akce lze pouze v dané skupině. Akce se vkládají do skupin tak, že se vypočítá hash pomocí algoritmu **SHA256** z hodnot data konání (`start_date`), data ukončení (`end_date`) a časového pásma (`timezone`), které jsou získány rovnou z databáze bez převodu do jiného časového pásma. Pro lepší ilustraci může posloužit následující tabulka 6.1.

Datum konání	**Datum ukončení**	**Časové pásmo**	**Spojený řetězec**	**Výsledný hash**
2022-06-17 10:00:00	2022-06-19 16:00:00	Europe/Prague	2022-06-1710:00:002022-06-19 16:00:00Europe/Prague	45aefadec95a65566783f9544423d55f70fa2a928c4c3a327df474b08bb697fc
2022-06-17 10:00:00	2022-06-19 16:00:00	Australia/Sydney	2022-06-1710:00:002022-06-19 16:00:00Australia/Sydney	55e4c4270792aa1512080af2f156fc4f1666eccf02e59e0269a2982d001031cc

Tabulka 6.1: Vypočítání výsledného hashe pro skupiny akcí

Tento unikátní hash **SHA256** pro každou skupinu, který započítává i časové pásmo a tudíž nedojde ke kolizím v případě, kdy je stejné datum konání (viz tabulka výše), umožňuje dále v aplikaci zjistit, kam maximálně lze posunout danou akci.

Po kliknutí na příslušnou šipku se spustí událost v JavaScriptu, která zkontroluje, o jaký směr šipky se jedná a případně povolí posun odpovídajícího řádku tabulky o jeden řádek nahoru či dolů. Zároveň se změní pořadí akce, které je ukládáno v HTML pomocí atributů `data`. Skript, který se stará o posun akcí, si zároveň interně uchovává dva objekty: `oldEventOrderData` pro uložení starých čísel pořadí a `newEventOrderData` pro uložení nových čísel pořadí. Při každém posunu se zkontroluje, zda se nové pořadí dané akce rovná starému pořadí stejné akce a pokud ano, nastaví nové pořadí této akce na `undefined`, čímž signalizuje, že nedošlo k žádné změně oproti předchozímu stavu.

Tlačítko *„Uložit“*, které lze vidět na spodku tabulky, se objeví po změně pořadí. Po kliknutí na tohle tlačítko se odešle požadavek technologií *AJAX* pomocí knihovny *naja*, který na serveru uloží nové pořadí akcí. Nejdříve se však musí ověřit práva uživatele. Pokud má uživatel právo spravovat všechny akce, může měnit pořadí všem akcím. Jestliže uživatel takové právo nemá, ale má právo na změnu pořadí svých akcí, je proveden cyklus pro každou akci získanou ze zaslaného požadavku, přičemž se kontroluje, zda je uživatel organizátorem (pořadatelem) této akce. Za předpokladu, že se najde alespoň jedna taková akce, se změna uloží, jinak dojde k navrácení chyby uživateli.

Autorizace se provádí tímto způsobem, protože kdyby se ověřovala práva pro každou akci, téměř vždy by došlo k záporné odpovědi. Člen sice nemá práva na správu ostatních akcí, ale má právo na změnu pořadí své akce, čímž pak posouvá pořadí jiné akce. Změna pořadí jen v rámci svých akcí by nebyla moc k užitku a navíc by to bylo pracnější na implementaci.

6.4 Rodinní příslušníci

Jak již bylo zmíněno v podkapitole 3.3, lze odstranit rodinného příslušníka s ponecháním jeho údajů. To znamená, že taková osoba se poté může zaregistrovat se stejnými údaji, přičemž se tím obnoví předtím smazaný účet. V případe, kdy by se uživatel rozhodl údaje

rodinného příslušníka smazat, nahradí se všechny citlivé údaje v databázi zástupným textem „DELETED“.

Tenhle způsob registrace a odstraňování rodinných příslušníků byl přidán proto, jelikož smazaní uživatelé se nemažou fyzicky z databáze, ale pouze se jim nastaví datum smazání `deleted_date`. Pokud má tento atribut hodnotu `NULL`, uživatel smazán není.

Před smazáním jeho neregistrovaného účtu navíc mohla být uživateli poslána pozvánka do rodiny, takže by mohl využít své osobní údaje k aktivaci svého „smazaného“ účtu, který byl vytvořen jeho rodinným příslušníkem. Pozvánka je sice po smazání uživatele z rodiny neplatná, ale tento případ řeší komponenta starající se o registraci uživatelů (třída `RegistrationWizard`).

Registrace rodinného příslušníka

Registrační formulář pro nové uživatele je rozdělen na tři kroky. V prvním kroku zadávají své přihlašovací a osobní údaje. Pokud se přijde zaregistrovat uživatel, který zadá osobní údaje (křestní jméno, příjmení, datum narození a e-mailovou adresu), jež se shodují s již existujícími údaji v databázi a současně patří smazanému uživateli (`deleted_date` není `NULL`), vypíše se uživateli hláška, že se snaží aktivovat existující účet.

Takový uživatel má na výběr dvě možnosti. Buď si nechá zaslat ověřovací kód na danou e-mailovou adresu, nebo se zaregistruje pod jinou e-mailovou adresou. V případě, že zvolí jinou e-mailovou adresu, registrace pokračuje klasicky jako v případě registrace nového uživatele.

Pokud si však nechá zaslat kód, započne tím sezení v jazyce PHP, které bude průběžně monitorovat potřebné informace (např. počet pokusů). Kód je šestimístné náhodné číslo, které uživatel musí zadat do příslušného textového pole, aby tím ověřil svou identitu. Uživatel má tři pokusy na zadání správného kódu. Pokud vyčerpá všechny tři pokusy, ověřovací kód se zneplatní a uživatel si bude muset nechat zaslat nový kód. Doba platnosti daného kódu je vždy 20 minut.

Důvod, proč uživatel má tři pokusy, není jen ten, že se může uživatel zmýlit, ale rovněž se jedná také o to, aby náhodná osoba nezneužívala e-mailového klienta webového serveru ke spamování zadané e-mailové adresy – nový kód proto není možné zaslat před vypršením platnosti předchozího.

Jakmile uživatel zadá správný ověřovací kód, do sezení v PHP se uloží informace o tom, že uživatel byl ověřen. Následně, když uživatel pokročí do druhé fáze formuláře, se uživateli zobrazí vyplněná adresa, kterou předtím vyplnil jeho rodinný příslušník. Do sezení se uloží ID daného účtu, aby se po dokončení registrace nepřidal nový účet, ale naopak aktualizoval již existující. Od této fáze pokračuje registrace stejně jako u klasického uživatele.

Po odsouhlasení podmínek a kliknutí na „Registrovat“ se obvykle znovu kontroluje, zda nekoliduje uživatelské jméno a e-mailová adresa s některými v databázi. V tomhle případě však e-mailová adresa už v databázi existuje, a proto se kontroluje stav sezení v jazyce PHP. Pokud se jedná o přeregistraci (aktivaci, obnovu) existujícího účtu, e-mailová adresa se kontrolovat nebude a ověří se jen přihlašovací jméno.

Po aktualizací položek v databázi je uživatel přihlášen a má aktivní účet na rozdíl od nově zaregistrovaného uživatele, který si účet musí aktivovat pomocí speciálního odkazu zaslaného na jeho e-mailovou adresu. V tomto případě již není třeba znovu ověřovat účet, jelikož byl ověřen zadáním správného ověřovacího kódu na začátku registrace.

Kapitola 7

Testování

.

Tato kapitola se věnuje testování informačního systému. Velmi pečlivě bylo testováno zejména importování bankovních transakcí do databáze systému, jelikož se jedná o kritickou část systému. Dále bylo testováno vybírání pořadatelů a moderátorů, neboť tam je mnoho proměnných, které mohou narušit správný chod funkce. Během vývoje byl konzultován postup řešení pro rodinné příslušníky, protože možnosti, které nabízí, jsou náchylné na *„social engineering“*.

7.1 Testování importu plateb

Jelikož uživatelé předchozího systému nejsou importováni do databáze, kterou jsem používal při vývoji, nemohl jsem spárovat platby s uživateli (v mé lokální databázi takoví uživatelé neexistovali).

Nejdříve jsem testoval, zda správně funguje posílání příkazů na bankovní API a vrací správné bankovní výpisy za dané období. Bankovní účet spolku je transparentní, takže jsem si mohl zjistit první a poslední výpis v daném období a ten porovnat s výsledkem dotazu. Zde stačilo provést pár testů na příkazy pro stáhnutí plateb za určité období a pro stáhnutí posledních plateb od úspěšného stažení, neboť bankovní API již bylo otestováno samotnou Fio bankou.

Testovalo se tedy zejména to, zda se ukládají správné výpisy do tabulky `payment_import` a na základě toho se posílají správné příkazy, případně omezení obnovy dat z účtu globálním intervalem nastaveným v konfiguraci.

Poté jsem testoval zjišťování plateb pro vložení do databáze pomocí variabilního symbolu. Sepsal jsem si příchozí platby, které neměly uveden variabilní symbol, jelikož těch bylo pár jednotek. Poté jsem si vypsal výsledek seznamu plateb a porovnal, zda v seznamu nejsou platby se špatně uvedeným formátem a naopak, zda v seznamu jsou platby, které sice nemají variabilní symbol, ale ID uživatele je vloženo do zprávy pro příjemce.

Tuto část jsem testoval tak dlouho, dokud řešení pro všechny dostupné testovací případy inspirované reálnými daty z transparentního účtu spolku nefungovalo zcela správně.

Další testy již byly systematičtější. Jak bylo zmíněno výše, platby stažené z banky nešlo spárovat s testovacími uživateli v lokální databázi. Proto jsem si vytvořil základní testovací případy bankovních transakcí obsahující jen ty informace, které se v aplikaci skutečně používaly. Následně byly tyto uměle vytvořené bankovní transakce předány na

místo pro zpracování dat a jejich následné vložení do databáze. Do těchto bankovních transakcí byly vloženy existující identifikátory uživatelů z lokální databáze.

Takto jsem testoval po dobu vývoje zpracování dat a vkládání plateb do databáze. Zejména jsem testoval to, že se platby nevloží v případě chyby a že dojde k resetování zarážky na straně bankovního API.

7.2 Testování vybírání pořadatelů a moderátorů

V jazyce SQL jsem si vytvořil dva testovací uživatele. Jedním z nich byl obyčejný člen a druhým člen výboru. Otevřel jsem dvě paralelní instance prohlížeče a zkoušel, zda fungují určité případy na základě změn vlastností akce členem výboru.

Když test prošel a něco jsem v kódu upravil, původní akci jsem smazal a vytvořil novou předem připravenou akci pomocí SQL dotazu a celý proces opakoval. Průběžně jsem kontroloval výstup webové stránky s daty z databáze.

7.3 Rodinní příslušníci

Testování bylo podobné jako v případě pořadatelů a moderátorů. Nicméně se jednalo o větší počet uživatelů. U těchto testů jsem zkoušel zejména různé nebezpečné varianty, jak obejít připravená omezení.

Například jsem zjistil, že mi nefungovalo porovnávání UTF-8 znaků u opětovné aktivace smazaného uživatele (viz podkapitola 6.4). To způsobilo, že jsem mohl zadat jména bez diakritiky a podmínka na ověření, zda se jedná o správného uživatele, prošla.

Tato chyba se vyskytovala v klauzuli `WHERE` dotazu SQL při porovnávání jména z databáze se vstupním jménem uživatele. Opravil jsem ji tak, že jsem za výsledek dotazu navíc přidal podmínku v jazyce PHP, která jméno uživatele znovu porovnala. Tato oprava fungovala, jelikož porovnávání v jazyce PHP podporuje znaky UTF-8, a tedy i českou diakritiku.

7.4 Nasazení aplikace

Aplikaci jsem pro testování zkusil nasadit na zdarma dostupný hosting Endora.cz[1]. Během registrace a testování zasílání ověřovacích e-mailů jsem zjistil, že chybové hlášky, které se zobrazí v případě neúspěchu, jsou velmi neintuitivní. Také byl problém v pořadí, jelikož při neúspěšném zaslání e-mailu nebyla registrace dokončena.

Musel jsem tedy pozměnit pořadí prováděných akcí a zejména vylepšit odchytávání chyb e-mailového klienta. Nyní se výjimka odchytí, zapíše do souboru na serveru a uživateli je zobrazeno korektní hlášení. Nedochází při tom k přerušení jiných důležitějších operací (např. přidání uživatele do databáze).

Rovněž jsem požádal kolegu o výpomoc při testování. Testoval jsem, jak je pro něj prostředí přívětivé a jak rychle se v něm dokáže vyznat. Jednotlivé časy jsem si zapisoval. Po každé větší změně jsem jej opět požádal, zda by mi nepomohl s testováním. Byl jsem si vědom nepřesného měření, protože se jednalo o stejného kolegu, který si po určité době na systém zvykl a úkony prováděl rychleji bez menších chyb, ale naneštěstí jsem neměl k dispozici jiné testery.

[1] https://www.endora.cz/

Nicméně tohle testování mělo smysl alespoň v tom, že objevil spoustu uživatelských chyb, na které bych sám nemusel narazit (např. změna data narození v úpravě registrovaného rodinného příslušníka nad 18 let způsobila nesmyslnou chybovou hlášku).

7.5 Shrnutí výsledků a nalezených chyb

V tabulce 7.1 je shrnutí některých zaznamenaných chyb. Pod touto tabulkou následuje tabulka 7.2 s naměřenými časy pro jednotlivé úkony, jak popisovala podkapitola výše. Dané činnosti se testovaly nejvýše třikrát, jelikož pak už tester uměl systém používat a měření začalo být zkreslené.

Popis chyby	**Kategorie**	**Opraveno**	**Popis opravy**
Uživatel mohl přijmout pozvánku, když byl v rodině.	Pozvánky	Ano	Přidána podmínka na ID rodiny.
Po odchodu posledního zletilého rodinného příslušníka nebyla rodina rozpuštěna.	Rodina	Ano	Přidána podmínka pro věk rodinného příslušníka.
Po odchodu posledního rodinného příslušníka nebyla rodina zrušena.	Rodina	Ano	Přidána chybějící podmínka na začátek metody `leaveFamily()`.
Uživatel, který byl pozván jiným uživatelem v rodině a následně z této rodiny odešel a vytvořil si vlastní rodinu, byl po přijetí pozvánky přidán do nové rodiny uživatele.	Rodina	Ano	Chyběl atribut `family_id` v tabulce `family_invite`.
Po načtení úpravy akce v jiném časovém pásmu s velkým časovým posunem přestanou fungovat omezení JavaScriptové komponenty pro výběr data a času.	Akce	Ne	
Po uložení pořadatelů s vypnutým fórem byli všichni vybraní moderátoři zrušeni.	Akce	Ano	Pro vypnuté fórum se načtou příznaky moderátorů z databáze a ty se použijí. Nově přidaným pořadatelům je moderování automaticky zrušeno.

Tabulka 7.1: Nalezené chyby

Název úkonu	**Číslo pokusu**	**Naměřený čas**	**Provedené změny**	**Zjištěné nejasnosti**
Vybrání místa konání akce	1	41 s	Žádné	Chtěl nastavit adresu, ale byl tam popis „Nastavit souřadnice“.
Vybrání místa konání akce	2	25 s	Přidány dvě tlačítka „Nastavit souřadnice i adresu“ a „Nastavit pouze souřadnice“.	Vybral špatné místo a chtěl se vrátit zpět.
Vybrání místa konání akce	3	11 s	Přidáno tlačítko „Vrátit zpět“.	Žádné
Vybrání zajištěné aktivity, která je povinná.	1	20 s	Žádné	Omylem vybral dobrovolnou aktivitu a formulář prošel.
Vybrání zajištěné aktivity, která je povinná.	2	10 s	Přidáno omezení pro typy aktivit.	Žádné

Tabulka 7.2: Naměřené časy provedených operací

Kapitola 8

Závěr

Tato práce zahrnovala analýzu požadavků a návrh nového rezervačního systému pro spolek Českomoravská federace naturistů (ČMFN). Systém byl navržen s podstatnými rozšířeními oproti zadání. Všechny části zadání a některá rozšíření, jako například rodinní příslušníci či univerzální způsob zadávání času akce, byla implementována a otestována.

Práce by tedy měla plnit funkci rezervačního systému, kdy mají zájemci o danou akci zjednodušený proces přihlašování a mohou snadno uhradit zálohu. Vytváření akcí je nyní systematické, což rovněž usnadní organizátorům akcí práci s přípravou a zajištěním dané akce a přidružených aktivit.

Systém navíc poskytuje funkci automatického strhávání členských příspěvků, což spolku ČMFN může zjednodušit jejich placení a omezit problémy se zapomenutými platbami.

Práce byla vypracována díky podrobným konzultacím, které zajistily navržení systému i v těch méně viditelných detailech.

Do budoucna lze implementovat rozšíření pro vybírání prostředků k aktivitám, fórum k akci, případně šablony sestav pro tiskové výstupy z evidence záznamů. Tato rozšíření byla důkladně konzultována a navržena, takže implementace těchto částí by neměla představovat větší problémy.

Dalším možným rozšířením, které však nebylo diskutováno v rámci této práce, by mohlo být připojení portálu pro nové zájemce o členství ve spolku, kde budou moci klást dotazy a jejich schvalování bude automatické s využitím umělé inteligence. Tím se usnadní práce mnoha moderátorům takových příspěvků.

Jako další rozšíření by se dalo uvažovat placení členských příspěvků pomocí debetní karty nebo zasláním SMS na určené telefonní číslo. To by mohlo zvýšit rozsah systému pro více uživatelů a spolek by poté mohl pořádat více akcí díky vyššímu rozpočtu.

Literatura

[1] Amit. Use Case Diagram. *All You Need to Know About UML Diagrams: Types and 5+ Examples* [online]. St. Louis (Missouri): Tallyfy, c2014–2021 [cit. 2022-07-21]. Dostupné z: https://tallyfy.com/uml-diagram/#use-case-diagram.

[2] Anand, V. How to display dates and times in clients timezone. *PrideParrot* [online]. PrideParrot, 25. září 2011 [cit. 2022-07-26]. Dostupné z: http://prideparrot.com/blog/archive/2011/9/how_to_display_dates_and_times_in_clients_timezone.

[3] Andrew, S. JavaScript – Cookies. *QuirksMode - for all your browser quirks* [online]. Editoval Peter-Paul Koch. [cit. 2022-07-28]. Dostupné z: https://www.quirksmode.org/js/cookies.html.

[4] Böhmer, M. *Návrhové vzory v PHP: 23 vzorových postupů pro rychlejší vývoj*. 1. vyd. Přeložil Lukáš KREJČÍ. Brno: Computer Press, únor 2012. 320 s. ISBN 978-80-251-3338-5.

[5] Čápka, D. Lekce 1 – Úvod do JavaScriptu. *ITnetwork.cz – Učíme národ IT* [online]. Editoval Samuel HÉL. Praha: [b.n.], c2022. Revidováno 4. 5. 2022 [cit. 2022-07-25]. ISSN 2464-6326. Dostupné z: https://www.itnetwork.cz/javascript/zaklady/javascript-tutorial-uvod-do-javascriptu-nepochopeny-jazyk.

[6] Čápka, D. Lekce 4 – UML – Doménový model. *ITnetwork.cz – Učíme národ IT* [online]. Praha: [b.n.], c2022. Revidováno 12. 2. 2021 [cit. 2022-07-28]. ISSN 2464-6326. Dostupné z: https://www.itnetwork.cz/navrh/uml/uml-domenovy-model-diagram.

[7] Čápka, D. Lekce 5 – UML – Class diagram. *ITnetwork.cz – Učíme národ IT* [online]. Praha: [b.n.], c2022. Revidováno 12. 2. 2021 [cit. 2022-07-28]. ISSN 2464-6326. Dostupné z: https://www.itnetwork.cz/navrh/uml/uml-class-diagram-tridni-model.

[8] Contributte. *Contributte Forms-wizard* [online]. Uživatelský manuál, 3.1. © 2020, Revidováno 20. 12. 2021 [cit. 2022-07-25]. Dostupné z: https://contributte.org/packages/contributte/forms-wizard.html.

[9] Contributte. *Contributte Translation* [online]. Uživatelský manuál, 1.0. © 2020, Revidováno 23. 1. 2022 [cit. 2022-07-25]. Dostupné z: https://contributte.org/packages/contributte/translation.html.

[10] Contributte. *Contributte Webpack* [online]. Uživatelský manuál, 2.1. © 2020, Revidováno 16. 12. 2021 [cit. 2022-07-25]. Dostupné z: https://contributte.org/packages/contributte/webpack.html.

[11] Fio banka. *FIO API BANKOVNICTVÍ* [online]. Dokumentace, Verze 1.7.3. © 2022, Revidováno 2. 2. 2022 [cit. 2022-07-28]. 59 s. Dostupné z: https://www.fio.cz/docs/cz/API_Bankovnictvi.pdf.

[12] Fisher, A. a steampowered. Using time zones in a PHP web application. *Stack Overflow* [online]. Editoval Bhanu Singh. New York: Stack Exchange, 8. srpna 2012. Revidováno 5. 8. 2021 [cit. 2022-07-26]. Dostupné z: https://stackoverflow.com/questions/11873234/using-time-zones-in-a-php-web-application.

[13] Fromaget, P. Why is MD5 not secure? *3 Reasons why MD5 is not Secure* [online]. InfosecScout, c2022 [cit. 2022-07-24]. Dostupné z: https://infosecscout.com/why-md5-is-not-safe/#Why_is_MD5_not_secure.

[14] Grudl, D. Nette Framework: MVC & MVP. *Začínáme s Nette Framework* [online]. Praha: Devel.cz Lab. Březen 2009, č. 3, [cit. 2022-07-27]. ISSN 1803-5620. Dostupné z: https://zdrojak.cz/clanky/nette-framework-mvc-mvp/.

[15] Grudl, D. *Nette Framework: Nette – Pohodlný a bezpečný vývoj webových aplikací v PHP* [online]. Verze 3.1. Nette Foundation, c2008–2022 [cit. 2022-07-25]. Dostupné z: https://nette.org/.

[16] Grudl, D. *Latte šablony: Začínáme s Latte* [online]. Dokumentace. © 2008–2022 [cit. 2022-07-28]. Dostupné z: https://latte.nette.org/cs/guide.

[17] Hanel, D. Terminologický slovník: Framework. *Vývoj webových aplikací pomocí frameworku JavaServer Faces* [online]. Praha: Fakulta informatiky a statistiky VŠE v Praze, 9. prosince 2009 [cit. 2022-07-25]. Dostupné z: https://java.vse.cz/jsf/chunks/go01.html.

[18] Koppers, T. *Guides | webpack* [online]. Verze 5. JS Foundation, [c2022] [cit. 2022-07-25]. Dostupné z: https://webpack.js.org/guides/.

[19] Kraval, I. Kdy použít v Use Case Diagramu vztah Extend a kdy Include (část 1). *Návrh IS pomocí UML: Use Cases* [online]. Čermná: Object Consulting. Prosinec 2016, [cit. 2022-07-21]. Dostupné z: https://www.objects.cz/wp/is_uml/kdy-pouzit-v-use-case-diagramu-vztah-extend-a-kdy-include/.

[20] Kumar, R. a Gamal, M. Invalid use of group function while updating a table with sum function. *Stack Overflow* [online]. New York: Stack Exchange, 3. listopadu 2012. Revidováno 15. 4. 2015 [cit. 2022-07-28]. Dostupné z: https://stackoverflow.com/questions/13207069/invalid-use-of-group-function-while-updating-a-table-with-sum-function.

[21] Nothrem. JAK SPRÁVNĚ VYTVOŘIT VLASTNÍ MAKRA V LATTE. *CSS3 Tipy a Triky (NCZ)* [online]. Srpen 2017 [cit. 2022-07-28]. Dostupné z: http://css.nothrem.cz/latte-makra/.

[22] OpenJS Foundation. *jQuery* [online]. Verze 3.6.0. © 2022 [cit. 2022-07-25]. Dostupné z: https://jquery.com/.

[23] Peterson, J. *Official documentation site for Tempus Dominus: Powerful and robust date and time picker* [online]. Verze 6.0.0. © 2021 [cit. 2022-07-25]. Dostupné z: https://getdatepicker.com/.

[24] Seznam.cz. *API Mapy.cz* [online]. Verze 4.13. © 2022 [cit. 2022-07-27]. Dostupné z: https://api.mapy.cz/.

[25] State Machine Diagrams. *UML 2 Tutorial – State Machine Diagram* [online]. Creswick (Australia): Sparx Systems, c2000–2022 [cit. 2022-07-23]. Dostupné z: https://sparxsystems.com/resources/tutorials/uml2/state-diagram.html. Path: Home; Resources; Tutorials; UML 2 Tutorial.

[26] the PHP Documentation Group. *PHP Manual* [online]. Dokumentace. Editoval Peter Cowburn. The PHP Group, c1997–2022, Revidováno 27. 7. 2022 [cit. 2022-07-28]. Dostupné z: https://www.php.net/manual/en/.

[27] Tiny Technologies. Bundling an npm version of TinyMCE with ES6 and Webpack. *TinyMCE 6 Documentation* [online]. Verze 6. © 2022 [cit. 2022-07-25]. Dostupné z: https://www.tiny.cloud/docs/tinymce/6/webpack-es6-npm/.

[28] Wall, P. a PtrTon. Why repository should not return DTO. *Stack Overflow* [online]. New York: Stack Exchange, 14. dubna 2020 [cit. 2022-07-28]. Dostupné z: https://stackoverflow.com/questions/61206934/why-repository-should-not-return-dto.

[29] Welling, L. a Thomson, L. *Mistrovství – PHP a MySQL*. 1. vyd. Přeložil Ondřej BAŠE. Brno: Computer Press, listopad 2017. 800 s. ISBN 978-80-251-4892-1.

[30] Wen, Z., Hernández, D. a Utecht, D. *Bootstrap Table* [online]. Verze 1.20.2. © 2012–2019 [cit. 2022-07-25]. Dostupné z: https://bootstrap-table.com/.

[31] Zendulka, J. a Rudolfová, I. Konceptuální modelování. *Databázové systémy IDS: Studijní opora* [online]. Verze 18. 7. 2006. Brno: Fakulta informačních technologií VUT v Brně, c2005–2006 [cit. 2022-07-26]. Dostupné z: https://wis.fit.vutbr.cz/FIT/st/cfs.php.cs?file=%2Fcourse%2FIDS-IT%2Ftexts%2FDatabazove_systemy.pdf&cid=13974.