ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Vývoj aplikace Evidence školních plateb
Student:	Bc. Miroslav Novotný
Vedoucí:	Mgr. Matěj Jan Morávek
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2017/18

Pokyny pro vypracování

Cílem práce je analýza, návrh a implementace nového modulu desktopové aplikace pro evidenci plateb, které přijímá škola (mateřská, základní, střední) od svých žáků a to jednak za zájmové aktivity (kroužky atp.), školní aktivity (exkurze, výlety, školy v přírodě atp.), případně další, které vyplynou z podrobné analýzy požadavků.
Požadavky na novou aplikaci jsou zejména:
- Evidence osobních účtů žáků /zaměstnanců.
- Implementace do systému Jídelna od firmy Barda SW, HW s.r.o.
- Vazba na moduly hotovostní pokladny, fakturace a komunikace s internetovým bankovnictvím (formát ABO).
- Výstup v .xml pro aplikaci www.jidelna.cz.
- Omezení přístupových práv uživatelů.

Postupujte v těchto krocích:
1. Popište stávající prostředí.
2. Analyzujte, zkompletujte a formalizujte požadavky.
3. Vypracujte návrh, zvolte vhodnou implementační platformu a řešení implementujte.
5. Aplikaci řádně otestujte a zdokumentujte.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 8. února 2017

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Vývoj aplikace Evidence školních plateb

Bc. Miroslav Novotný

Katedra softwarového inženýrství
Vedoucí práce: Mgr. Matěj Jan Morávek

5. ledna 2018

Poděkování

Rád bych poděkoval své rodině a přítelkyni za neoblomnou podporu a svému vedoucímu panu Mgr. Morávkovi za velmi přátelskou pracovní atmosféru.

Prohlášení

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

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

V Praze dne 5. ledna 2018 .....

Odkaz na tuto práci

Novotný, Miroslav. *Vývoj aplikace Evidence školních plateb*. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.

Abstrakt

Tato práce popisuje analýzu, návrh a realizaci rozšíření aplikace Jídelna 2017 poskytované firmou Barda SW, HW s.r.o. Rozšíření spočívá v obecné automatizaci a zjednodušení spravování školních aktivit a plateb žáků za tyto aktivity. Vývoj proběhl v jazyce Java za použití technologií Swing, Hibernate, JasperReports a Spring. Implementace proběhla v pořádku a program splňuje všechny požadavky.

Klíčová slova vývoj aplikace, školní aktivity, platby žáků, Jídelna 2017, Barda SW, HW s.r.o.

Abstract

This work describes the analysis, draft and implementation of additional functionality for Jídelna 2017 application provided by Barda SW, HW s.r.o. The addition deals with general automation and simplification of administration of school activities and student payments for said activities. The development was done in Java programming language using Swing, Hibernate, JasperReports and Spring. The implementation went along successfully and the program meets all its requirements.

Keywords application development, school activites, student payments, Jídelna 2017, Barda SW, HW s.r.o.

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Cíl práce

Cílem práce je rozšíření stávající aplikace Jídelna 2017 od firmy Barda SW, HW s.r.o.[1] o funkcionalitu evidence školních plateb.

Spravovat na škole kroužky, výlety nebo jakékoliv jiné školní aktivity je časově velmi náročná aktivita, která je z velké části velmi repetitivní. Pracovník musí evidovat všechny aktivity, jejich účastníky a platby. Toto často degraduje na neustálou kontrolu velkého počtu seznamů. Navíc je nutné integrovat informace z různých zdrojů jako např.: platby, které mohou být na účet, fakturou a jiné. Někteří rodiče platí aktivity vícero platbami, jiní posílají platby po době splatnosti a je nutné je urgovat. Toto je mnoho poměrně jednoduchých úkonů, které ovšem kumulovaně přes desítky, ne-li stovky různých aktivit představují obrovské množství nesnadné práce.

Výsledek této práce by měl uživateli aplikace Jídelna 2017 od firmy Barda SW, HW s.r.o. velmi zjednodušit, zrychlit a hlavně zpříjemnit celou tuto práci náhrazením velké části jeho práce prací stroje.

Struktura práce

Práce je rozdělana na několik logických bloků, jejich název a popis je uveden níže.

Stávající prostředí

Tato kapitola pojednává o prostředí v jakém byla práce tvořena.

Podkapitola použité technologie obsahuje seznam použitých tenchnologií a jejich stručný popis.

Podkapitola existující funkcionality programu stručně popisuje již implementované schopnosti programu.

Analýza

V této kapitole je rozebrána studie problémové domény a požadavků na výsledek práce.

V podkapitole Rozbor problémové domény jsou rozepsány entity, se kterými se systém bude zabývat. U každé této entity je nastíněno, jak by měla přibližně vypadat a jak by se měla chovat.

V podkapitole Požadavky jsou do většího detailu rozebrány funkční a nefunkční požadavky na práci. Dále jsou zde případy užití a diagram případů užití.

Návrh

Tato kapitola se zabývá návrhem řešení rozšíření splňující požadavky na funkcionalitu programu.

V první podkapitole se hovoří o nutné úpravě přihlašování uživatelů do aplikace.

V druhé podkapitole je velmi stručně nastíněno, jak bude vypadat uživatelské rozhraní.

V dalších kapitolách jsou podrobně rozebrány třídy odpovídající konceptům analýzy a jejich model.

Realizace

V této kapitole je rozebrána implementace potřebných prvků pro zaručení požadované funkcionality.

Podkapitola entity obecně popisuje skutečnosti, které se týkají všech entit nově vytvářených v rámci práce.

Podkapitola databáze rozebírá podobu implementace na úrovni databáze.

Podkapitola mapování popisuje jak jsou tabulky databáze mapovány do programu.

Podkapitola GUI popisuje základní stavební prvky použité při výstavbě grafického uživatelského prostředí.

Testování

V této kapitole jsou popsány různé druhy testování, jejich provedení a výsledky. Dále je zde popsáno jestli výsledek testu odpovídá požadavkům

Vývojové prostředky

Během práce byly použity některé technické prostředky, jejich výčet a popis lze najít níže.

IDE

Pro vývoj a implementaci rozšíření bylo použité vývojové prostředí NetBeans[2]. Toto IDE obsahuje mnoho funkcí, které velmi usnadňují vývoj aplikací.

Zvláštní zmínku pak zaslouží NetBeans IDE GUI Builder[3] díky kterému je tvorba grafického uživatelského prostředí zásadně usnadněna.

Studio pro vytváření tisků

Pro vytváření tisků bylo použito Jaspersort® Studio[4].

UMLet

UMLet[5] je prostředí pro tvorbu UML modelů a byl využit pro všechny modely této práce.

Správa verzí

Pro správu a uložení postupných iterací produktu byl použit verzovací systém Git[6].

Kapitola 1

Stávající prostředí

Jídelna 2017 je desktopová aplikace napsaná v jazyce Java. Java je moderní objektově orientovaný programovací jazyk s širokou komunitou a velkou škálou existujících řešení. Toto je pouze pár z mnoha důvodů jeho velké popularity nejen mezi vývojaři, ale i korporacemi. Pro představu, stránka[7] tvrdí, že 15 miliard zařízení je schopno pustit Java program.

1.1 Použité technologie

Pro zjednodušení práce se základními požadavky na aplikaci je použito několik technologií, některé z nich jsou zmíněny níže.

1.1.1 Swing

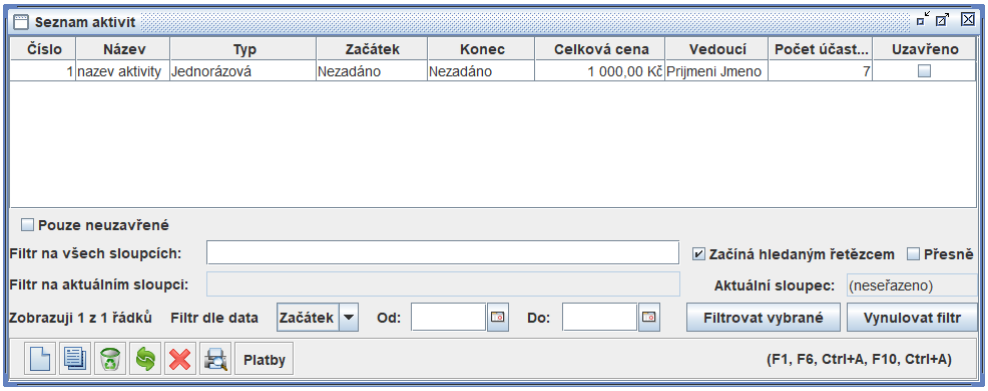
Swing je nástroj pro tvorbu grafického uživatelského prostředí (GUI). Obsahuje mnoho předpřipravených grafických prvků jako např.: okno, textové pole atd., které by klasická aplikace mohla potřebovat a umožňuje velmi snadnou integraci existujících prvků v prvky složitější struktury.

1.1.2 Hibernate

Hibernate[8] je ORM framework, který zásadně zjednodušuje práci s daty v DB. Třídy programu mapuje na řádky databáze pomocí konfiguračních souborů a nebo anotací přímo v kódu.

1.1.3 JasperReports

Pro tvorbu tisků byl použit JasperReports® Library[9].



Obrázek 1.1: Příklad klasického seznamového okna

1.1.4 Spring

Spring Framework[10] je knihovna poskytující mnoho funkcionalit pro podnikové projekty. V této práci je hlavně využito vpravování závislostí (dependency injection).

1.2 Existující funkcionality programu

V následujím textu budou popsány již existující funkcionality programu Jídelna 2017, které mají nějakou spojitost s rozšířením prováděným v rámci této práce.

1.2.1 GUI

Program obsahuje grafické komponenty pro zrychlení tvorby standardních oken. Tyto okna jsou použity prakticky všude a uživatelé jsou na ně zvyklí a chápou jak s nimi pracovat. Všechny následující prvky je možné si prohlédnout na obrázku 1.1 na stránce 6.

1.2.1.1 Tabulka

Chování této komponenty je velmi podobné chování klasické Swingové tabulky, ze které je také právě odvozena. Obsahuje ovšem nadstavby, mezi které patří např. usnadnění vybírání prvků v rámci výběrových oken. Dále obsahuje speciální třídy na odposlech událostí a poskytuje speciální akce pro panel akcí.

1.2.1.2 Panel filtrace tabulky

Panel filtrace tabulky je kompozitní prvek poskytující uživateli možnost vybraným způsobem filtrovat řádky zobrazené v tabulce. Pro podrobnější přehled možností filtrování viz obrázek 1.1 na stránce 6.

1.2.1.3 Panel akcí

Panel akcí poskytuje uživateli možnost okno ovládat. Akce mohou mít více zdrojů:

- okno, např. zavřít
- tabulka, např. nový záznam
- speciální akce přidané extra

1.2.2 Autentizace

Program poskytuje možnost zapnutí autentizace, která probíhá za použití uživatelského jména a hesla.

1.2.3 Autorizace

Při zaplé autentizaci je možné vybrat, který uživatel má přístup ke kterému modulu.

1.2.4 Pokladna

Program disponuje možností evidence virtuální pokladny, která reflektuje stav a historii pokladny fyzické. Tato pokladna je fixně jedna. V rámci této jedné pokladny lze evidovat pokladní doklady dvou typů:

- příjmové, typicky uživatel přijal peníze od žáka
- výdajové, typicky vratky

1.2.5 Fakturace

Obdoba virtuální pokladny pro evidenci přijatých a vydaných faktur. V tomto případě ovšem nejde rovnou o reálně předané či obdržené peníze, ale pouze o předpis. Program obsahuje jednu řadu faktur vydaných a jednu řadu faktur přijatých.

1.2.6 Homebanking

Tento modul dává uživateli možnost importovat výpisy banky a exportovat příkazy pro banku. Podobně jako u pokladny a fakturace, v rámci programu existuje fixně jedna řada výpisů a jedna řada příkazů.

1.2.7 Evidence osob

Program obsahuje evidenci osob relevantních pro běh jídelny. Tyto osoby mohou být nejen žáci dané školy, ale i učitelé a jiní pracovníci.

1.2.8 Evidence firem

Program obsahuje evidenci firem relevantních pro běh jídelny. Tyto firmy mohou být různí dodavatelé či odběratelé služeb.

Kapitola 2

Analýza

2.1 Rozbor problémové domény

Problémová doména přináší několik nových entit, které je nutné rozebrat a přiblížit.

2.1.1 Aktivita

Škola pořádá, či zaštiťuje různé aktivity pro své žáky. Těchto aktivit je velké množství a jsou různého typu. Aktivity mohou být různé kroužky jako např.: odpolední programování, keramika a jiné. Dále do aktivit zapadají různé výlety a kurzy, např.: výlet do divadla, lyžařský kurz a další. Z tohoto se dá usoudit, že existují aktivity dvou typů:

1. aktivity jednorázové
2. aktivity periodické

Aktivity těchto dvou typů se jedna od druhé určitým způsobem liší. Jejich společné znaky a odlišnosti jsou rozebrány ve třech následujících blocích.

2.1.1.1 Obecná aktivita

Aktivita musí nutně mít někoho, kdo ji vede, vedoucího. Aktivita bez vedoucího nemůže existovat. Tato osoba je reálně za žáky během konání aktivity zodpovědná.

Co se zodpovědnosti týče existuje limit, kolik dětí může jedna zodpovědná osoba hlídat najednou. Je tedy nutné, aby kromě vedoucího aktivity, bylo možné aktivitě přidělit také asistenty. Není nutné, aby aktivita asistenta měla, např. účastníků je málo a na daný počet dětí stačí jeden vedoucí. Ovšem může dojít i k opačnému extrému, tj. mnoho účastníků. Např. pro společný výlet několika tříd na lyžařský kurz je pak nutné přiřadit více než jednu zodpovědnou osobu.

Ostatní nezbytné údaje, které jsou v rámci každé aktivity potřeba evidovat jsou:

- název aktivity
- datum splatnosti
- splatná částka
- účastníci

2.1.1.2 Jednorázová aktivita

Jednorázová aktivita je taková aktivita, která se v čase stane jenom jednou a po jejím ukončením ji lze uzavřít, je nepravděpodobné, že by se ve stejném nastavení opakovala. Taková aktivita může být např.: školní výlet třídy 2.C na konci roku 2018, prosincový výlet do divadla vybraných žáků nebo jiné.

Taková aktivita má jeden jasně daný začátek a jeden jasně daný konec. Např. na třídní výlet se jede v pátek v 17:00 a návrat bude tentýž týden v sobotu v 10:00.

Pokud se jedná o jednorázovou aktivitu, která je nákladnější jako např. lyžařský kurz, je nutné mít možnost platit takovouto aktivitu postupně. Minimálně je nutné mít kromě data splatnosti a splatné částky datum zálohy a výši zálohy.

Naopak existují aktivity, které jsou poměrně levné a není nutné, aby jejich placení procházelo přes školu. Např. výlet na horu Říp, kde účastník efektivně platí pouze dopravu tam a zpět. Po debatě na toto téma bylo usouzeno, že i takovéto aktivity je dobré v systému evidovat. Jednorázová aktivita má tedy navíc přepínač, jestli je placena na místě, nebo ne. Toto efektivně anuluje význam datumu splatnosti, splatné částky, datumu zálohy a výši zálohy.

2.1.1.3 Periodická aktivita

Periodická aktivita je taková aktivita, která se v čase opakuje s určitou periodou. Taková aktivita může být třeba školní kroužek jako keramika, plavání či jiné.

Každá periodická aktivita má určitý časový úsek, kdy se periodicky opakuje. K tomuto úseku je nutné navíc udat periodu. Např. na plavání budou žáci tento rok jezdit od 1.9. do 1.12. jednou týdně.

Z hlediska přehlednosti je dobré u periodické aktivity dobré ještě evidovat čas od kdy do kdy tato aktivita probíhá. Ve větších školách může být více stejných kroužků, probíhajících v jiné časy. Např. programování od 15:00 a programování od 16:00.

2.1.2 Instituce

Co se týče zaštiťování a pořádání aktivit pro žáky, škola jako taková nemusí nutně být jediný aktér. Mohou existovat další spolky, jako např. SRPŠ, které pro žáky školy zařizují aktivity a i ty by měly mít možnost využít evidenci plateb. Je tedy nutné aby se základní funkcionalita dala dělit mezi více pseudo rozlišnými institucemi. Každá taková instituce si spravuje svoje vlastní aktivity, na ty vybírá peníze atd., ovšem penzum osob je identické.

V rámci každé instituce je nutné odstínit její specifická data od dat jiných institucí. Odstínění se týká:

- aktivit
- jakýchkoliv peněžních záznamů
  - pokladní doklady
  - faktury
  - výpisy a příkazy banky

V rámci každé instituce aktivity jako takové procházejí nějakou formou schválení a tím pádem je nutné, aby ne každý uživatel, ačkoliv má přístup k modulu školních plateb, protože je např. vedoucí aktivity a potřebuje vidět tato data, měl možnost např.: založit novou aktivitu, smazat aktivitu a další. Stejně tak není možné aby, mohl kdokoliv s přístupem k evidenci plateb přijmout a do systému zaevidovat platbu.

2.1.3 Osoba aktivity

Osoba může v rámci aktivity figurovat v několika rozdílných rolích.

2.1.3.1 Vedoucí aktivity

Vedoucí aktivity je taková osoba, která reálně aktivitu vede a je za ní zodpovědná. Osoba aktivity musí mít povoleno být vedoucím, aby se v této pozici mohla ocitnout. Toto má předejít nežádoucím chybám, jako např. žák 1. ročníku základní školy jako vedoucí kurzu.

2.1.3.2 Asistent aktivity

Asistentem aktivity může být v rámci systému prakticky kdokoliv. Nezávisle, jestli je osoba učitel nebo žák, je možné ji do této role přidělit. Často se reálně stává, že žáci starších ročníků pomáhají během aktivit žáků ročníků nižších. Jediná výjimka je, že asistent aktivity nemůže být zároveň vedoucí té samé aktivity.

2.1.3.3 Účastník aktivity

Účastníkem aktivity může být kdokoliv v rámci systému. Je nutné, aby účastníkem mohla být i osoba, která je u dané aktivity vedena jako vedoucí nebo asistent. Jelikož platby za aktivity vedoucích a asistentů se mohou velmi výrazně lišit jak od plateb účastníků tak ještě mezi sebou, nejsou tyto platby v evidenci vedeny. Pokud se od plateb účastníků neliší, přidáním těchto osob navíc mezi účastníky zajistí jejich evidenci jako obecných účastníků.

2.1.4 Příspěvek na aktivitu

Občas se může stát, že žák dostane na aktivitu příspěvek. Tyto příspěvky je nutné evidovat a to zvlášť od peněz zaplacených účastníkem.

U příspěvků je nutné evidovat následující skutečnosti

- kdo příspěvek dostal
- kdo příspěvek poskytl
- na kterou aktivitu je příspěvek poskytnut
- výše příspěvku

Příspěvěk může dostat libovolný žák na kteroukoliv z jeho aktivit. Počet příspěvků pro jednoho účastníka na jedné aktivitě není limitován.

2.1.5 Placení aktivit

Aktivita může být zaplacena více způsoby, mezi ty patří:

- na místě
- na účet instituce
- hotově
- fakturou

Co se týče aktivit s vyšší cenou, není nutné, aby byly zaplaceny jednou platbou. Naopak, je požadováno, aby taková aktivita mohla být zaplacena více platbami.

Placení na místě bylo rozebráno v rámci jednorázové aktivity 2.1.1.2 na stránce 10 a nebude zde opakováno.

2.1.5.1 Platby na účet

Každá osoba vedená v systému má přidělen unikátní variabilní symbol. Nově tvořené aktivity budou mít přiřazen unikátní specifický symbol. Pro správné automatické spárování v systému je nutné, aby plátce za účastníka správně zadal variabilní symbol a specifický symbol. Zádání správné částky není vyžadováno. Naopak, plátce musí mít možnost rozplánovat si platbu do více plateb.

Pokud plátce aktivitu přeplatí nebo zaplatí špatně, systém již v nynější verzi umožňuje peníze na účet vrátit.

2.1.5.2 Platby pokladnou a fakturou

Účastník by měl mít možnost zaplatit aktivitu hotově nebo přes fakturu. Systém tyto způsoby již umožňuje. Akorát je třeba je upravit, aby je bylo možné používat ve spojistosti s účastníkem a aktivitou.

2.1.5.3 Konto účastníka

Je nutné, aby bylo v systému poznat, která aktivita je zaplacena a která ne. Není tedy možné, aby se vedlo jednoduché konto, kde by byl pouze stav aktuálních peněžních prostředků osoby, jak tomu bylo v systému doposud.

2.1.6 Posílání emailů

Pokud je v programu nastaveno SMTP, pak lze odesílat emaily. Problém však tkví ve skutečnosti, že emaily o událostech se odesílají ihned, jak událost vznikne. Toto v závislosti na evidenci plateb musí být upraveno, jelikož by bylo dobré poslat jednu urgenci za všechny aktivity najednou a ne několik urgencí za každou aktivitu dle toho, jak je postupně uživatel generoval.

2.2 Požadavky

Každý projekt má své požadavky. Tyto požadavky mohou být dvojího charakteru, funkční a nefunkční. Sbírání požadavků bylo cíleno na požadavky funkční.

2.2.1 Funkční požadavky

Funkční požadavky jsou takové požadavky na projekt, které přímo ovlivňují nějakou funkčnost, schopnost systému. Např. je nutné umět vytvořit aktivitu.

V tabulce 2.1 na stránce 15 jsou vypsány funkční požadavky rozšiřující požadavky zadání.

Co se týče požadavků zadání, požadavek „Výstup v .xml pro aplikaci www.jidelna.cz“ byl během tvorby práce vypuštěn z důvodu jeho nízké důležitosti.

2.2.2 Nefunkční požadavky

Nefunkční požadavky jsou takové požadavky na projekt, které ovlivňují aspekty chování aplikace. Oproti funkčním požadavkům nejsou zaměřeny na určitou funkci systému. Nefunkční požadavek může být např. musí to být rychlé.

V tabulce 2.2 na stránce 16 jsou vypsány nefunkční požadvky rozšiřující požadavky zadání. Parametry splnění některých těchto požadavků se nedají lehce určit a tedy jejich splnění je spíše téma k diskuzi. V rámci práce jim je věnovaná vyšší pozornost a je diskutováno jejich splnění či porušení.

2.3 Případy užití

V následujících pár blocích jsou rozebrány některé důležité případy užití.

2.3.1 UC1: Scénář vyhledání

Následující výčet osahuje seznam kroků scénáře vyhledávání. Nekteré kroky tohoto seznamu jsou vypsány kurzívou. Tyto kroku jsou nepovinné a není třeba je provést k úspěšnému dokončení scénáře.

- zobrazení okna seznamu pro danou entitu
- *zadání vyhledávacích parametrů*
- *zobrazení detailu*
- vybrání hledané entity / hledaných entit

2.3.2 UC2: Scénář vytvoření aktivity

Uživatel otevře okno seznamu aktivit a zde použije ikonu nový. V rámci vyplňování údajů o aktivitě je nutné zvolit vedoucího. K tomuto se použije UC1. Dále je možné zvolit asistenty a účastníky, opět za použití UC1. Dokončení vytváření se provede kliknutím na ikonu uložit. Nově vytvořená aktivita se automaticky zobrazí v okně seznamu aktivit.

2.3.3 UC3: Scénář urgování plateb

Uživatel otevře okno seznamu osob aktivit a zde použije akci „Urgovat nedoplatky“. Tato akce automaticky vyhledá osoby aktivit a jejich aktivity, které je momentálně třeba urgovat a vygeneruje pro ně email. Tyto emaily lze odeslat ihned nebo později.

Tabulka 2.1: Funkční požadavky

Název	Popis	Závažnost
Evidence institucí*	Systém bude evidovat všechny jemu známé instituce, které zaštiťují aktivity žáků.	Vysoká
Evidence aktivit*	Systém bude evidovat všechny aktivity všech evidovaných institucí.	Vysoká
Evidence účastníků*	Systém bude evidovat které osoby se zůčastňují kterých aktivit.	Vysoká
Evidence příspěvků*	Systém bude evidovat které osoby mají jaké příspěvky na kterých aktivitách.	Vysoká
Evidence plateb	Systém bude evidovat platby účastníků.	Vysoká
Asistenti aktivity	Zobrazit, přidat, odebrat, změnit.	Vysoká
Účastnící aktivity	Zobrazit, přidat, odebrat, změnit.	Vysoká
Platba účastníka	Párování plateb s účastníky.	Vysoká
Platební opozdilci	Zobrazení účastníků, kteří nemají zaplacenou aktivitu.	Vysoká
Urgence	Zaslání upozornění, že má být aktivita zaplacena.	Vysoká
Platební opozdilci tisk	Duplicitní, přehledný tisk.	Nízká
Aktivity uživatele	ST a správce vidí vše. Ostatní vidí pouze aktivity, kde jsou vedoucím, asistentem nebo účastníkem.	Vysoká
Detail aktivity tisk	Duplicitní, přehledný tisk.	Střední
Unikátnost SS	Různé aktivity musí mít různé specifické symboly.	Vysoká
Aktivita v minulosti	Varovat uživatele, že vytváří aktivitu v minulosti.	Nízká
Zodpovědnost	Varovat uživatele, že počet dětí na zodpovědné osoby překračuje danou hranici.	Nízká
Málo účasníků	Zobrazení aktivit, které mají méně než X účastníků.	Střední
Účast zodpovědných	Možnost přidat vedoucího či asistenta jako účastníka.	Vysoká
Více SMTP	Možnost konfigurovat SMTP v rámci každé instituce jinak.	Střední
Bilance aktivity tisk	Tisk zobrazující peněžní aktivity.	Střední

* Pod pojmem evidence je v těchto bodech myšlena schopnost založit nový záznam, upravit záznam, smazat záznam a zobrazit seznam známých záznamů.

Tabulka 2.2: Nefunkční požadvaky

Název	Popis	Závažnost
Standardní GUI	Implementace bude používat standardních firemních prvků GUI pro danou aplikaci.	Vysoká
Standardní tisky	Implementace bude používat standardních firemních prvků vzhledu tisků pro danou aplikaci.	Vysoká
Výpočty mimo EDT	Delší výpočty by měly probíhat mimo EDT, tzn. aplikace by neměla „zamrznout“.	Vysoká
Minimální dopad	Dopad změn na stávajícího uživatele, který nebude pořizovat evidenci plateb by měl být co možno nejmenší.	Střední

2.3.4 UC4: Scénář zobrazení plateb osoby vůči aktivitě

Jsou dva možné průchody grafickým uživatelským prostředím jak zobrazit platby osoby na aktivitě.

2.3.4.1 1. průchod

Uživatel otevře seznam osob aktivit a vybere osobu jejíž platby chce zobrazit a otevře detail. Zde najde záložku aktivit, kterých se osoba účastní. V této záložce vybere požadovanou aktivitu a použije akci „Platby aktivity“.

2.3.4.2 2. průchod

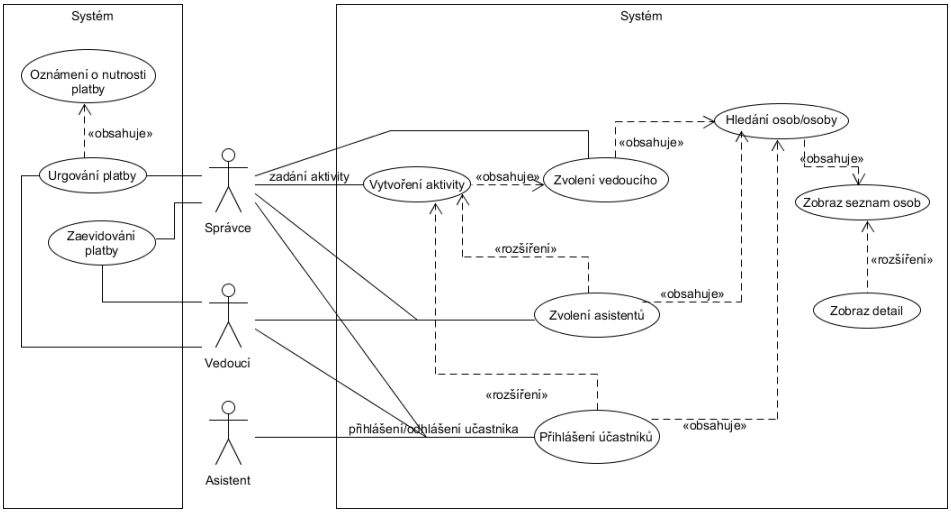
Uživatel otevře seznam aktivit a vybere aktivitu, ke které chce zobrazit platby a otevře detail. Zde najde záložku účastníků. V této záložce vybere požadovaného účastníka a použije akci „Platby účastníka“.

2.3.5 UC5: Scénář zobrazení plateb na aktivitě

Uživatel otevře seznam aktivit a vybere aktivitu, ke které chce zobrazit platby a otevře detail. Zde použije akci „Platby“.

2.4 Diagram případů užití

Obrázek 2.1 na stránce 17 zobrazuje diagram případů užití. Blok systému je rozdělen do dvou částí pouze z důvodu nižšího křížení čar, žádná hlubší sémantika se pod tímto zobrazením neskrývá.



Obrázek 2.1: Diagram případů užití

2.4.1 Aktéři

V diagramu jsou tři aktéři. Správce, vedoucí a asistent. Účastník se v diagramu neobjevuje, jelikož většinou osoba, která figuruje jenom jako účastník, nebude mít do systému vůbec přístup.

Správce je role v rámci instituce. Každá instituce bude muset mít alespoň jednoho správce. Osoba označená jako správce, bude mít pravomoc provádět skoro všechny akce systému.

Vedoucí a asistent je role v rámci aktivity. Každá aktivita musí mít vedoucího a může, ale nemusí mít asistenty.

Je nutno podotknout, že osoba a uživatel nejsou zaměnitelné termíny. Tzn. že nezávisle na roli osoby, pokud nemá zřízený účet uživatele a tedy nemá přístup do systému, nemůže provádět žádné akce.

2.4.2 Tvorba a editace aktivity

Vytvořit aktivitu smí pouze uživatel, který je správcem dané instituce. Během vytvoření aktivity je nutné zvolit vedoucího. Nepovinné další kroky jsou zvolení asistentů a zvolení účastníků.

2.4.2.1 Změna vedoucího

Vedoucího aktivity může změnit jenom správce instituce. Vedoucí aktivity, pokud není správcem instituce, nemůže změnit vedoucího aktivity. Neboli nemůže předat svoji pozici vedoucího někomu jinému. Asistent aktivity taktéž nemůže změnit vedoucího.

2.4.2.2 Změna asistentů

Změna asistentů se bere jako přidání či odebrání asistenta z dané aktivity.

Asistenta aktivity může změnit správce instituce a vedoucí aktivity. Asistent aktivity nemůže měnit sestavu asistentů.

2.4.2.3 Změna účastníků

Změna účastníků, obdobně jako u změny asistentů viz 2.4.2.2, se bere jako přidání či odebrání účastníka z dané aktivity.

Účastníka aktivity smí změnit správce instituce, vedoucí aktivity i asistent aktivity.

2.4.2.4 Změny obecně

Každá změna vedoucího, asistentů či účastníků se skládá ze stejných kroků. Tyto kroky jsou popsány v rámci UC1 viz 2.3.1 na stránce 14.

2.4.3 Evidování a urgence plateb

Evidovat platby za aktivity nebo urgovat nedoplacené či nezaplacené aktivity mohou pouze spravci instituce a vedoucí aktivity. Asistent aktivity by k těmto funkcím neměl mít přístup.

2.5 Nutné změny stávající funkcionality

Zavedení evidence plateb sebou nese několik nevyhnutelných změn. Tyto změny je nutné do systému představit, aby mohla evidence plateb řádně fungovat.

2.5.1 Autentizace a autorizace

Momentálně poskytne přihlášení do systému téměř plnou funkcionalitu v rámci licence. Jediné existující omezení, které je možné na uživatele použít v tuto chvíli, je odstranění povolení na použití určitého modulu. Takováto autorizace je pro evidenci plateb příliš hrubá a je nutné ji rozšířit (více viz 2.1.2 na stránce 11).

Evidence uživatelů není nikterak propojena s evidencí osob. Jelikož se zásadně pracuje s evidencí osob a data musí být filtrována dle přihlášeného uživatele viz tabulka 2.1 na stránce 15, je nutné najít způsob jak tyto dvě evidence propojit.

2.5.2 Peněžní informace

Každá instituce musí mít kompletně celý modul pokladny, fakturace a homebankingu jen pro sebe a své uživatele. Tyto agendy musí být nezávislé a

oddělené více viz 2.1.2 na stránce 11. Navíc je potřeba všechny tyto agendy upravit tak, aby mohly být napojeny na účastníka a jeho aktivitu a mohly tak evidovat jejich platby.

Ostatní položky menu finanční operace, což jsou česká spořitelna a česká pošta, nebudou aktivity a účastníky vůbec řešit. Tato funkcionalita je výběhová a není nutné ji do této práce integrovat. Ovšem musí být oddělena, stejně jako výše zmíněné položky, aby si navzájem instituce nemohly nahlédnout do finančních záležitostí.

2.5.3 Pokladna

V rámci jedné instituce by mělo být možné založit více než jednu pokladnu. Reálně na škole může vybírat peníze za kroužek či výlet více než jedna osoba a systém by to měl reflektovat.

2.5.4 Elektronická pošta

Nastavení SMTP serveru by mělo být přesunuto pod instituci, aby si každá mohla zvolit to svoje. Dále je potřeba změnit způsob odesílání emailů tak, aby se neodesílali ihned po jejich vytvoření viz 2.1.6 na stránce 13.

Kapitola 3

Návrh

Návrh rozšíření bude mít podobu klasické třívrstvé architektury. Vrstvy této architektury jsou:

- prezentační vrstva
- logická vrstva
- datová vrstva

Prezentační vrstva se stará o prezentaci dat uživateli a přijímá od něj požadavky. Jinými slovy to je uživatelské rozhraní.

Logická vrstva se stará o naplnění byznys logiky dané aplikace. Tato vrstva reálně naplňuje tu funkci, která se od systému očekává.

Datová vrstva se stará o perzistentní uložení spravovaných dat. Databáze velmi často musí zaručovat principy ACID, neboli atomicitu, konzistenci, izolaci a stálost.

3.1 Přihlášení

Požadavky na práci sebou nesou změny, které se nutně musí odrazit na přihlašování do aplikace.

3.1.1 Uživatel

Tabulka uživatelů není nijak propojena se seznamem osob. Toto je nutné pro splnění požadavků změnit.

Nejjednoduší a nejprůhlednější je dle informací z tabulky uživatelů založit do systému osoby nové, které pak budou napojeny na tabulku uživatelů. Uživatelé kromě uživatelského jména a hesla, které jsou přímou záležitostí uživatele, nemají žádné jiné jednoduché atributy, které by již nebyly přítomny v záznamu osoby. Takto se zlepší a zjednodušší práce budoucích programátorů a navíc se sníží redundace v systému.

Tato změna se zásadně projeví pouze v kódu. Uživatel systému si možná všimne několika osob navíc v evidenci osob a toho, že lze v detailu uživatele specifikovat detail osoby, ale to je vše. Na jeho práci to prakticky nemá žádný dopad.

3.1.2 Instituce

Jelikož některé části systému jsou omezeny na instituci a ne celek, je nutné někde před jejich dosažením tuto instituci zvolit. Ideálním místem je přihlášení uživatele do systému.

V okně přihlášení se tedy nově objeví možnost zvolit instituci. Uživatelé, kteří nebudou používat školní platby, budou nutně mít v systému jenom jednu instituci a to tu, která se založí automaticky během aktualizace, více viz 3.4 na stránce 23. Tato instituce bude automaticky předvolena a jelikož neexistuje žádná další, nepůjde s prvkem více manipulovat. Tím pádem je tedy dopad na stávajícího uživatele, který nepoužívá novou funkcionalitu, velmi nízký.

3.1.3 Viditelnost

Od přihlášení bude zobrazeno pouze to, co je napojeno na danou přihlášenou instituci. Toto platí pro části systému, které mají na instituci návaznost. Ostatní části bez závislosti na instituci se budou zobrazovat v plné míře v rámci každé instituce.

3.1.4 Souhrn

Stejně jako funguje globální nastavení momentálně přihlášeného uživatele bude nyní navíc fungovat momentálně přihlášená instituce. Je nutné projít všechna seznamová okna, kterých se omezení dle instituce týká a upravit metody a dotazy tak, aby tuto informaci využily.

3.2 Prezentační vrstva

Co se týče uživatelského rozhraní, jediné uživatelské rozhraní přístupné uživateli bude rozhraní grafické. V rámci tohoto rozhraní budou existovat dva základní prvky, pomocí kterých uživatel bude provádět potřebné úkony. Tyto prvky jsou okno seznamu a okno detailu.

3.2.1 Okno seznamu

V rámci seznamových oken bude zobrazen přehled všech dostupných entit. Tento přehled bude obsahovat pouze vybrané informace o prvcích. Některá seznamová okna budou obsahovat omezení na množinu zobrazených entit. Např. seznam aktivit bude omezen momentálně přihlášenou institucí a momentálně přihlášeným uživatelem.

3.2.2 Okno detailu

Detailová okna budou obsahovat a umožňovat změnu většiny informací. V rámci některých oken mohou existovat různá omezení nebo úplný zákaz editace vybraných prvků. Např. v detailu instituce nepůjde normálnímu uživateli změnit název instituce a některé další informace, které jednoznačně určují subjekt.

3.2.3 Dosavadní zvyklost

Momentálně jsou uživatelé zvyklí většinu přehledových informací získávat ze seznamových oken a do oken detailu většinou přistupují pouze kvůli editaci. Jelikož je ale evidence plateb vysoce propojena, není možné, aby byly všechny zásadní informace v řádku seznamového okna. Např.: řádek aktivity v sobě nemůže mít všechny asistenty, účastníky a další. Je tedy nutné v rámci detailových oknech zřídit extra další tabulky těchto propojení a uživatelské akce cílit na detail, ne seznam. Toto je změna, která má dopad na uživatele a musí s ní být seznámen, ovšem v rámci práce se jeví jako nevyhnutelná.

3.3 Návrh tříd

Na obrázku 3.1 na stránce 24 je zobrazen návrh tříd. Tento návrh se zabývá pouze perzistentními třídami.

3.3.1 Základní popis

V návrhu tříd jsou některé části podbarevny oranžovou barvou. Toto jsou třídy či propojení, které již jsou součástí systému.

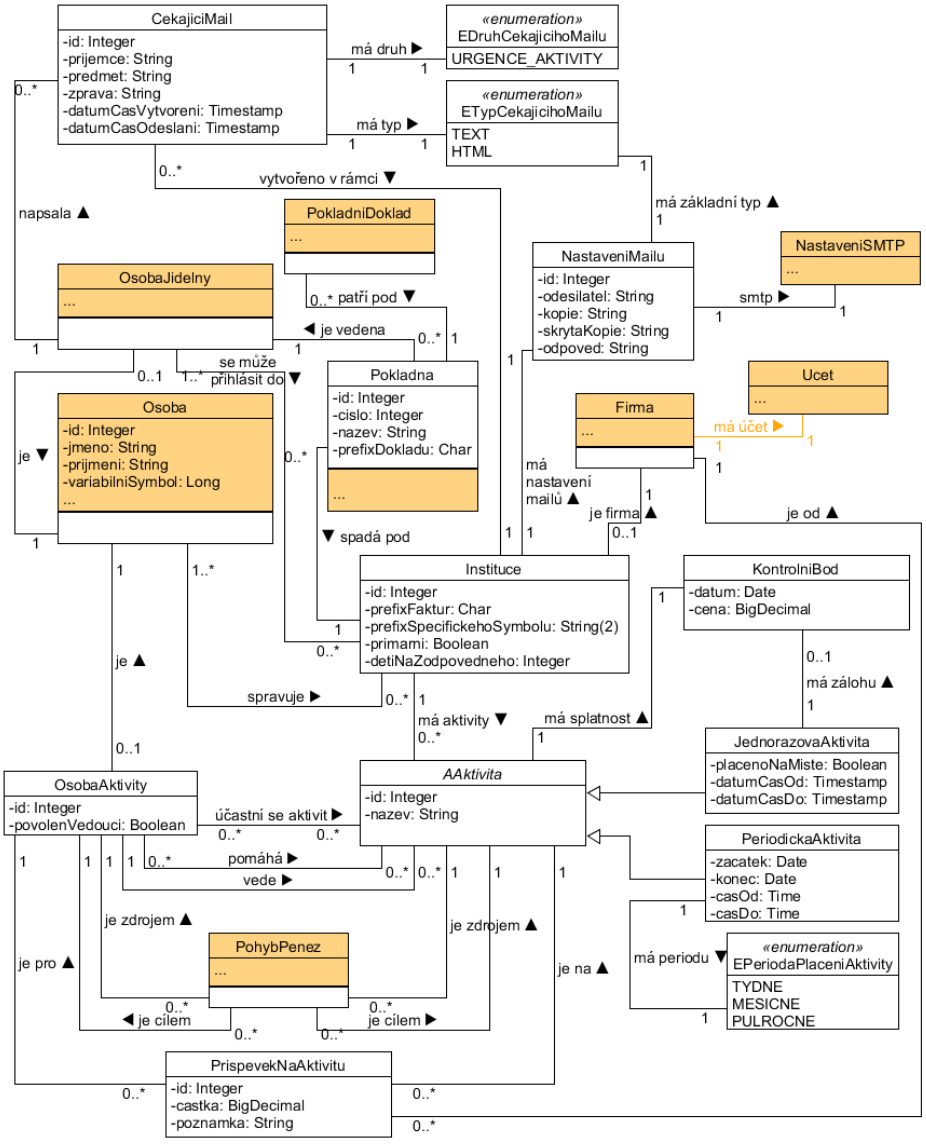
S ohledem na přehlednost a velikost návrhu tříd jsou některá propojení a třídy vynechány. OsobaAktivity a AAktivita jsou propojeny se všemi zdroji či předpisy peněz jako je např.: pokladní doklad a další. Chybí i některé atributy, které jsou buď převzaté a nebo nijak zvlášť relevantní.

V návrhu jsou zobrazeny pouze abstraktní a konkrétní třídy, rozhraní jsou úplně vypuštěna. Reálně však bude muset mít obecné rozhraní naprostá většina.

3.4 Instituce

V rámci programu musí instituce mít unikátní prefix faktur a unikátní specifický symbol. Toto pomůže lepší orientaci a kontrole chyb.

Jelikož bude nyní možné v rámci jedné jídelny mít více institucí, je nutné rozlišit řady jejich učetních dokladů. První instituce, vytvořená během upgradu DB, musí mít nastaven prefix faktur na mezeru. Je nutné, aby se během účetního období nezměnila řada, do které dané doklady patří. Použitím mezery se označení dokladů v podstatě nezmění.



Obrázek 3.1: Návrh tříd

3.4.1 Upgrade DB

V rámci upgradu DB je nutné vytvořit jednu novou instituci a převést na ní všechny momentálně existující záznamy, které v nové verzi systému budou institucí odstíněny.

3.4.2 Vazby na ostatní třídy

Třída instituce má vazby na následující seznam tříd. V následujících blocích budou tyto vazby stručně popsány.

- Firma
- NastaveniMailu
- CekajiciMail
- Pokladna
- OsobaJidelny
- Osoba
- Aktivita
- Stanice

3.4.2.1 Vazba na firmu

Toto je jednoduchá vazba, která říká, že instituce jako taková, je vlastně firma. V tomto případě, instituce obsahuje firmu. Co se tedy týče evidence firem, nově vytvořené instituce budou mít firemní složku, která se v této evidenci objeví.

3.4.2.2 Vazba na nastavení emailů

Tato vazba je jednoduchá a dává možnost nastavit posílání emailů v každé instituci jinak.

3.4.2.3 Vazba na čekající emaily

Čekajících emailů může být v rámci jedné instituce jakýkoliv počet. Je nutné čekající emaily na instituci napojit, aby byly poslány pod správným nastavením.

3.4.2.4 Vazba na pokladny

Pokladen může být v jedné instituci více. Pokladny musí být na instituci napojeny, aby v rámci přihlášení bylo možné zobrazovat pouze relevantní data. Instituce si navzájem do pokladen nesmí vidět.

3.4.2.5 Vazba na uživatele

Tato vazba plní požadavek omezení uživatelů, kteří mají povolení se do dané instituce přihlásit.

3.4.2.6 Vazba na osoby

Tato vazba zajišťuje možnost přidat instituci správce. Na první pohled by se mohlo zdát, že vazba by měla být do tabulky uživatelů, ne osob. Pro zvolení vazby na osoby jsou dva hlavní důvody.

Za prvé to rozvolňuje posloupnost uživatelských kroků při vytváření správce instituce. Za druhé může reálně existovat osoba správce instituce, který ovšem nemá přístup do programu. Např. ředitel, který oficiálně má možnost vidět vše, ale tuto možnost zatím nevyužívá.

3.4.2.7 Vazba na aktivity

Tato vazba zajišťuje schopnost filtrovat aktivity dle momentálně přihlášené instituce a tedy odstínit aktivity jedné instituce od druhé.

3.4.2.8 Vazba na stanici

Tato vazba pouze usnadňuje a zrychluje přihlašování. Pokud přihlašující se stanice už někdy byla v programu přihlášena, je u ní uvedeno pod jakou institucí pracovala a proběhne předvybrání této instituce.Tato vazba má momentálně smysl pouze ze strany stanice.

3.4.3 Přístup

Uživatel bude nově napojen na evidenci osob. Tím pádem by každý uživatel měl přístup do všech institucí. Toto musí být omezeno, instituce tedy musí mít seznam uživatelů, kterým povoluje přihlášení.

3.4.4 Vypnuté přihlašování

Někteří zákazníci mohou mít vypnuté přihlašování do aplikace. V tomto případě je nutné vědět, která instituce se má primárně spustit. Po upgradu DB bude primární instituce ta, která byla během upgradu vytvořená. Toto by mělo jít změnit v rámci detailu instituce. Je nutné podotknout, že vždy musí být právě jedna instituce primární.

3.4.5 Viditelnost

Uživatel bude vidět pouze ty instituce, ve kterých je uveden jako správce. Ostatní uživatelé tyto informace vůbec nepotřebují vidět.

3.4.6 Ochrana licence

Rozšíření programu o evidenci plateb efektivně duplikuje některé funkcionality pro použití větším počtem subjektů. Je potřeba tedy zabránit tomu, že by se fakticky pořídila individuální licence, pod kterou by se spravovalo více škol. Toto je zařízeno tak, že založení instituce může provést pouze ST.

Podobný problém by nastal, pokud by uživatel měl možnost měnit klíčové atributy firmy instituce 3.4.2.1. Změnou názvu, adresy a bankovního spojení efektivně změní subjekt. Je tedy nutné změnu těchto údajů také omezit na ST. Toto se ovšem týká pouze firem napojených na instituci. Pokud takové napojení neexistuje, není potřeba tuto ochranu zavádět.

3.5 Aktivita

AAktivita je abstraktní předek pro JednorazovaAktivita a PeriodickaAktivita.

3.5.1 Vazby na ostatní třídy

Třída aktivity má vazby na následující seznam tříd. V následujících blocích budou některé tyto vazby stručně popsány. Vynechané vazby již byly popsány z druhé strany.

- Instituce, 3.4.2.7
- PrispevekNaAktivitu
- PohybPenez
- OsobaAktivity

3.5.1.1 Vazba na příspěvky

Tato vazba určuje příspěvky, které k dané aktivitě patří. Každý příspěvek má ještě navíc vazbu na osobu aktivity, pevně určující, která osoba aktivity dostala příspěvek na kterou aktivitu.

3.5.1.2 Vazba na pohyby peněz

Tyto vazby spojují aktivity s virtuálním pohybem peněz v systému. Aktivita může být cílem pohybu nebo může být zdrojem pohybu.

Pokud je uvedena pouze aktivita a ne osoba aktivity na pohybu peněz, znamená to, že pohyb se žádné osoby aktivity netýká. To se může stát, když např. uživatel zaplatí poskytovateli služby za uskutečněnou aktivitu.

3.5.1.3 Vazba na osoby aktivit

Vazby na osoby aktivit jsou tři různé druhy, vazby na účastníky, vazby na asistenty a vazba na vedoucího.

Vazba na vedoucího je jednoduchá 1:N vazba, přes kterou je identifikován vedoucí aktivity.

Vazby na účastníky a asistenty jsou M:N vazby.

3.5.2 Vypnuté přihlašování

Co se týče aktivit, pokud bude mít uživatel vypnuté přihlašování, bude brán jakoby byl příhlášen správce instituce.

3.5.3 Viditelnost

Správce instituce uvidí všechny aktivity vedené pod danou institucí. Nesprávce uvidí pouze ty aktivity, se kterými má něco k dočinění, tzn. je vedoucím, asistentem a nebo účastníkem.

3.6 Osoba aktivity

Jediný jednoduchý atribut osoby aktivity je značka, jestli daná osoba může figurovat jako vedoucí aktivity. Tento atribut se dá předvyplnit z osob, do kterých ho některé školy vyplňují z aplikace Bakaláři[11]. Tato položka říká, jestli daná osoba figuruje jako učitel nebo žák.

3.6.1 Vazby na ostatní třídy

Třída osoba aktivity má vazby na následující seznam tříd.

- Osoba
- PrispevekNaAktivitu, 3.5.1.1
- PohybPenez, 3.5.1.2
- Aktivity, 3.5.1.3

3.6.1.1 Vazba na osobu

Váže rozšířenou osobu aktivity na obecnou osobu. Tímto způsobem není osoba kontaminována atributy, které se jí nutně nemusí týkat a osoba aktivity benefituje ze všech informací a funkcionalit osoby.

3.6.2 Viditelnost

Správce instituce uvidí všechny osoby. Nesprávce uvidí pouze ty osoby, se kterými má nějakou společnou aktivitu a to v jakékoliv roli.

3.7 Emaily

V rámci nezahlcování rodičů vysokým počtem emailů je nutno tyto emaily určitým způsobem agregovat a kombinovat.

3.7.1 Čekající email

Emaily se nesmí odesílat ihned, místo toho se budou ukládat do databáze, která bude sloužit jako jejich fronta. Odeslání se bude provádět až zpětně.

3.7.2 Kombinování

Jelikož jsou emaily generované, je jejich struktura i obsah velmi dobře znám. Navíc jsou uložené v DB, kde mohou čekat na odeslání libovolnou dobu. Není tedy vůbec těžké před odesláním zpráv jich co největší počet spojit. Emaily, které patří stejnému adresátovi, jsou stejného typu a jedná se o stejnou věc. Např. místo dvou emailů pro pana Nováka, typu text a druhu urgence aktivity, lze odeslat jeden, kde bude kombinované tělo těchto dvou emailů. Co se předmětu emailu týče, je možné ho náhodně vybrat z množiny předmětů sjednocovaných emailů, jelikož jsou všechny identické.

3.7.3 Vazby na ostatní třídy

Třída osoba aktivity má vazby na následující seznam tříd.

- OsobaJidelny
- Instituce, 3.4.2.3

3.7.3.1 Vazba na uživatele

Tato vazba nemá žádný dopad na email samotný nebo jeho poslání. Pouze eviduje autora emailu. Za autora se uživatel počítá i když je email generovaný, neboli přihlášený uživatel spustil akci generující emaily.

3.8 Pokladna

Pokladny jako takové spadají pod instituci a v rámci této instituce musí mít unikátní prefixy pokladních dokladů. Dále nově musí mít pokladníka, což bude uživatel jídelny.

První pokladna, založená během upgradu DB, musí mít nastaven prexif pokladních dokladů na mezeru. Důvod je stejný jako u prefixu faktur u instituce, během účetního období se nesmí změnit prefix.

3.8.1 Upgrade DB

V rámci upgradu DB je nutné, stejně jako u instituce, vytvořit prvotní pokladnu a převést pod ní všechny existující pokladní doklady. Jako pokladník se nastaví ten uživatel, který je během upgradu veden jako vedoucí jídelny.

3.8.2 Nastavení

Nastavení prvotní pokladny se převezme z údajů, které jsou v nastavení systému a odtud se následně odstraní.

3.8.3 Vypnuté přihlášení

Při vypnutém přihlašování se bude zobrazovat pokladna vedená pod vedoucím jídelny.

3.8.4 Vazby na ostatní třídy

Třída pokladna má vazby na následující seznam tříd.

- OsobaJidelny
- Instituce, 3.4.2.4

3.8.4.1 Vazba na uživatele

Tato vazba určuje pokladníka odpovědného za danou pokladnu. Pokladník může vybírat a vydávat peníze za aktivity, ovšem pouze za ty, které reálně může vidět viz 3.5.3.

3.8.5 Viditelnost

Pokud je uživatel správcem momentálně přihlášené instituce, pak uvidí všechny pokladny této instituce. Pokud správcem není, uvidí pouze ty pokladny, na kterých je uveden jako pokladník.

3.8.6 Dopad na uživatele

Jelikož bude možné mít více pokladen, je nutné vytvořit nové seznamové okno, přes které se bude uživatel do pokladny dostávat. To mění typický průchod uživatele skrz grafické uživatelské rozhraní. Dokud byla pokladna jenom jedna fixně pro celý program, mohla být ihned otevřena.

3.8.6.1 Správce instituce

Pokud je uživatel správcem instituce, nelze obejít seznamové okno, jelikož bude zodpovědný za správu všech pokladen, bude mít možnost tyto pokladny vytvářet a další.

3.8.6.2 Ostatní uživatelé

Pokud uživatel je pokladníkem jedné pokladny, pak lze otevřít rovnou tuto pokladnu a tím tento uživatel nepocítí jakoukoliv změnu práce se systémem. Pokud má ovšem více pokladen, nelze se seznamovému oknu vyhnout.

Kapitola 4

Realizace

4.1 Entity obecně

Problémová doména vyžaduje, aby byly entity mezi sebou poměrně závažně propojené. Toto má dopad na některé základní aspekty implementace.

4.1.1 Načítání z DB

Je nutné, aby maximální většina komplexních atributů byla z databáze načítána líně, neboli až když se reálně potřebují, ne ihned. Propojení je natolik vysoké, že natažení jednoho řádku by velmi pravděpodobně skončilo natažením skoro celé databáze evidence plateb, pokud by se líně nenačítalo.

Navíc toto nejde triviálně nechat na frameworku, jelikož GUI je odstíněno od databázové transakce. Je tedy nutné před předáním dat GUI vědět, kde budou data využita a které položky je nezbytné donačíst.

Zdá se tedy dobré vyvinout jednotný přístup pro donatažení těchto atributů nezávisle na daném rozhraní dané třídy. Co se entit týče, v rámce projektu existují pouze tři možné typy atributů. Tyto atributy jsou:

- jednoduché (celočíselné, řetězec...)
- jiné entity
- kolekce

4.1.1.1 Jednoduché atributy

Všechny tabulky projektu mají velmi nízký počet sloupců, je tedy naprosto zbytečné tyto atributy označovat jako líně donačítané. Naopak by toto označení vedlo k delším časovým prodlevám a vyšší zátěži na procesoru a paměťovém médiu.

4.1.1.2 Jiné entity

Tyto entity mohou mít složitější strukturu a již se vyplatí je označovat jako líně donačítané. Všechny entity perzistované v databázi implementují rozhraní IPokystovatelID, jehož jedinou metodou je získání id. Pokud je instance momentálně nenačtena a je zavolána tato metoda, Hibernate automaticky data načte a naplní instanci.

4.1.1.3 Kolekce

Kolekce je často velmi dobré donačítat líně, dopředu nemusí být vůbec jasné, kolik řádků tato kolekce bude mít nebo kolik dalších donačtení může v rámci entity proběhnout. Co se týče Hibernate, jakmile se na kolekci zavolá metoda na získání její velikosti, jsou automaticky donačteny všechny instance.

4.1.1.4 Souhrn

Za použití Java Reflection API[12] lze donačtení všech líných atributů poměrně jednoduše zařídit. Kód 4.1 na stránce 35 je ukázkou, jak implementace vypadá. Nutný požadavek, aby tento přístup fungoval je:

- gettry na proměnné
  - veřejné
  - bez parametru
  - dle Javovské jmenné konvence

Je nutno podotknout, že toto řešení pracuje na jedné úrovni entity a nijak se nezanořuje. Není však žádný problém toto řešení použít opakovaně.

4.1.2 Hibernate proxy

Hibernate proxy třídy jsou třídy, které pomáhají s líným načítáním atributů během transakce. Hibernate dynamicky vytvoří třídy, které rozhraním odpovídají namapované třídě. Toto velmi usnadňuje práci s líně načteným atributem, ovšem také to přináší některé problémy.

4.1.2.1 Porovnávání

NetBeans a jeho automaticky vygenerovaná metoda na porovnání instancí třídy nefunguje při porovnání namapované třídy a proxy třídy, jelikož je v této vygenerované metodě přímé porovnání tříd objektů. Toto se dá snadno obejít za použití porovnávání na implementaci stejného rozhraní.

```
public static void fetch( IPoskytovatelId entita, Map<String,
    Boolean> fetchMap ) {
  if( entita == null || fetchMap == null ) {
    return;
  }

  for( Entry<String, Boolean> entry : fetchMap.entrySet() ) {
    if( entry.getValue() ) {
      fetch(get(entita, entry.getKey()));
    }
  }
}

private static Object get( IPoskytovatelId entita, String key )
    {
  try {
    Method method = entita.getClass().getMethod("get" + key.
    substring(0, 1).toUpperCase() + key.substring(1));
    return method.invoke(entita);
  } catch (NoSuchMethodException | SecurityException |
    InvocationTargetException | IllegalArgumentException |
    IllegalAccessException ex) {
    throw new RuntimeException(ex);
  }
}

private static void fetch( Object object ) {
  if( object != null ) {
    if( object instanceof IPoskytovatelId ) {
      ((IPoskytovatelId) object).getId();
    } else if( object instanceof Collection ) {
      ((Collection) object).size();
    } else {
      throw new RuntimeException();
    }
  }
}
```

Ukázka kódu 4.1: Použití Java Reflection API pro spuštění auto. donačtení

4.1.2.2 Podtřídy

Co se podtříd týče není v rámci transakce možné proxy instanci přetypovat na rozhraní podtřídy. Jsou možné dva přístupy řešení.

- manuálně nařídit změnu proxy na instanci třídy projektu
- změna rozhraní

Pokud je instance třídy HibernateProxy je možné ji na tuto třídu přetypovat a získat implementaci danou projektem. Toto ovšem nutí programátora použít přímo třídy Hibernate, pomocí JPA tohoto nelze dosáhnout. Nezdá se to tedy jako iděální řešení.

Pokud logika objektů a informací, které se programátor snaží z podtřídy dostat, dovoluje metody rozhraní pro získání těchto informací přesunout do rozhraní supertřídy, pak Hibernate, pokud nutno, automaticky získá danou implementaci z projektu. Tento způsob nepoužívá žádné speciální implementace a kód je velmi čistý.

V rámci projektu byl zvolen druhý přístup. Ukázkový příklad: počítání ceny aktivity. Cena jednorázové aktivity a periodické aktivity se liší ve výpočtu. Jednorázová aktivita má jednu fixní cenu. Periodická aktivita má také jednu fixní cenu, ovšem tato cena je násobena počtem period. Lze tedy v předkovy přidat metodu rozhraní na výpočet ceny, která bude v potomcích implementována dle lokální logiky. Takto není v těchto případech nutné přetypovávat instance tříd a tedy problém s proxy vůbec nenastane.

4.2 Databáze

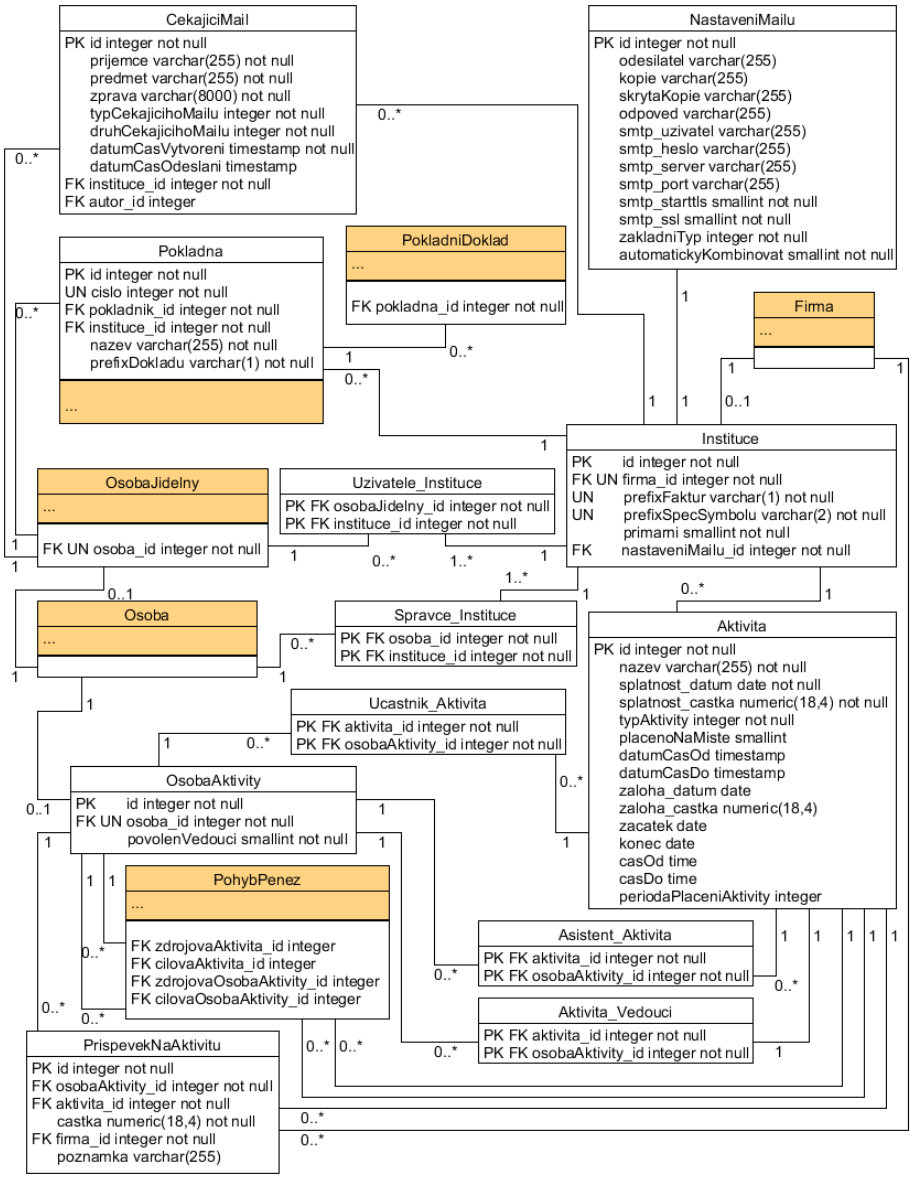
Následující sekce se bude zabývat převodem třídního modelu na model databázový. DB model je k náhlédnutí na obrázku 4.1 na stránce 37.

4.2.1 Vnoření tříd

Třídy KontrolniBod a NastaveniSMTP nebudou samostatné tabulky. Místo toho budou vestavěny v tabulkách na které by odkazovaly. Toto řešení ušetří mnoho zbytečných databázových propojení během dotazů.

4.2.2 Vazby v DB

V rámci databáze se dají specifikovat cizí klíče. Za pomoci tohoto mechanizmu lze tabulky provázat několika různými způsoby.



Obrázek 4.1: DB model

4.2.2.1 Vazby 1:1

Tyto vazby jsou implementovány přes spojovací sloupec, na který je přidáno omezení unikátnosti. Díky tomuto omezení je zaručeno, že každý jeden řádek jedné tabulky může mít pouze jeden jiný řádek.

4.2.2.2 Vazby 1:N

Většina těchto vazeb bude implementována přes spojovací sloupec v rámci jedné z tabulek.

Výjimkou je vedoucí aktivity, který ačkoliv je vazbou 1:N bude realizován přes spojovací tabulku. V budoucnu je možné, že bude potřeba aktivitě nastavit na pozici vedoucího více než jednu osobu. Díky tomuto řešení bude tato změna poměrně nenáročná.

4.2.2.3 Vazby M:N

Tyto vazby budou v rámci celého projektu realizovány přes spojovací tabulku.

4.2.3 Vyjmenované typy

Tyto typy budou implementovány přes celočíselné číslo, které bude odpovídat ordinálnímu číslu vyjmenovaného typu.

4.2.4 Dědičnost

V rámci této práce se objevuje pouze jedna třída s podtřídami a to je AAktivita. Podtřídy JednorazovaAktivita a PeriodickaAktivita nemají hodně atributů navíc. Dále je možné, že některé atributy momentálně evidované pouze u jedné podtřídy, např. placeno na místě, bude nutné evidovat u obou podtříd. Z těchto důvodů je pro implementaci těchto tříd zvolena realizace jednou tabulkou, která bude obsahovat sloupce sjednocení obou.

4.2.5 Role

Role mezi osobou aktivity a aktivitou jsou navrženy tak, že každá tato role je jedna samostatná propojovací tabulka. Toto je velmi jednoduché a průhledné řešení. Navýšení počtu rolí nebo diference v rámci jedné role do určité stromové struktury není předpokládáno. Tudíž je zvolené řešení dostačující.

4.2.6 Validace dat

Validace správnosti dat posílaných do databáze probíhá pouze v rámci GUI a servisních tříd. Databáze kontroluje jenom primární klíče, cizí klíče, unikátnost a nulovost.

4.3 Mapování

S připravenou databází a modelem je nutné všechny tyto věci dostat do prostředí Javy. V této podkapitole jsou rozebrány některé základní pojmy a jejich použití v práci.

Řádky tabulek jsou mapovány na klasické Javovské třídy, které mají akorát bezparametrový konstruktor a gettry a settry na své atributy.

4.3.1 Vazby v Javě

Co se týče javy, vazby zmíněné v 4.2.2 lze namapovat pomocí JPA anotací. V závislosti na požadavcích řešení pak lze ještě rozhodnout, jestli má daná vazba být jednosměrná nebo obousměrná.

4.3.1.1 OneToOne

Tato JPA anotace se používá pro mapování jedné entity na jednu jinou entitu, viz 4.2.2.1. Např. osoba aktivity má jednu osobu a naopak osoba má maximálně jednu osobu aktivity.

4.3.1.2 OneToMany a ManyToOne

Tyto JPA anotace se používají při mapování vazby 1:N viz 4.2.2.2 v závislosti na fakt, ze které strany se entita na vazbu dívá. Např. aktivita má ManyToOne vedoucího (mnoho aktivit na jednoho vedoucího), naopak vedoucí má OneToMany vedené aktivity (jeden vedoucí na mnoho aktivit).

4.3.1.3 ManyToMany

Tato JPA anotace se využije při mapování vazby M:N viz 4.2.2.3. Např. aktivita může mít mnoho účastníků a stejně tak na druhé straně osoba aktivity může mít mnoho aktivit, kterých se účastní.

4.3.1.4 Jednosměrné

Jednosměrné vazby jsou takové vazby, které jsou mapovány jenom z jedné strany. Např. instituce má namapovanou firmu v sobě, ale firma v sobě namapovanou instituci nemá. V mnohých případech není nutné, aby byla vazba dostupná z obou objektů.

4.3.1.5 Obousměrné

Obousměrné vazby jsou takové vazby, které jsou mapovány z obou stran. Např. osoba aktivity má namapované aktivity, kterých se účastní a stejně tak na druhé straně má aktivita namapované účastníky.

Co se Hibernate týče, obousměrné vazby jsou spravované pouze z jedné strany. Tzn. že jednoduchá úprava v rámci jednoho objektu během transakce je promítána do db, ale jednoduchá úprava v rámci objektu druhé strany promítnuta není. Ideální je upravit vztahy z obou stran a zachovat tak správnou vnitřní strukturu logiky. Pokud toto není třeba, např. při konci transakce instance, které se dál nebudou používat, stačí upravit pouze tu stranu, která je spravovaná.

To, která strana je spravovaná závisí na faktu, ve které třídě je uvedeno mapování. Např. u aktivity je u množiny účastníků anotace JoinTable se specifikací, jak se mají tabulky spojit. Z této strany je závislost spravována. Ze strany osoby aktivity je pouze u ManyToMany parametr mappedBy, který ukazuje na atribut aktivity mapující druhou stranu této vazby. Z této strany vazba spravovaná není.

4.3.2 Mapování dědičnosti

Namapování dědičnosti na jednu sjednocenou tabulku zmíněné v 4.2.4 se dá poměrně jednoduše zařídit pomocí JPA anotací.

4.3.2.1 Inheritance

Tato anotace specifikuje u předka jaká dědičnost na dané entitě proběhne. V případě jedné sjednocené tabulky se použije strategie InheritanceType.SINGLE_TABLE.

4.3.2.2 DiscriminatorColumn

Tato anotace specifikuje u předka, který sloupec a jeho hodnota se mají použít pro určení jaké specifické podtřídy řádek je. Co se Javy týče, rozlišovací atribut je enum. Tzn. že discriminatorType anotace je nutno nastavit na DiscriminatorType.INTEGER. Pak již stačí doplnit jméno sloupce.

4.3.2.3 DiscriminatorValue

Tato anotace je použita ve specifické podtřídě, v každé jedna, a určuje jakou přesnou hodnotu má sloupec upřesněný pomocí DiscriminatorColumn mít, aby byl tento řádek namapaván na tuto podtřídu. V tomto případě se pro JednorazovaAktivita použije „0“ a pro PeriodickaAktivita „1“. Tyto hodnoty odpovídají ordinálním hodnotám, neboli pořadí indexovanému od nuly, enumu použitého pro typ aktivity.

4.3.3 Přenos dat mezi aplikací a databází

V rámci komunikace s databází je použito několik anotací. Jejich popis a využití viz níže.

4.3.3.1 Service

Springovská anotace, která určuje dle[13] samostatnou funkcionalitu představovanou nějakým rozhraním.

Takto jsou označeny třídy, které komunikují s databází, provádí její čtení a zápis a starají se o správné naplnění bysnys logiky. Co se týče kontextu, každá tato třída má pouze jednu instanci.

4.3.3.2 PersistenceContext

Anotace JPA, která dle[14] specifikuje závislost na kontextem spravovaném manažeru entit.

Díky této anotaci se dá jednoduše vložit manažer entit do kterékoliv třídy. Výhradně použito v rámci servisních tříd.

4.3.3.3 Autowired

Tato Springovská anotace specifikuje dle[15], na které pole, metodu aj. má zafungovat dependency injection.

V servisních třídách se tato anotace nejčastěji používá pro získání ostatních servisních tříd.

4.3.3.4 Transactional

Springovská anotace která automaticky zařídí vytvoření transakce při vstupu do metody a její commit či rollback v závislosti na výsledku.

Výhradně použito v servisních třídách na metodách, které potřebují komunikovat s databází.

4.4 Logická vrstva

Veškerá byznys logika se ukrývá v servisních třídách. Tyto třídy v sobě mají manažera entit pro komunikaci s databází a jejich metody jsou volány z grafického uživatelského rozhraní, více o napojení na databázi viz 4.3.3 na stránce 40.

4.4.1 Rozdělení

Každá entita, která je spravována manažerem entit, má svoji servisní třídu. Tato servisní třída se stará o veškerou logiku spojenou s touto entitou.

4.5 GUI

Implementace grafického uživatelského prostředí proběhla za pomoci standardních prvků používaných v projektu. Fakticky se jednalo o přepoužití stá-

vajících prvků, rozšíření a změny stávajících prvků a vytvoření nových prvků z prvků již existujících.

Všechny grafické prvky jsou vytvořeny za pomoci NetBeans IDE GUI Builder[3]. Tzn. že kus Java kódu, vytvoření, nastavení a pozicování všech grafických prvků, takto vytvořených tříd je generován dle XML souboru s příponou form, který je vytvářen naklikáním v designeru. Tento přístup odpovídá již existujícím prvkům v projektu.

Co se týče tabulek, většina z nich obsahuje kontextové menu, které do určité míry kopíruje funkcionalitu panelu akcí v daném okně. Toto opět odpovídá standardnímu chování ostatních částí projektu.

Implementace uvedeným způsobem přímo splňuje nefunkční požadavek „Standardní GUI“ viz tabulka nefunkčních požadavku 2.2 na stránce 16.

4.5.1 Internacionalizace

Pro internacionalizaci se používají soubory formátu klíč hodnota. Tyto klíče jsou pak použity v GUI pro vyhledání potřebné hodnoty. Nicméně aplikace používá pouze jednu množinu těchto souborů a to v češtině.

4.5.2 Tisky

Pro tvorbu nových tisků bylo použito standardních hlavičkových šablon, které se vyskytují v rámci celé aplikace. Tento přístup přímo splňuje nefunkční požadavek „Standardní tisky“ viz tabulka nefunkčních požadavků 2.2 na stránce 16.

4.5.3 Anotace Springu

V rámci GUI se používá několik anotací Springu. Jejich výčet a hlavní použití je v následujících sekcích.

4.5.3.1 Component

Efekt této anotace dle[16] je, že Spring si vytvoří instanci, jedináčka pokud není specifikováno jinak, a ta lze v rámci kontextu použít.

Anotace je využita pro všechny hlavní okna. To většinou znamená okna seznamová a detailová.

4.5.3.2 Configurable

Efekt této anotace dle[17] je, že pokud se kdekoliv v programu vytvoří nová instance takto oanotované třídy, Spring upraví bytekód tak, že na nich pak může provest dependency injection a další.

Tato anotace je použita hlavně pro prvky, které samostatně nemohou existovat a jsou součástí něčeho většího, např. panely.

```
public void setPrispevekNaAktivitu(IPrispevekNaAktivitu
    prispevekNaAktivitu) {
  IPrispevekNaAktivitu old = this.prispevekNaAktivitu;
  this.prispevekNaAktivitu = null;
  firePropertyChange("prispevekNaAktivitu", old, null);

  this.prispevekNaAktivitu = prispevekNaAktivitu;
  firePropertyChange("prispevekNaAktivitu", null,
    prispevekNaAktivitu);
}
```

Ukázka kódu 4.2: Setter pro binding instance vlastní třídy

4.5.3.3 Autowired

Popis anotace Autowired viz 4.3.3.3.

Anotace je zejména použitá pro využití tříd komunikujících s databází v rámci GUI. Často využíváné funkce jsou např.: načti seznam, ulož položku atd.

4.5.4 Naplnění GUI daty

Drtivá většina vyplnění GUI daty probíhá pomocí nastavení vázání mezi grafickými prvky okna a vázanou instancí. V rámci seznamových oken je vázan seznam prvků, v rámci detailového okna je vázan prvek jeden. Toto vázání je možné snadno naklikat v NetBeans IDE GUI Builder[3].

Při použití vlastních nestandardních grafických prvků je nutné tyto prvky doplnit o možnost vázaní. Toto lze udělat v rámci předpřipraveného podkladu gettrem a settrem na vázanou třídu, kde setter pálí událost změny vlastnosti.

Pokud se v rámci vázání nejedná o základní typ, ale třídu, složenou z více atributů je nutné upravit setter ještě více. V rámci zpracování vypálené události o změně se kontroluje, jestli momentální stav není stejný jako stav nový. Ovšem pokud se změní vnitřek instance, reference momentálního objektu a nového objektu jsou stejné. Možné vypracování viz ukázka kódu 4.2 na stránce 43.

4.5.5 Validace dat

Validace probíhá podobně jako v již existujících oknech projektu přesně tak, jak jsou na to uživatelé zvyklí.

Kompletní validace dat probíhá po zavolání uložení v daném detailovém oknu. U některých složitějších detailových oken probíhá tato validace částečně již během specifických akcí. Např. kontrola konta/příspěvků při odstranění účastníka aktivity v okně detailu aktivity.

4.5.6 Dlouho trvající operace

Dlouho trvající operace jsou zabaleny pod spuštění SwingWorkeru a jejich výpočet je tedy spuštěn mimo EDT na vlákně v pozadí. Takováto implementace zajišťuje naprosté splnění nefunkčního požadavku „Výpočty mimo EDT“ viz tabulka nefunkčních požadavků 2.2 na stránce 16.

4.5.7 Duplikace „jedináčků“

Jelikož jsou všechna okna v rámci Springu zavedeny jako Component 4.5.3.1, existují primárně v kontextu jako jediná sdílená instance. V některých případech je ovšem nutné, aby těchto oken bylo víc. Pokud se tedy jedná o seznamové okno s možností výběru prvků, je do kontextu vpravena ještě jedna instance, která je vyvolávána při výběru.

Kapitola 5

Testování

Hlavní obavou, kromě samotné funkcionality, byla v rámci projektu časová náročnost často prováděných operací. Problémová doména vyznačuje vysokou mírou propojenosti a prakticky každý dotaz má možnost se rozpadnout na další podmnožinu dotazů. Toto by při nevhodné implementaci mohlo způsobit dlouhé časové prodlevy, kdy by uživatel čekal na výsledek akce. Toto je jev nežádoucí a byl tedy testován.

5.1 Testování funkčnosti aplikace

Testování správnosti funkcionality se provádělo výhradně manuálně. Projekt sice fakticky obsahuje automatizované testy, ale tyto testy nejsou již několik let používané a celé testovaní v projektu je vypnuto. Nové automatizované testy byly tedy kvůli velikosti nutných úprav kódu vynechány.

V rámci manuálního testování byl odzkoušen běh aplikace při simulovaném normálním provozu.

5.1.1 Funkční požadavky

Co se funkčních požadavků týče viz tabulka 2.1 na stránce 15, tabulka 5.1 na stránce 46 zobrazuje všechny tyto požadavky a způsoby jejich splnění.

Většina požadavků lze ověřit jednoduchým průchodem grafickým uživatelským prostředím.

5.1.2 Nefunkční požadavky

Většina nefunkčních požadavků je úspěšně splněna zvoleným způsobem implementace viz 4.5 na stránce 41. Dle tabulky nefunkčních požadavků 2.2 na stránce 16 pak zbývá pouze jeden požadavek a tím je „Minimální dopad“. Tento požadavek byl rozebírán na místech změny chování programu a ve všech

Tabulka 5.1: Test požadavků

Název	Splněno
Evidence institucí	GUI: Systém → Nastavení → Institucí
Evidence aktivit	GUI: Školní platby → Aktivity
Evidence účastníků	GUI: Školní platby → Aktivity → detail → Účastníci
Evidence příspěvků	GUI: Školní platby → Aktivity → detail → Příspěvky
Evidence plateb	GUI: Školní platby → Aktivity → detail → Platby
Asistenti aktivity	GUI: Školní platby → Aktivity → detail → Asistenti
Účastnící aktivity	GUI: Školní platby → Aktivity → detail → Účastníci
Platba účastníka	GUI: Školní platby → Aktivity → detail → Účastníci → Platby účastníka
Platební opozdilci	GUI: Školní platby → Aktivity → detail → Účastníci → Tisk → Zaplacenost
Urgence	GUI: Školní platby → Osoby aktivit → Urgovat nedoplatky
Platební opozdilci tisk	GUI: Školní platby → Aktivity → detail → Účastníci→ Tisk→ Zaplacenost
Aktivity uživatele	Vidno přehlašováním v GUI: Školní platby → Aktivity
Detail aktivity tisk	GUI: Školní platby → Aktivity → detail → Účastníci → Tisk → Seznam
Unikátnost SS	SS je složen z pevného prefixu, unikátního prefixu instituce a unikátního čísla ativity v rámci instituce. Tzn. je nutně unikátní.
Aktivita v minulosti	GUI: Školní platby → Aktivity → detail → Ulož
Zodpovědnost	GUI: Školní platby → Aktivity → detail → Ulož
Málo účasníků	GUI: Školní platby → Aktivity → sloupec Počet účastníků
Účast zodpovědných	GUI: Školní platby → Aktivity → detail → Ulož
Více SMTP	Každá instituce má svoje. GUI: Systém → Nastavení → Institucí → detail → Nastavení mailů
Bilance aktivity tisk	GUI: Školní platby → Aktivity → Tisk → Bilance aktivit

instancích bylo usouzeno, že je dopad buď malý nebo nevyhnutelný. Dá se tedy říct, že i tento požadavek je splněn.

5.1.3 Souhrn

Aplikace splňuje veškeré funkční i nefunkční požadavky, které byly vyžadovány během projektu.

5.2 Testování rychlosti

V rámci testování rychlosti se soustředilo hlavně na často opakované operace uživatele. U často používaných operacích velmi záleží na rychlosti jejich výpočtu, uživatelé často nechtějí čekat nepřiměřeně dlouho. Operace, které se využívají málo nebo jednou, nejsou v testování rychlosti vůbec zahrnuty.

5.2.1 Testovací prostředí

Mezi jednotlivými testy byla aplikace vypínána a zapínána, aby se anuloval dopad Javovské inherentní optimalizace provedené v rámci předchozího testu na test momentální.

Rychlosti byly testovány na počítači s následujícími parametry. Při testování byla použita lokální databáze.

- Windows 10 (64-bit)
- Java 8 Update 152 (64-bit)
- 16GB RAM
- Procesor Intel® Core™ i7-4500U
  - Počet jader: 2
  - Počet vláken: 4
  - Základní frekvence: 1.8GHz
  - Maximální turbo frekvence: 3GHz
- SK Hynix SH920 2.5 7MM 256GB (SSD)

5.2.2 Výsledky testování

V tabulce 5.2 na stránce 48 jsou zobrazeny výsledky testování často prováděných uživatelských operací.

Tabulka 5.2: Výsledky testování rychlosti operací

Test	Rozšiřující informace	Délka operace	
		1. spuštění	průměr 5 spuštění
Uložení nové aktivity	0 asistentů, 0 účastníků, 0 příspěvků	61ms	23,8ms
Uložení nové aktivity	20 asistentů, 20 účastníků, 20 příspěvků	96ms	59,0ms
Uložení nové aktivity	1440 asistentů, 1440 účastníků, 1440 příspěvků	1906ms	1651,0ms
Načtení aktivity	0 asistentů, 0 účastníků, 0 příspěvků	52ms	24,6ms
Načtení aktivity	20 asistentů, 20 účastníků, 20 příspěvků	44ms	42,0ms
Načtení aktivity	1440 asistentů, 1440 účastníků, 1440 příspěvků	1560ms	1404,6ms
Tisk účastníků	0 účastníků	2532ms	818,4ms
Tisk účastníků	20 účastníků	2554ms	828,4ms
Tisk účastníků	1440 účastníků	4431ms	1341,6ms
Načtení plateb	0 plateb	20ms	6,2ms
Načtení plateb	20 plateb	80ms	46,0ms
Načtení plateb	1576 plateb	1063ms	584,8ms
Tisk plateb	0 plateb	2293ms	743,4ms
Tisk plateb	20 plateb	2795ms	822,2ms
Tisk plateb	1576 plateb	2660ms	904,8ms

5.2.2.1 Výsledky komunikace s DB

Rozdíl mezi délkami trvání prvního spuštění a následujících spuštění není nijak velký. Jednoduchým pohledem na časy, lze odhadnout, že složitost ukládání i načítání je lineární. Přesně taková by složitost těchto operací měla být. Všechny časy z pohledu čekání uživatele jsou přijatelné a to i ty, které jsou výsledkem nesmyslně velkých dat.

5.2.2.2 Výsledky tisků

Ze všech výsledků je velmi dobře vidět, že první vytvoření samotného tisku spotřebuje velmi mnoho času. První spuštění je několikrát náročnější než spuštění následující. Je nutné podotknout, že první spuštění je započítáno do průměru pěti spuštění, což znamená, že následující čtyři spuštění byla o hodně

rychlejší než průměr uvedený.

Většina výpočetního času během tisků se spotřebuje na inicializaci samotného tisku. A to dokonce i pro tisky dat, které se reálně pravděpodobně nikdy neobjeví, jako jsou tisk přehledu aktivity s 1440 účastníky a tisk přehledu 1576 plateb. Všechny tyto hodnoty časů jsou přijatelné a to i ty, které jsou výsledkem nesmyslně velkých dat.

5.2.3 Vytížení klíčových prostředků

Za klíčové prostředky s ohledem na testovanou aplikaci jsou považovány CPU a disk.

5.2.3.1 Vytížení CPU

Vytíženost byla zkoumána za pomocí Task Manageru, který je standardní součástí OS Windows. Během všech testů, které byly dostatečně dlouhé, aby se ukázaly na grafu, bylo vytížení procesoru maximální.

5.2.3.2 Vytížení disku a odezva DB

Za pomoci Task Manageru nebylo možné vytíženost disku rozumně pozorovat, jelikož se neustále pohybovala kolem 0%. Toto je pravděpodobně způsobeno příliš dlouhým intervalem přírůstku grafu.

Hibernate poskytuje možnost generování statistik všech prováděných JDBC dotazů. Důležité hodnoty pro toto testování jsou počet vygenerovaných dotazů, doba přípravy těchto dotazů a doba provádění. Výsledky těchto statistik pro testy z tabulky 5.2 na stránce 48 jsou obdobné a stejně tak jsou podobné i závěry z nich vyvozené, z tohoto důvodu je zmíněn pouze jeden příklad, načtení aktivity s 1440 asistenty, 1440 účastníky a 1440 příspěvky. Statistika tohoto případu je v následujícím seznamu.

- počet dotazů : 47
- příprava dotazů: 158318761 ns
- provedení dotazů: 29120980 ns

Co se týče 47 dotazů, základní velikost dávky je nastavena na 100. Počet dotazů tedy odpovídá předpokladům.

Provedení dotazů nepřesahuje 30ms, což je o mnoho méně než čas uvedený v tabulce 5.2 na stránce 48, který je průměrně 1651ms. Většina času výpočtu tedy netkví v získání dat z DB, ale ve vytvoření Javovských instancí a jejich propojení. HDD by musel být více jak čtyřicetkrát pomalejší, aby se rychlost načtení vyrovnala rychlosti zpracování Javou. Toto je příliš velký rozdíl, než aby ho způsobil pouze disk. Dá se tedy vyvodit, že úzkým místem v systému je procesor.

5.2.4 Souhrn

Z výsledků vyplývá, že časové prodlevy testovaných akcí jsou adekvátní.

Závěr

Cílem této práce bylo rozšíření stávající aplikace Jídelna 2017, kterou poskytuje firma Barda SW, HW s.r.o. o možnost evidence studentských aktivit, plateb za tyto aktivity a celkové usnadnění a z části automatizace této práce.

Analýza, návrh a kompletní implementace rozšíření proběhly v řádném termínu k spokojenosti jak zadavatele tak vedoucího práce. Cíle bylo v rámci práce dosaženo a tedy rozšíření aplikace i tato diplomová práce se dají prohlásit za úspěšné.

Námět pro další vývoj

Z důvodu uvedení institucí do programu je momentálně celkový pohled na přístup programu poněkud nejednotný. Některé části aplikace jsou vždy stejné nezávisle na přihlášené instituci. Z pohledu logiky problému by bylo jistě dobré, kdyby další vývoj aplikace směřoval ke kompletnímu přesunu k modelu institucí.

Literatura

[1] BARDA SW, HW s.r.o. software pro jídelny. [online]. [cit. 2017-12-12]. Dostupné z: http://www.barda.cz/

[2] Welcome to NetBeans. [online]. [cit. 2017-12-12]. Dostupné z: https://netbeans.org/

[3] Designing a Swing GUI in NetBeans IDE - Tutorial. [online]. [cit. 2017-12-12]. Dostupné z: https://netbeans.org/kb/docs/java/quickstart-gui.html

[4] Jaspersoft® Studio | Jaspersoft Community. [online]. [cit. 2017-12-20]. Dostupné z: https://community.jaspersoft.com/project/jaspersoft-studio

[5] UMLet - Free UML Tool for Fast UML Diagrams. [online]. [cit. 2017-12-19]. Dostupné z: http://www.umlet.com/

[6] Git. [online]. [cit. 2017-12-12]. Dostupné z: https://git-scm.com

[7] Java Powers Our Digital World. [online]. [cit. 2017-12-11]. Dostupné z: https://go.java/index.html

[8] Hibernate. Everything data. - Hibernate. [online]. [cit. 2017-12-11]. Dostupné z: http://hibernate.org/

[9] JasperReports® Library | Jaspersoft Community. [online]. [cit. 2017-12-17]. Dostupné z: https://community.jaspersoft.com/project/jasperreports-library

[10] Spring Framework. [online]. [cit. 2017-12-19]. Dostupné z: https://projects.spring.io/spring-framework/

[11] Bakaláři - mezi školou a rodinou | Bakaláři. [online]. [cit. 2017-12-20]. Dostupné z: https://www.bakalari.cz/

[12] java.lang.reflect (Java Platform SE 8). [online]. [cit. 2017-12-19]. Dostupné z: https://docs.oracle.com/javase/8/docs/api/java/lang/reflect/package-summary.html

[13] Service (Spring Framework 5.0.2.RELEASE API). [online]. [cit. 2017-12-19]. Dostupné z: https://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/stereotype/Service.html

[14] PersistenceContext (Java(TM) EE 7 Specification APIs). [online]. [cit. 2017-12-19]. Dostupné z: https://docs.oracle.com/javaee/7/api/javax/persistence/PersistenceContext.html

[15] Autowired (Spring Framework 5.0.2.RELEASE API). [online]. [cit. 2017-12-19]. Dostupné z: https://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/beans/factory/annotation/Autowired.html

[16] Component (Spring Framework 5.0.2.RELEASE API). [online]. [cit. 2017-12-19]. Dostupné z: https://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/stereotype/Component.html

[17] Configurable (Spring Framework 5.0.2.RELEASE API). [online]. [cit. 2017-12-19]. Dostupné z: https://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/beans/factory/annotation/Configurable.html

Příloha **A**

Seznam použitých zkratek

GUI Graphical User Interface

ORM Object-relational mapping

DB Database

SRPŠ Sdružení rodičů a přátel školy

IDE Integrated development environment

SMTP Simple Mail Transfer Protocol

ST Servisní technik

SS Specifický symbol

EDT Event dispatching thread

UML Unified Modeling Language

JPA Java Persistence API

API Application Programming Interface

XML Extensible Markup Language

CPU Central Processing Unit

JDBC Java Database Connectivity

SSD Solid State Drive

HDD Hard Disk Drive

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `download.txt` .... textový soubor s odkazem na web pro stažení aplikace
- `src`
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF