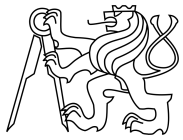
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Webový systém daňové evidence
Student:	Martin Hudský
Vedoucí:	Ing. David Buchtela, Ph.D.
Studijní program:	Informatika
Studijní obor:	Informační systémy a management
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2016/17

Pokyny pro vypracování

Cílem práce je návrh a implementace webového systému daňové evidence (jednoduchého účetnictví) pro podnikatele v ČR.

1. Seznamte se s problematikou a legislativním zázemím daňové evidence (dříve jednoduchého účetnictví) pro podnikatele v ČR.
2. Proveďte analýzu funkčních i nefunkčních požadavků na evidenční systém s ohledem na legislativní požadavky a očekávané změny v legislativě.
3. Navrhněte a implementujte webovou aplikaci (Apache, PHP, MySQL).
4. Navrženou aplikaci řádně otestujte a nasaďte ve zvolené firmě po dohodě s vedoucím práce.
5. Proveďte ekonomickou analýzu a zhodnocení přínosů systému pro podnikatele v ČR.

Seznam odborné literatury

Dodá vedoucí práce.

L.S.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 7. října 2015

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Bakalářská práce

Webový systém pro vedení daňové evidence

Martin Hudský

Vedoucí práce: Ing. David Buchtela, Ph.D.

30. června 2016

Poděkování

Děkuji především Ing. Davidu Buchtelovi za jeho rady při vývoji celého webového systému. Dále bych chtěl poděkovat rodině, která mě podporovala během celé doby studia.

Prohlášení

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

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

V Praze dne 30. června 2016 .....

Odkaz na tuto práci

Hudský, Martin. *Webový systém pro vedení daňové evidence.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

Abstrakt

Cílem této práce je zanalyzovat, navrhnout, implementovat a otestovat webový systém pro vedení daňové evidence pro podnikatele v České republice. Hlavní výhoda je responzivita obsahu. Implementace je založena na spojení webových technologií PHP, MySQL a Apache.

Klíčová slova Webový systém, daňová evidence, responzivní návrh

Abstract

The goal of this bachelor thesis is to analyze, design, implement and test web system for single entry book-keeping in Czech republic. Main advantage is a responsive design. Impementation is based on these web technics: PHP, MySQL and Apache.

Keywords Web system, single entry book-keeping, responsive design

Obsah

Seznam obrázků

Úvod

Cíl práce

Cílem práce je analyzovat, navrhnout, implementovat a otestovat webovou aplikaci, která bude svým obsahem responzivní k zařízením s různou velikostí displeje z důvodu, aby uživatel nebyl omezen využitím aplikace pouze z konkrétního typu zařízení.

Struktura práce

V 1. kapitole seznámím s právním řádem podnikání v České republice a proč by uživatel měl tuto aplikaci používat.

V další části popíšu, jaké řešení v současné době existují a jaké mají výhody či nevýhody.

V následující kapitole navrhnu požadavky, které budou kladeny na systém. Předmětem 4. kapitoly bude návrh systému tak, aby odpovídal požadavkům, které byly definovány v kapitole č. 3.

V další kapitole bude nastíněno, jak probíhala implementace a budou popsány nástroje, které byly během vývoje využity.

V předposlední části tohoto dokumentu budou popsány postupy, kterými bude výsledná aplikace otestovaná a bude vyhodnocen výsledek z testování.

A v poslední části bude zhodnocen celkový vývoj aplikace a možnosti, které jsou pro aplikaci do budoucna.

Kapitola 1

Podnikání v České republice

Důvod podnikání je docela jednoduchý. Osoba podniká za jediným účelem a to především kvůli zisku. Téměř každý podnikatel má ještě před začátkem podnikání nějakou svoji vizi či nápad, kterým by chtěl vyplnit místo na trhu. Samotné podnikání lze podle [1] definovat takto:

„Jedná se o soustavnou činnost provozovaná podnikatelem samostatně, vlastním jménem, na vlastní odpovědnost a za účelem dosažení zisku“

1.1 Druhy podnikání

Pokud osoba chce začít podnikat, tak podle zákonů v České republice musí podnikat na základě určitého oprávnění. Podnikat lze několika způsoby. Od nejjednoduššího způsobu jako fyzická osoba se živnostenským listem až po obchodní společnosti. Každá forma podnikání má své výhody a nevýhody a tedy je vhodné se při začátku podnikání zamyslet, která z těchto forem je nejvhodnější pro daný obchodní záměr.

1.1.1 Fyzická osoba

V případě podnikání jako fyzická osoba může vykonávat svoji obchodní činnost na základě:

- živnostenského oprávnění
- jiného oprávnění, které spadá pod zvláštní předpisy. Jedná se například o lékaře, advokáty či daňové poradce, kteří podnikají na základě členství v profesních komorách

Mezi hlavní výhody podnikání jako fyzická osoba patří především:

- jednoduché založení – některé živnostenské oprávnění jsou tzv. „volné“, není potřeba žádné odborné vzdělání a živnostenské oprávnění je vydáno na základě ohlášení.

- nízké pořizovací náklady spojené se založením – v případě registrování prvního živnostenského oprávnění je v současné době zpoplatněno částkou 1 000 Kč, každé další 500 Kč.
- není vyžadován žádný základní kapitál.
- rozhodování je v rukou jednoho vlastníka.
- při ročním obratu do 25 mil. Kč možnost vést daňovou evidenci.
- možnost uplatnit výdaje paušálem, nutné evidovat pouze příjmy.

Mezi hlavní nevýhodou podnikání jako fyzická osoba je především fakt, že podnikatel ručí celým svým majetkem. Jednoduchým příkladem je, že pokud dluží například 400 000 Kč, ale v obchodním majetku má prostředky dostupné pouze v hodnotě 250 000 Kč, tak zbylých 150 000 Kč je možné vymáhat například pomocí zabavení osobního majetku podnikatele.

1.1.2 Právnická osoba

Pod tímto pojmem si většinou představíme obchodní společnost, pod kterou mohou spadat fyzické či další právnické osoby (společníci). Mezi právnické osoby patří především:

- veřejné obchodní společnosti.
- komanditní obchodní společnosti.
- společnost s ručením omezeným.
- akciová společnost.

Mezi nejrozšířenějším typem obchodní společnosti je společnost s ručením omezením, ve které je od roku 2014 nutný minimální základní kapitál pouze 1 Kč a kde nikdo ze společníků neručí svým osobním majetkem.

Mezi hlavní nevýhody právnické osoby jsou vysoké zřizovací náklady, které mohou sahat až částce okolo 15 000 Kč, a složitější evidence během celého účetního období.

1.2 Evidence podnikatelské činnosti

Ať se podnikatel rozhodne pro jakoukoliv formu podnikání, tak jeho činnost je nutné evidovat formou daňové evidence nebo účetnictví. Daňovou evidenci, která je jednodušší, mohou vést pouze fyzické osoby, které:

- nejsou zapsány v obchodním rejstříku.
- jejich roční obrat nepřesáhl 25 milionů korun.

Hlavním rozdílem mezi daňovou evidencí a účetnictvím je u výpočtu výsledku hospodaření (základu daně). V daňové evidenci se rovná rozdílu příjmů výdajů, u účetnictví se vypočte jako rozdíl mezi výnosy a náklady.

1.3 Daňová evidence

Daňová evidence (dříve jednoduché účetnictví) má jediný cíl a tím je zajistit správné podklady pro výpočet základu daně z příjmů. Forma a způsob vedení daňové evidence není přesně stanovena. Zákonem dané položky, které je nutno doložit, jsou informace o:

- příjmech a výdajích v potřebném členění pro zjištění základu daně, tedy rozdělení příjmů a výdajů na ty, které jsou zahrnované do základu daně a které nikoliv.
- majetku a závazcích podnikatele.

Její zákonná úprava je v zákonu o daních z příjmů § 7b [2].

Kapitola 2

Analýza

Analýza patří mezi důležitou část vývoje jakéhokoliv softwaru a je jedno, zda se jedná o malý či velký projekt. Na základě analýzy je potřeba zjistit, co daná aplikace má umět.

V následujících odstavcích budou rozepsány funkční a nefunkční požadavky, které jsou kladeny na systém, model případu užití a ilustrace doménového modelu, který byl vytvořen na základě vyhodnocené analýzy.

2.1 Funkční požadavky

Funkční požadavky obsahují vše, co by měl výsledný systém umět. Na základě nich se již při analýze dají psát testy, díky nimž bude možné systém otestovat proti právě těmto požadavkům.

2.1.1 Vytvoření dokladů a jejich uložení

Jedná se o nejpodstatnější část systému. Ze zákona je nutné evidovat příjmy a výdaje, které nastanou během podnikání. Tedy vytváření dokladů by mělo být velmi jednoduché, přehledné a rychlé.

2.1.2 Editace a případné mazání dokladů

V případě, že ve vytvořeném dokladu je chyba, tak je potřeba, aby byla možnost doklad vymazat/editovat.

2.1.3 Vytváření dokladů na základě jiného dokladu

Kvůli zrychlení vytváření dokladů je kladen požadavek na vytváření dokladů na základě jiného již zaevidovaného dokladu. Jednoduchý příklad. Máme již vytvořenou příjemku se skladovými položkami, při následném vytvoření faktury je možné propojit příjemku s fakturou a díky tomu budou ve faktuře obsaženy veškeré položky z příjemky s možností přidat další položky.

2.1.4 Filtrování faktur, pokladních dokladů a dokladů k DPH pomocí data vytvoření

Při vybírání případně vyhledávání položek je vhodné je omezit například datem, či viditelností v seznamech.

2.1.5 Vedení skladové evidence

Z důvodu inventarizace je vhodné vést přehlednou skladovou evidenci, která na konci roku poskytne informace o stavech jednotlivých skladových položek.

2.1.6 přihlášení uživatele

Jelikož je předpoklad, že aplikace bude nasazena na vzdáleném serveru, tak je nutná autorizace. Systém bude navržen pouze pro jednoho uživatele systému.

2.2 Nefunkční požadavky

Nefunkční požadavky jsou vlastnosti a omezení, které jsou kladeny na výsledný systém. Je možné v nich definovat omezení, které se týkají například odezvy systému, spolehlivosti či jednoduchosti.

2.2.1 Přístup webové aplikace ze zařízení nezávislé velikosti displeje

Hlavní záměrem vývoje této aplikace přizpůsobení velikosti displeje zařízení, ze kterého je do aplikace přistupováno. V případě změny (například přetočením tabletu z klasického módu do landscape) se aplikace okamžitě přizpůsobí bez nutnosti obnovení stránky případně přesměrování na jinou adresu aplikace.

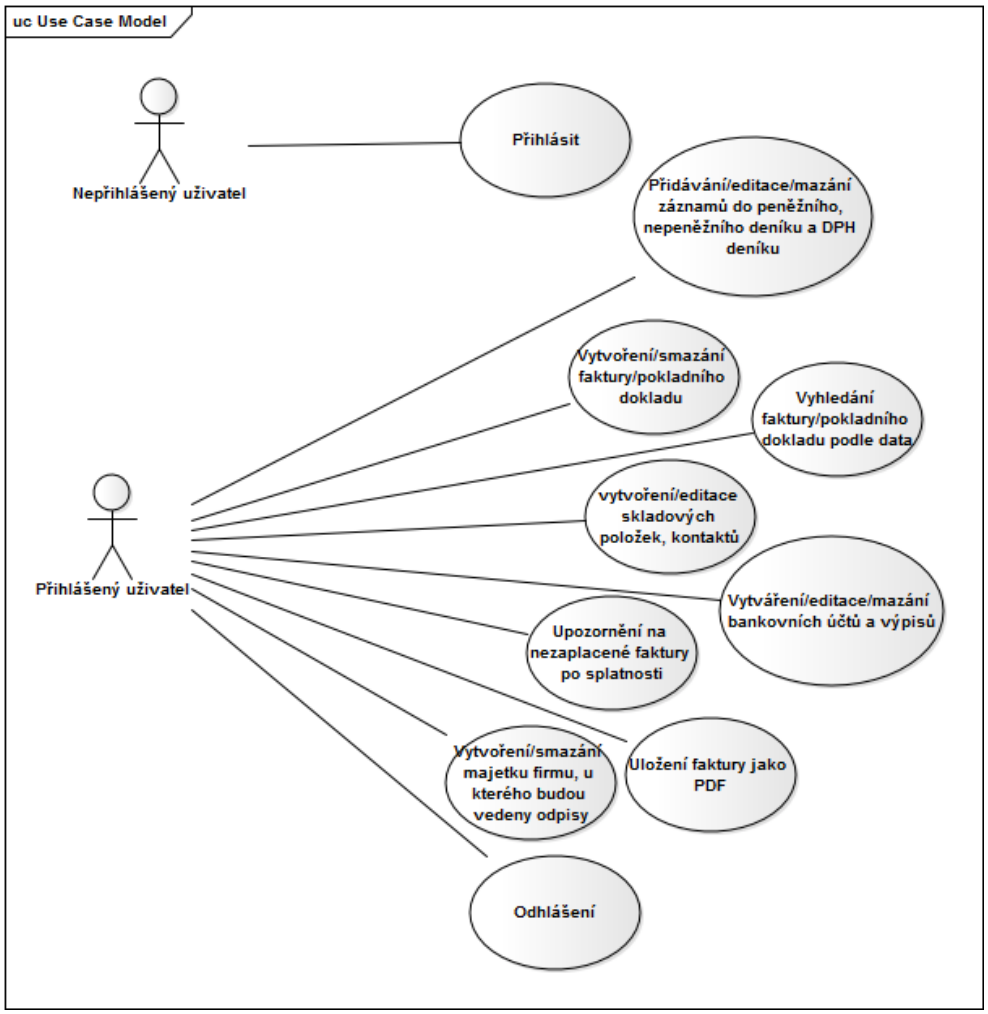
2.2.2 Uživatelsky přívětivý design

Běžně znalý uživatel bude schopen se orientovat v aplikaci po několika průchodech.

2.2.3 Nezávislost na konkrétním operačním systému

Jelikož se jedná o webovou aplikaci, tak jediným spojením s aplikací je moderní internetový prohlížeč. Konkrétně budou podporovány tyto:

- Google Chrome (verze 45.0.0 a vyšší)
- Firefox (verze 43 a vyšší)
- Internet Explorer (verze 9 a vyšší)
- Safari(verze 8.0 a vyšší)



Obrázek 2.1: Model případy užití

2.2.4 Validace a zpracování dat

Cílem je zabezpečit, aby do databáze nemohly být uloženy a dále zpracovány data, které neodpovídají zákonem stanoveným požadavkům. [3]

2.3 Případy užití

Diagram případu je jeden z prvních diagramů, který se vytváří při analýze. Úkolem diagramu je nastínit základní funkcionalitu systému, ale nezobrazuje, jak tato funkcionalita bude implementována.

Případy užití analyzovaného systému vystihuje obrázek.2.1

2.3.1 Přihlášení uživatele

Po spuštění aplikace je nepřihlášený uživatel vyzván k zadání uživatelského jména a hesla, kterým se přihlásí do aplikace. V případě neúspěchu je uživatel upozorněn a má možnost zadat přihlašovací údaje znovu.

2.3.2 Přidávání/editace/mazání záznamů do peněžního, nepeněžního deníku a DPH deníku

Pro vytvoření dokladu se otevře formulář podle zvoleného typu dokladu. V případě pokusu o uložení nevalidních dat bude uživatel upozorněn a požádán o opravu nesprávně zadaných údajů. V případě úspěšné kontroly dat bude uživatel informován o úspěšně provedené operaci. V případě mazání se otevře stejný formulář s již předvyplněnými daty, které jsou načteny z databáze. Po úspěšně provedené editaci bude uživatel informován. Mazání záznamů bude možné z okna pro editaci položky. Před samotným smazáním bude uživatel vyzván k potvrzení smazání položky.

2.3.3 Vytvoření/smazání faktury/pokladního dokladu

Akce vyvolá formulář, který bude mít již předvyplněné ID dokladu, které bude odlišné v závislosti na vytvářeném dokladu. Po vyplnění položek formuláře budou všechny položky zvalidovány. V případě správného vyplnění všech povinných položek proběhne přesměrování, kde bude možnost přidání skladových položek ze seznamu zaevidované v systému, či případné propojení s již zaevidovanou příjemkou/výdejkou. V této chvíli bude možné fakturu buďto zrušit či uložit.

2.3.4 Vyhledání faktury/pokladního dokladu podle data

Při zobrazení požadovaného dokladu se zobrazí veškeré zaevidované doklady, které bude možné filtrovat parametry *od* a *do*. V případě existence požadovaných dokumentů budou vypsány na obrazovku.

2.3.5 Vytvoření/editace skladových položek, kontaktů

Případ užití začíná vyvoláním požadovaným formulářem pro vytvoření položky. Položka bude zvalidována a následně uložena do databáze pro použití v ostatních dokladech. V položkách bude možné editovat jednotlivé hodnoty atributů a také možnost zobrazení/skrytí položky při výběru v jiných dokladech.

2.3.6 Vytváření/editace/mazání bankovních účtů a výpisů

Podobné jako 2.3.3, kde při přidávání položek výpisu bude možné přidat položky z nezaplacených faktur.

2.3.7 Uložení faktury jako PDF

Po vyhledání bude možnost uložit fakturu na pevný disk jako dokument typu application/pdf.

2.3.8 Vytvoření/smazání majetku firmu u kterého budou vedeny odpisy

Dále bude možné evidovat a mazat majetek ve firmě (postup vytváření a mazání podobný viz. 2.3.5). U každé majetku bude možné vypočítat odpis následujícího roku. Tento odpis bude systémem ověřen, zda je podle legislativních zákonů možné uplatnit v požadované výši. Pokud ne, tak aplikace uživatele upozorní o maximální výši odpisu a uživatel bude moci svůj požadavek upravit. Pokud zadá již správnou hodnotu, tak systém odpis zaeviduje k majetku a uživatel bude mít možnost zaevidovat tento odpis do nepeněžního deníku.

2.3.9 Upozornění na nezaplacené faktury po splatnosti

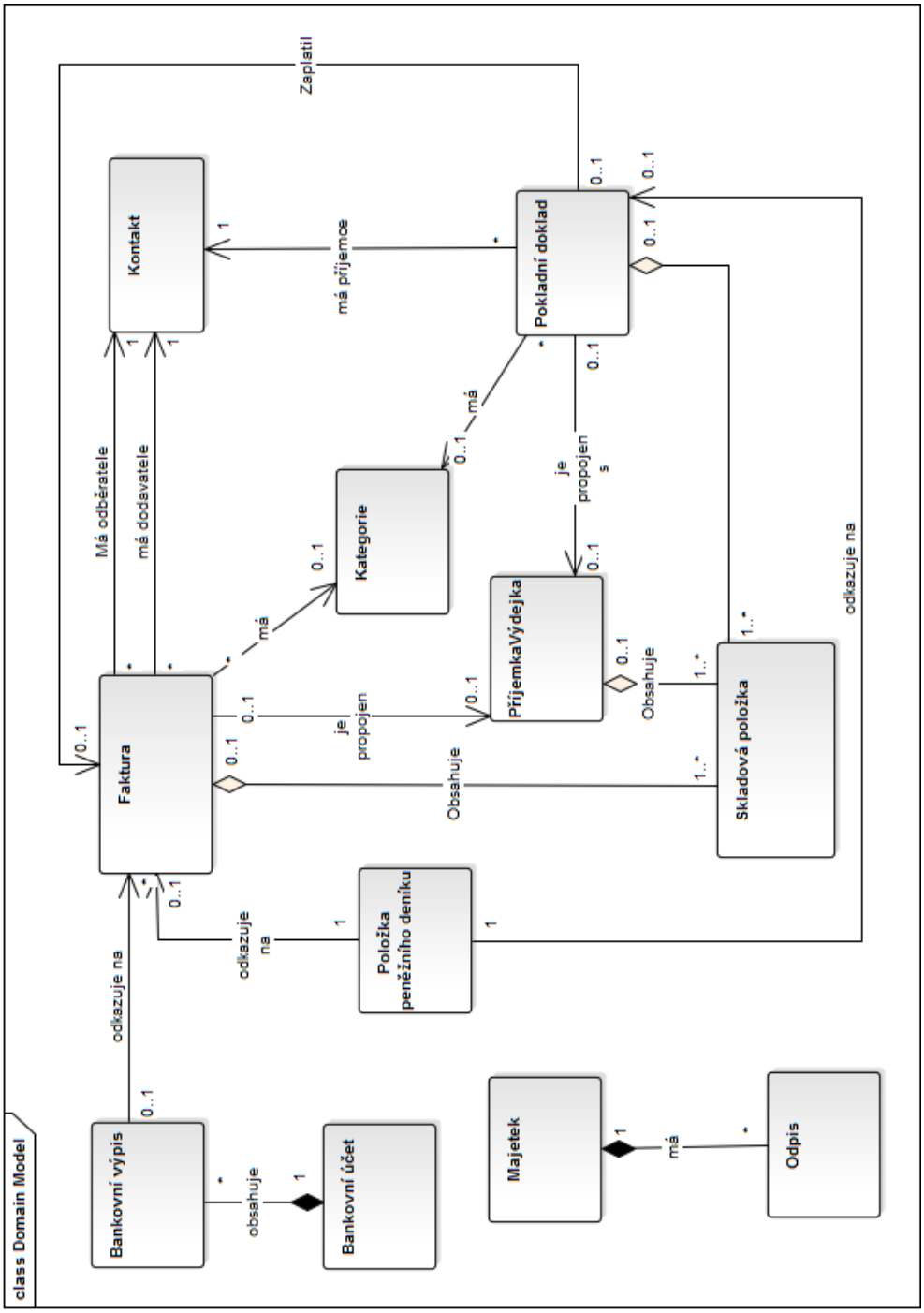
V případě, že některé faktury nebudou zaplacené po době splatnosti faktury, tak bude uživatel informován na základní obrazovce na tzv. homepage.

2.3.10 Odhlášení

Uživatel se bude moci odhlásit ze kteréhokoliv stavu aplikace. Touto akcí se dostane do stavu 2.3.1.

2.4 Doménový model

Doménový model je jeden ze základních diagramů při analýze SW. V tomto diagramu jsou zakresleny entity (objekty), které je potřeba evidovat a u kterých jsou zobrazeny vzájemné vazby. Jedná se o velice zjednodušený model aplikace, který bude rozšířen v diagramu tříd v sekci návrh 4. Pro zpřehlednění diagramu 2.2 jsem záměrně vynechal atributy.



Obrázek 2.2: Doménový model výsledné aplikace

Kapitola 3

Rešerše

Dostupných SW pro vedení daňové evidence je mnoho. Především se jedná o desktopové účetní SW, ve kterém je možnost vést daňovou evidenci, což není příliš přehledné, jelikož aplikace nabízí mnoho funkce, které pro daňovou evidenci nejsou vůbec potřeba.

Je tedy potřeba rozdělit SW pro evidenci do několika skupin podle způsobu použití.

3.1 Vedení papírové evidence

Vedení papírové evidence v dnešní době je poněkud zastaralé a nepřehledné. V případě chyby jsou složitější opravy, než u dokladů elektronických. Nutné vše manuálně vyplňovat a přepočítávat, kde hrozí další možnost udělat chybu.

3.2 Vedení v elektronické v podobě tabulkových editorů

Oproti papírové evidence odpadá nevýhoda manuálního počítání, ovšem velikou nevýhodou je nepřehlednost, která roste s přibývajícími zaevidovanými doklady. Další nevýhodou je nutnost sdílení souborů při potřebě editace na více pracovních stanicích například pomocí e-mailů, kde může nastat problém se synchronizací záznamů.

Mezi hlavními zástupci je například tabulkový editor Excel od firmy Microsoft[4], případně tzv. svobodný editor Calc z balíku LibreOffice[5].

3.3 Software pro pracovní stanice

V případě zaměření SW pro pracovní stanice (desktop) je výběr lepší. Na českém trhu je možnost pořídit několik druhů komplexních SW pro vedení

účetnictví. To je podle mě jeden z problémů nepříliš velkého využití při vedení daňové evidence. Licence za SW pro vedení účetnictví bývá velmi vysoká, v případě použití na více pracovních stanicích s využitím SQL databáze poskytovatele se hodnota šplhá až k 30 000 Kč.

Mezi významnější zástupce této kategorie patří program Pohoda[6] v několika verzích, případně ekonomický software Stereo[7].

3.4 Mobilní software

Na poli mobilního softwaru jsem nenalezl prakticky žádné aplikace pro vedení daňové evidence pro mobilní zařízení. Momentálně není vyloženě běžné vést evidenci pomocí mobilního telefonu malých rozměrů. Ale v případě tabletů (které se berou většinou zařazují také mezi mobilní zařízení) už tento problém nevidím.

Mezi zástupci existují aplikace iDoklad[8] či mPohoda[9], které umožňují vést pouze přijaté a vystavené firemní faktury.

3.5 Webové aplikace

Webových aplikace není také mnoho. Většinou i v nich (jako v mobilním SW) je možnost evidovat pouze faktury, které je možnost sesynchronizovat s mobilní aplikací.

Zástupce v této kategorii je například aplikace iÚčto [10], kde hlavními nedostatky je neresponzivní design aplikace – nelze použít v mobilním prohlížeči a pak další nedostatek je nemožnost vést skladovou a majetkovou evidenci.

3.6 Závěr

Mezi dostupnými aplikace jsem nenašel žádné vhodné řešení, které by využívalo možnosti responzivní webové aplikace vhodnou pro vedení daňové evidence. Pouze mezi desktopovými aplikacemi je možné najít vhodné kandidáty, kde hlavní nevýhodou mohou být pro podnikatele vysoké licenční poplatky (z důvodu, že se jedná o aplikace pro kompletní vedení účetnictví). Proto se tito podnikatelé raději spokojí s jednodušší formou evidence a právě na tyto obchodníky se má zaměnit Webový systém pro vedení daňové evidence.

Kapitola 4

Návrh

4.1 Použité technologie

Ještě před samotným vývojem je nutné zvolit vhodné technologie a nástroje, díky kterým výsledný SW splní funkční požadavky. Se špatně zvolenou technologií je možné se dostat v projektu do stavu, kdy není možné splnit veškeré požadavky specifikace a je tedy na rozhodnutí, zda začít od začátku na základě jiných technologií, či omezit funkčnost výsledné aplikace.

4.1.1 PHP

PHP[11] je skriptovací programovací jazyk, který mění statické webové stránky na dynamické. PHP skripty jsou zpracovány na serveru, který po přijetí požadavku na daný obsah zpracuje skript a poté odešle odpověď žadateli ve formě HTML kódu.

4.1.2 MySQL

MySQL[12] je tzv. relační databáze, ve které jsou data uložena v tabulkách, kde každý řádek tabulky má atributy. Jedná se o typovanou databázi, tedy každý sloupec má přesně daný typ. Každá položka (řádek) v tabulce musí mít unikátní identifikátor, pomocí kterého lze vyhledat požadovaný záznam.

4.1.3 Apache

Apache[13] server je nejrozšířenějším webovým serverem, který slouží k obsluze jednotlivých uživatelů. Hlavní výhodou je dostupnost pro všechny hlavní platformy a především fakt, že je vyvíjen jako open-sourced (volně šiřitelný zdrojový kód).

4.1.4 Symfony2 a Doctrine2

PHP framework Symfony2[14], který je v současné době jeden z nejlepších a nejrychleji rozšiřujícím frameworkem pro webové technologie. Symfony je open-sourced a podporuje velké množství druhů databází. Jedná se o aplikační framework, který vychází z návrhové vzoru MVC. Architektura MVC dělí aplikaci na 3 logické části tak, aby je šlo upravovat samostatně a výsledný dopad byl na ostatní části co nejmenší. Tyto tři části jsou:

- Model,
- Viewer
- Controller

Model reprezentuje data a business logiku aplikace, View zobrazuje uživatelské rozhraní a Controller má na starosti tok událostí v aplikaci a obecně aplikační logiku.

Jednou ze základních komponent Symfony2 je Doctrine2[15], která má na starost mapování objektů do klasických relačních databází, jako jsou například Oracle[16] nebo MySQL pomocí anotací v kódu viz. zjednodušený příklad v sekci implementace 5.2.1.

4.1.5 jQuery

Jquery[17] je jedna z nejpoužívanějších a nejrozšířenějších javascriptových knihoven a to především díky jednoduché syntaxi, jednoduché výběry pomocí CSS [**?**] selektorů a také díky optimalizaci pro většinu moderních webových prohlížečů.

4.1.6 Bootstrap

Bootstrap[18] je tzv. front-end framework, který zpřístupňuje několik webových technologií jako jsou HTML[**?**], CSS. Je navržený tak, aby jeho zobrazení bylo responzivní, tedy aby byl web přizpůsobivý pro zařízení všech velikostí displejů z jedné webové adresy.

4.1.7 JTable

JTable[19] je jQuery doplněk (plugin), který má na starost zobrazení tabulkových dat v několika formátech v závislosti na velikosti displeje daného zařízení.

4.1.8 wkHTMLtoPDF

WK<html>toPDF[20] je utilita, která pomocí příkazové řádky vytvoří z HTML stránky dnes velmi používaný PDF dokument. Jedna se o open-source software, který lze použít běžných systémech jako jsou Linux, Windows a Mac.

4.2 Diagram tříd

Diagram tříd (ang. Class diagram) je již velice detailní model výsledné aplikace, který je vázán na konkrétní programovací jazyk. Na rozdíl od doménového modelu diagram obsahuje metody jednotlivých tříd. Pro přehlednost jsem v diagramu 4.1 vynechal metody typu *get* a *set* pro nastavení třídních atributů.

4.2.1 Položka peněžního deníku

Třída představuje jeden záznam v peněžním deníku. Položky je možno zadávat manuálně nebo v případě vytvoření bankovního výpisu či zaevidování pokladního dokladu se záznam vytvoří automaticky na základě těchto dokladů.

4.2.2 Skladová položka

Třída **SkladovaPolozka** reprezentuje zaevidovanou skladovou položku ve firmě. Obsahuje atributy:

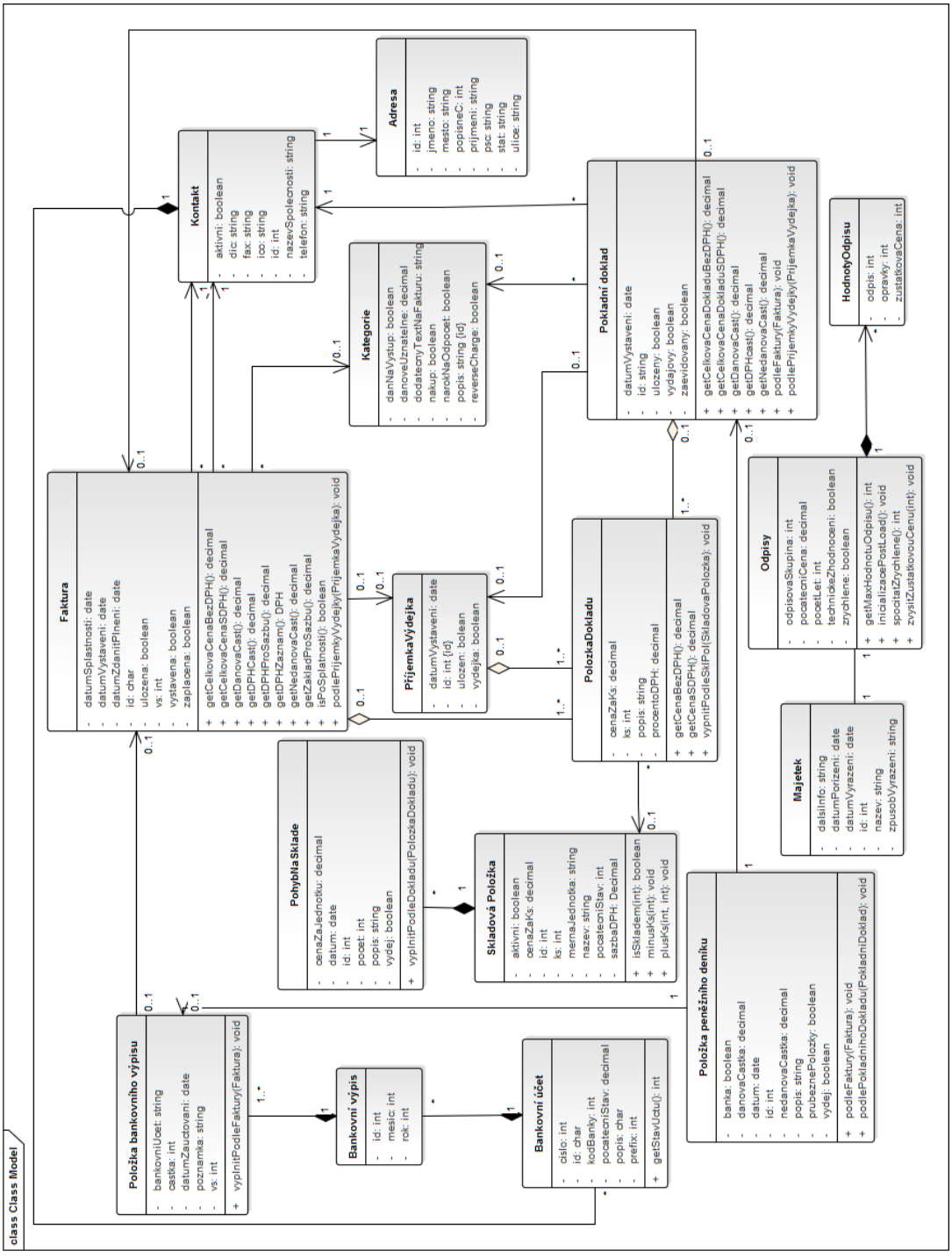
- *aktivni*: *boolean* hodnota, zda se skladová položka má zobrazovat v seznamu položek.
- *sazbaDPH*: sazba DPH pro položku. V případě, že uživatel aplikace není nastaven jako Plátce DPH, má tento atribut hodnotu *null*.

Metody:

- *isSkladem*(*int*): vrátí *boolean* hodnotu v závislosti na tom, zda je množství, předávané v parametru funkce, skladem.
- *minusKs*(*int*): sníží množství na skladě.
- *PlusKs*(*int,int*): při zvýšení množství na skladě se na základě dvou parametrů *novaCena* a *p*rirustek vypočítá nová skladová cena pomocí principu váženého aritmetického průměru.

4.2.3 Historie pohybu na skladě

Třída **PohybNaSklade** úzce souvisí se skladovou položkou, jelikož se jedná o historii příbytků a výdejů u dané položky ze skladu. Třída eviduje u všech položek množství, cenu, popis a datum při každé změně, která na skladě nastane.



Obrázek 4.1: Diagram tříd pro výslednou aplikaci

4.2.4 Položka dokladu

Třída **PolozkaDokladu** reprezentuje položky těchto dokladů: faktury, pokladního dokladu a příjemky/výdejky.

- *skladovaPolozka*: instance třídy **SkladovaPolozka**, nepovinný atribut. Pokud je vyplněn, tak v případě vytváření dokladu typu **PrijemkaVydejka** je na základě tohoto atributu upraveno skladové množství.

4.2.5 Kontakt

Třída **Kontakt** obsahuje údaje o odběratelích a příjemcích ve fakturách a příjemce v případě pokladního dokladu. Obsahuje tyto atributy:

- *aktivni*: boolean hodnota, v případě, že je *true*, tak je kontakt viditelný v seznamu odběratelů/dodavatelů.
- *id*: automaticky generovaná číselná hodnota.
- *ico*: je povinné, 8 ciferné identifikační číslo kontaktu, validované.
- *dic*: není povinné, 8–12 znakové daňové identifikační číslo, kde první dva znaky značí zemi plátce DPH.
- *telefon, fax* nepovinné, validované.
- *adresa*: instance třídy **Adresa**.

4.2.6 Adresa

Jedná se o povinný atribut třídy **Kontakt**. Třída **Adresa** má povinné atributy *jmeno, prijmeni, ulice, popisneC, psc, mesto* a nepovinný atribut *stat*. Atribut *id* je automaticky generovaná číselná hodnota.

4.2.7 Pokladní doklad

Třída realizuje evidování příjmových a výdajových pokladních dokladů. Má tyto atributy:

- *vydajovy*: na základě tohoto atributu je doklad rozeznán jako příjmový či výdajový pokladní doklad.
- *id*: automaticky generovaná hodnota při vytváření dokladu. Má tvar *xPDrrmmyyyy*, kde *x* je V/P podle toho, zda je jedná o výdajový/příjmový pokladní doklad, *rr* a *mm* je dvojčíslí aktuální roku a měsíce a *yyyy* je pořadové číslo daného dokladu.

- *prijemkaVydejka*: pokladní doklad je propojen s instancí třídy **PrijemkaVydejka**. Pokladní doklad obsahuje stejné položky jako tato příjemka/výdejka a může obsahovat další. Pokud je atribut nevyplněný, tak lze podle tohoto pokladního dokladu vytvářet příjemky/výdejky v případě, že alespoň některá položka pokladního dokladu obsahuje odkaz na instanci třídy **SkladovaPolozka**.
- *faktura*: nepovinný atribut, instance třídy **Faktura**, jedná se o případ, kdy je faktura zaplacená pokladním dokladem.
- *penezniDenik*: instance třídy **PenezniDenik**, odkazuje na položku v peněžním deníku. V případě vyplnění se bere pokladní doklad jako zaevidovaný v peněžním deníku.
- *kategorie*: instance třídy **Kategorie**, pokud je vyplněn, tak se na základě kategorie počítá daňově uznatelná část a DPH.
- *polozky*: kolekce třídy **PolozkaDokladu**. Minimálně obsahuje 1 položku.
- *ulozen*: nepovinný atribut typu *boolean*, při vytvoření má hodnotu *false*, pokud se do vytvoření dalšího dokladu nezmění na *true*, tak se neuložený doklad smaže.

Metody:

- *getCelkovaCenaDokladuBezDPH*(): vrátí celkovou hodnotu pokladního dokladu bez DPH.
- *getCelkovaCenaDokladuSDPH*(): vrátí celkovou hodnotu pokladního dokladu včetně DPH.
- *getDanovaCast*(): vrátí hodnotu daňově uznatelné části dokladu.
- *getNedanovaCast*(): vrátí hodnotu daňově neuznatelné hodnoty doklady.
- *podleFaktury*(*faktura*): vyplní doklad podle parametru typu **Faktura**.
- *podlePrijemkyVydejky*(*prijemkaVydejka*): vyplní doklad podle parametru typu **PrijemkyVydejky**.

4.2.8 Faktura

Třída představuje 2 typy faktur, které může uživatel evidovat – přijaté a vystavené. Podobné jako **PokladniDoklad**. Zabývat se budu pouze rozdíly:

- *id*: automaticky generovaná hodnota ve tvaru. *FAxrrmmyyyy*, kde x je V/P podle toho, zda se jedná o fakturu vystavenou/přijatou, *rr* a *mm* je poslední dvojčíslí roku a měsíce data vystavení a *yyyy* je pořadové číslo dokladu.

- *vystavena*: atribut typu *boolean*, rozlišuje mezi vystavenou a přijatou fakturu.
- *datumVystaveni*: datum vystavení faktury.
- *datumSplatnosti*: datum spatnosti faktury, automaticky přednastaveno jako *datumVystaveni* + 30 dní.
- *vs*: variabilní symbol faktury.
- *odberatel*: instance třídy **Kontakt**, v případě přijaté faktury automaticky nastaven jako společnost přihlášeného uživatele.
- *dodavatel*: instance třídy **Kontakt**, v případě vystavené faktury automaticky nastaven jako společnost přihlášeného uživatele.

Metoda, která je podstatná u faktur:

- *isPoSpatnosti*(): návratová hodnota typu *boolean* podle toho, zda je faktura po splatnosti.

4.2.9 Příjemky a výdejky

Třída **PrijemkyVydejky** reprezentuje výdejky a příjemky v evidenci. Představují mezičlánek mezi skladem a fakturami/pokladními doklady. Pouze při zaevidování příjemky/výdejky se bude měnit stav zásob na skladě. Na základě již vytvořených příjemek a výdejek lze vytvářet pokladní doklady/faktury a opačně.

- *id*: automaticky generované ve tvaru *xrrmmyyyy*, kde *x* je P/V podle toho, zda se jedná o příjemku či výdejku, *rr* a *mm* je dvojčíslí aktuálního roku a měsíce a *yyyy* je pořadové číslo dokladu.
- *polozky*: kolekce třídy **PolozkaDokladu**, kde každá instance má vyplněnou *skladovouPolozku*. Po přidání položky se automaticky změní množství na skladě.
- *faktura,pokladniDoklad*: pokud má vyplněn alespoň jeden z atributů, tak již není možné podle instance této třídy vytvořit fakturu či pokladní doklad a zároveň není možné vyplnit příjemku/výdejku položkami z další faktury či pokladního dokladu.

4.2.10 Bankovní účet

Třída **BankovniUcet** slouží k evidovaní více bankovních účtů k jednotlivým kontaktům. Lze k nim vytvářet výpisy, díky nímž je možné platit faktury. Má pár základních atributu a to:

- *cislo*: povinný atribut, číslo bankovního účtu, validovaný podle [21]
- *kod*: povinný atribut, kód banky, 4 – 7 místné číslo
- *prefix*: nepovinný atribut, předčíslí bankovního účtu, validovaný podle [21]
- *id*: vytvořené z předešlých 3 atributů: *prefix-cislo-kod*
- *bankovniVypisy*: kolekce instancí třídy **BankovniVypis**

4.2.11 Bankovní výpis

Obsahuje měsíc, rok a odkazuje na bankovní účet, pro který je vytvořen. Následně obsahuje položky bankovního výpisu. Jedná se o spojový objekt mezi bankovním účtem a položkami bankovního výpisu.

4.2.12 Položka bankovního výpisu

Třída obsahuje koncové položky, které budou následně evidovány do peněžního deníku. V případě výběru ze seznamu nezaplacených faktur se informace předvyplní automaticky.

- *id*: automaticky generovaná číselná hodnota.
- *cisloUctu*: stringová hodnota, číslo účtu na který/ze kterého přišla platba, při výběru faktury se nabídnou č. účtů, které jsou k dané faktuře vázány přes objekt typu **Kontakt**.
- *castkaDan*: kladná či záporná daňová částka.
- *castkaNedan*: kladná či záporná nedaňová částka.
- *datumZauctovani*: povinný atribut, datum zaevidování platby na účtu.
- *vs*: nepovinný atribut, variabilní symbol platby.
- *poznamka*: nepovinný atribut.

4.2.13 Majetek

Třída **Majetek** slouží k evidenci dlouhodobého majetku firmy, který je do nákladů rozpouštěn formou ročních odpisů. Třída obsahuje tyto atributy:

- *id*: automaticky generovaná hodnota typu string, která má formát *DMrr-mmxxx*, kde *rr* a *mm* je dvojčíslí roku a měsíce, kdy byl majetek zaevidován a *xxx* je pořadové číslo zaevidovaného majetku.

- *datumVyrazeni*: je vyplněno, pokud byl majetek vyřazen z evidence. Pokud je vyplněn, bere se majetek jako vyřazený a není zobrazen při výpisu aktivního majetku. Pokud je majetek vyřazen není zobrazena možnost k němu přidávat další odpisy, ani přidávat již zaregistrované odpisy do nepeněžního deníku.
- *odpisy*: instance třídy **Odpisy**.

4.2.14 Odpisy

Třída Odpisy představuje ucelený objekt pro evidování a výpočet odpisů majetku.

- *odpisováSkupina*: číslo odpisové skupiny, do které majetek patří.
- *hodnotaOdpisu*: kolekce instancí třídy **HodnotaOdpisu**, která představuje strukturu (odpis, oprávky, zůstatková cena) k jednomu odpisu majetku.
- *zrychlene*: atribut značící, zda se majetek bude odepisovat klasickým způsobem či zrychleně.
- *pocetLet*: počet let, po kolik byl majetek již odepsán (v případě zvýšené vstupní ceny se tato hodnota nuluje).

Metody:

- *getMaxHodnotuOdpisu*() vrátí maximální hodnotu následujícího odpisu. Metoda použita při ověření, zda hodnota požadovaného odpisu uživatele není vyšší, než maximální hodnota.
- *inicializacePostLoad*() na základě odpisové skupiny a druhu odpisů načtené z databáze se v této funkci nastaví další potřebné atributy, které jsou potřeba pro výpočet odpisů. Tato funkce je spuštěna automaticky po načtení objektu z databáze.
- *zvysitZustatkovouCenu*() v případě zvýšení ceny majetku po jeho zaevidování je zvýšena zůstatková i počáteční cena.

Kapitola 5

Vývoj

5.1 Použité nástroje

Žádná etapa při vývoji SW se neobejdete bez podpůrných nástrojů. Ať už jde při analýze nebo návrhu o UML diagramy, či při samotné implementaci o IDE vývojové prostředí.

5.1.1 PHPStorm

Jedná se o vývojové prostředí vhodné pro vývoj webových aplikací. Mezi jeho hlavní výhody patří především široký výběr pluginů, které při vývoji SW pomáhají odhalovat chyby ještě před samotným testováním. Vybral jsem ho především z důvodu podpory frameworku Symfony2, kde v případě rozsáhlejších projektů je jeho pomoc pro mě nepostradatelná. [22]

5.1.2 Enterprise Architekt

Kompletní nástroj vhodný především pro analýzu a návrh softwaru, kde jeho možnosti jsou velice rozsáhlé. V této práci jsem ho použil k vytvoření těchto diagramů:

- doménový model 2.2
- model tříd 4.1
- use case diagram 2.1

Jedinou nevýhodu vidím v tom, že v současné době není multiplatformní (pouze MS Windows).[23]

5.1.3 Ubuntu 15.10

Linuxová open-sourced distribuce pro osobní počítače a pro servery. Vývoj sponzorován společností Canonial, která se zabývá vývojem softwaru. Nabízí

širokou možnost pro vývoj a provozování serveru především v oblasti webových služeb.[24]

5.2 Implementace

Popis samotné implementace lze rozdělit do několika bodů. Jelikož jsem pracoval ve frameworku, který využívá návrhový vzor MVC, tak lze i podle toho rozdělit popis jednotlivých částí.

5.2.1 Model

Modelem mám v aplikaci na mysli především entity (třídy) a jejich logiku spolupráce. Mapování objektů do relační databáze zajišťuje Doctrine2, které stačí pouze třídu označit jako entitu, o kterou se má starat. Dále pomocí anotací u každého atributu označit o jaký typ atributu se jedná se vůči databázi. Není potřeba vytvářet tabulky či provádět select/insert příkazy do databáze. Pouze je potřebné vytvořit schéma databáze podle následujícího příkazu z kořenového adresáře Symfony:

```
app/console doctrine:schema:create --force
```

Tento příkaz vytvoří celé databázové schéma, které je potřeba mít k entitám označené pomocí anotací ve třídách. Anotace vypadá velice jednoduše. Pro jednoduchost třída Kontakt, u které budeme evidovat pouze jméno, věk a adresu typu **Adresa**:

```
/**
*@ORM\Entity
*/
class Kontakt\{
/** @ORM\Column(type="integer")
*  @ORM\Id
* @ORM\GeneratedValue(strategy="AUTO")
*/
private $id;
/**
* @ORM\Column(type="string")
*/
private $jmeno;

/**
* @ORM\Column(type="int")
*/
private $vek;

/**
* @ORM\ManyToOne(targetEntity='path/to/Adresa')
*/
```

Díky těmto anotacím v kódu Dotrine zajistí vytvoření tabulky s názvem Kontakt se sloupci id (automaticky generovaná hodnota, id, typ integer), jméno (string), věk (integer) a odkaz na základě parametru *id* třídy **Adresa**. Ani o načítání položek z databáze či o jejich úpravu se nemusíme na úrovni databáze vůbec starat, vše řešíme přes objekty v kódu PHP.

Další výhodou těchto anotacím je také to, že Symfony je schopné dogenerovat potřebné tzv. gettery a settery pro jednotlivé třídy. To většinou umí i vývojové prostředí, ale Symfony vygeneruje i metody, které jsou potřebné v případně, že jeden objekt obsahuje kolekci jiných objektů, což jsem při vývoji velmi ocenil.

```
app/console doctrine:generate:entities nazevBundlu
```

5.2.2 Viewer

Viewer představuje vlastně jedinou část, kterou vidí uživatel a se kterou interaguje. Symfony využívá věstavěný propracovaný bundle Twig[25]. Nabízí použití velkého množství filtrů, práce s kolekcemi, možnost dědění šablon, vkládání šablon a mnoho dalšího.

Další podstatnou část uživatelského rozhraní jsou formuláře, které tvoří jednu z hlavních částí Symfony a Twigu. Formuláře jsou jednoduché třídy, které je možné generovat na základě anotací ke zdrojovým třídám. Pomocí příkazu:

```
app/console doctrine:generate:form nazevBundlu:nazevTtridy
```

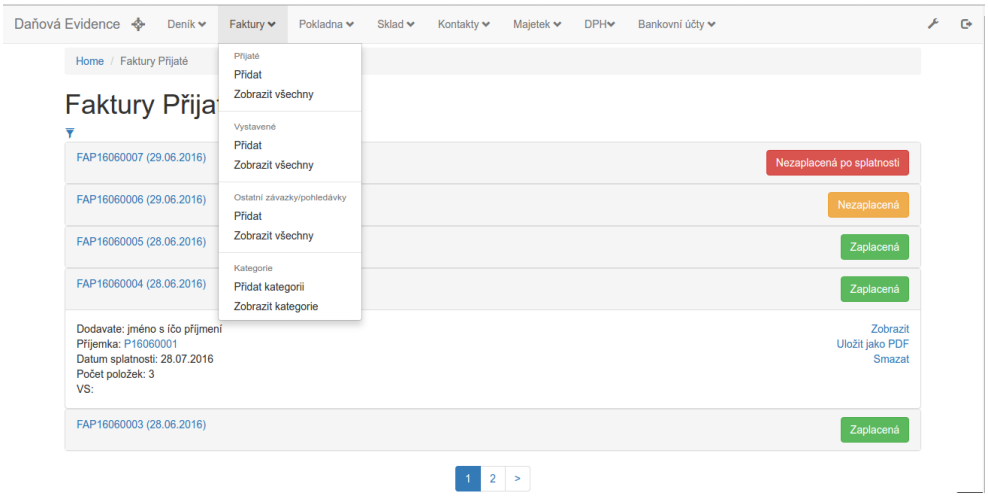
se vytvoří třída se dvěma důležitými metodami. *SetDefaultOptions*() kde je nastavení, vzhledem k jaké třídě se bude formulář zpracovávat. Pokud budu pokračovat s příkladem třídy **Kontakt**, tak to bude *path/to/Kontakt*. Atribut *buildForm* je vlastně jádro(obsah) formuláře, zde se definují veškeré položky, které bude formulář obsahovat:

```
$buildForm -> add('jmeno', 'string')
 -> add('vek', 'integer')
 -> add('adresa', 'class',array(
        'class' => 'path/to/Adresa'
 ));
 -> add('tlacitko_pridat_editovat','submit')
```

Výsledný formulář bude obsahovat 3 pole (jméno, příjmení a kolekci všech zaevidovaných adres) a tlačítko odeslat, které slouží pro odeslání formuláře ke zpracování, který je následně zachycen v Controlleru.

5.2.3 Controller

Controller je jádro celé aplikace. Zde se rozhoduje, co se udělá dál s vyplněným formulářem, či jaká akce bude následovat např. po zpracování formuláře. Jednotlivé controllery se volají (v případě použití anotací) na základě parametru



Obrázek 5.1: Screen z aplikace – výpis přijatých faktur

v tzv. *route*, které se používají k provázanosti celé aplikaci. Příklad jednotlivé route:

```
@Route('/denik/polozka/{polozkaPenzenihoDeniku}/edit',
        name='pen_den_polozka_edit')
```

Díky tomuto zápisu se v případě přechodu na tuto adresu spustí akce controlleru, který je definovaná pod touto routou. V případě přesměrování mezi akcemi controlleru nebo při vytváření odkazů v Twig šabloně lze použít jméno routy, které je uvedeno v příkladu výše.

Využití služeb (service) v Symfony patří mezi základní komponenty. Pomáhají především k zpřehlednění kódu, znovupoužitelnosti a předávání závislostí. Například je vhodné využít službu pro správu (přidávání, mazání) jednotlivých entit.

5.3 Závěr

S vhodnými nástroji a dobře propracovaným návrhem je samotný vývoj již jednoduchý. To bohužel nebyl můj případ, díky němuž jsem musel návrh několikrát upravovat během samotné implementace.

Screen 5.1 zobrazuje vzhled výsledné aplikace na klasickém PC a obrázek 5.2 byl zachynec na mobilním zařízení.

Obrázek 5.2: Screen z aplikace (mobilní zobrazení) – výpis přijatých faktur

Kapitola 6

Testování

Testování patří mezi důležité položky při vývoji SW. Jedna se o velice dlouhodobou záležitost, která by měla být prováděna již během vývoje, jelikož opravit chybu při vývoji je několikrát méně časově a finančně nákladné, než během pilotního nasazení, kdy je navíc nutné čekat na nasazení nové verze.

Testovat SW lze několika způsoby. Od automatického testování jednotlivých tříd (Unit Test) přes testování komunikaci více tříd až po uživatelské testování, které probíhá z pohledu běžného uživatele pomocí tzv. testovacích scénářů, které se vážou ke konkrétním funkčním požadavkům případně celé specifikaci aplikace.

6.1 Testování podle testovacích scénářů

Postupoval jsem podle následujících scénářů, které jsem prošel na svém vývojovém notebooku s operačním systémem Linux s webovými prohlížeči Google Chrome a Mozilla Firefox a také jsem měl možnost otestovat aplikaci v lokálním prostředí firmy Rydval s. r. o., kde byl nainstalován systém Windows 7 s prohlížečem Internet Explorer. V poslední řadě jsem také provedl test na mobilním zařízení se systémem iOS ve webovém prohlížeči Safari, kde roli serveru hrál můj osobní notebook s nainstalovaným Apache serverem a MySQL databází.

6.1.1 Testovací scénář: vytvoření dokladu a jeho uložení

1. přechod na stránku pro vytvoření přijaté faktury→stránka se načte.
2. nastavení jednotlivých atributů→atributy vyplněny.
3. přejít k přidávání jednotlivých položek k faktuře→stránka se načte.
4. vyplnit atributy položky a přidat položku k faktuře→položka se zaeviduje a je možnost přidat další.

5. opakovat bod 4.
6. fakturu uložit→zobrazí informace u uložení faktury.
7. vyhledat fakturu mezi přijatými fakturami →faktura nalezena a obsahuje přidané položky z bodu 4 a 5.

Testovací scénář pro 2.1.1 proběhl v pořádku.

6.1.2 Testovací scénář: editace a případné mazání dokladu

1. Přejít na stránku pro přidání položky v peněžním deníku→stránka se načte.
2. nastavení jednotlivých atributů, prohodit hodnoty atributů daňová částka a nedaňová částka →atributy vyplněny.
3. uložit záznam→zobrazí se informace o tom, že položka byla přidána.
4. přejít do seznamu zaevidovaných peněžních položek→seznam vypsán.
5. najít zaevidovanou položku z kroku 3→položka nalezena.
6. kliknout na položku→zobrazí se položka ve stejném formuláři jako v kroku 1, kde budou jednotlivá pole vyplněny podle kroku 2.
7. přepsat špatně vyplněné atributy a uložit položku→zobrazí se informace o úspěšné editaci položky.
8. nalezení editované položky→položka nalezena s opravenými položkami.
9. vybrat možnost smazat→zobrazí se informativní hláška ujišťující, zda si opravdu přejeme položku smazat a položka se smaže.

Testovací scénář proběhl pro 2.1.2 v pořádku.

6.1.3 Testovací scénář: vytváření dokladů na základě jiného dokladu

a) Na základě příjemky vytvořit fakturu.
   a) na skladě vytvořit několik skladových položek→položky vytvořeny.
   b) přejít k vytvoření příjemky, vytvořit příjemku minimálně o 2 položkách→příjemka vytvořena a je možné ji vyhledat mezi příjemkami.
   c) přejít k přidání nové přijaté faktury, vyplnit počáteční atributy a přejít k přidání položek→obrazovka s přidáním položek zobrazena.
   d) vybrat možnost přidat podle příjemky a ze seznamu vybrat příjemku z bodu 2→příjemka v seznamu nalezena.

e) vybrat možnost propojit doklad s fakturou→položky z příjemky jsou mezi položkami faktury.

f) přidat k faktuře další položky, uložit fakturu a následně ji vyhledat→faktura uložena a nalezena.

g) zkontrolovat, zda položky odpovídají položkám z příjemky a jestli jsou zaevidovány i další položky→položky ve faktuře odpovídají.

b) Na základě faktury vytvořit skladovou příjemku/výdejku.

a) vytvořit fakturu vystavenou s několika skladovými položkami a několika položkami, které nejsou spojené se skladovými položkami→požadovaná faktura vytvořena.

b) přejít k přidání výdejky, vyplnit její počáteční parametry a přejít k přidání položek→obrazovka pro přidání položek k faktuře zobrazena.

c) vyhledat v seznamu faktur bez výdejky fakturu z bodu 1 a spojit ji se skladovou výdejkou→faktura nalezena a připojena k výdejce.

d) výdejka obsahuje pouze položky z faktury, které byly zaevidovány ve spojení se skladovou položkou→pouze položky faktury s vyplněným atributem skladovaPolozka jsou přidány v příjemce.

e) přidat další skladové položky do výdejky a uložit výdejku→výdejka se uložila, navíc přidané položky se přidaly i do vystavené faktury.

Oba testovací scénáře pro 2.1.3 proběhly v pořádku.

6.1.4 Testovací scénář: Filtrování faktur, pokladních dokladů a dokladů k DPH pomocí data vytvoření

1. vytvořit a uložit minimálně 5 faktur/pokladních dokladů s rozdílným datem→doklady uloženy

2. nenastavit ani jeden parametr při vyhledání dokladů→zobrazí se všechny zaevidované faktury.

3. zobrazit filtr a nastavit datum *od*→zobrazili se faktury s datem vystavení vyšším než je datum v parametru *od*.

4. nastavení parametru *do*→zobrazili se faktury s datem vystavení nižším než je datum parametru *do*.

5. nastavení obou parametrů *od*, *do*→zobrazené faktury odpovídají předpokladu.

Testovací scénář pro 2.1.4 proběhl v pořádku.

6.1.5 Testovací scénář: vedení skladové evidence

1. vytvoření skladové položky s počátečním stavem→položka vytvořena.
2. zobrazit skladovou položku a ověřit, zda stav odpovídá zadanému požadovanému množství→současný stav odpovídá počátečnímu stavu, který je vidět i v historii pohybů na skladě.
3. vytvořit příjemku se zaregistrovanou položkou z bodu 1 a zadat požadované množství→současné množství na skladě se zvýšilo, je o tom záznam v historii pohybů na skladě u položky.
4. vytvořit výdejku s položkou z bodu 1 a zadat vyskladnění požadovaného množství→současné množství na skladě se snížilo, je o tom záznam historii pohybů na skladě u položky.
5. vytvořit fakturu/pokladní doklad se skladovými položkami (jedna z nich bude položka z bodu 1→doklad vytvořen.
6. na základě vytvořeného dokladu z bodu 5 vytvořit příjemku/výdejku a zkontrolovat stav položky na skladě→příjemka/výdejka vytvořena a stav odpovídá předpokládaném množšství.

Testovací scénář proběhl pro 2.1.5 v pořádku.

6.1.6 Testovací scénář: přihlášení uživatele

1. uživatel spustí aplikaci ve které není přihlášen→zobrazí se přihlašovací okno pro jméno a heslo
2. zadat špatné přihlašovací jméno či heslo→zobrazí se upozornění, že je špatně zadané přihlašovací jméno či heslo
3. zadat správné přihlašovací údaje→systém bude ve stavu přihlášeného uživatele
4. vybrat možnost odhlásit se→aplikace je přesměrovaná do bodu 1.

Testovací scénář proběhl pro 2.1.6 v pořádku.

6.1.7 Testovací scénář pro: Validace dat

1. vytvořit bankovní účet s nevalidním číslem účtu/předčíslím účtu→zobrazí se chybová hláška.
2. vytvořit kontakt beze jména/příjmení či nevalidní IČO→zobrazí se chybová hláška.

3. vytvořit položku v příjemce se zápornou cenou za skladovou položku→zobrazí se chybová hláška.

Testovací scénář proběhl pro 2.2.4 v pořádku.

6.2 Závěr z testování

Předešlých 7 testovacích scénářů bylo pouze k ověření základní funkčnosti. Ideální by bylo otestovat SW všemi průchody aplikace, ale těch je opravdu mnoho, což není možné v rámci bakalářské práce kompletně otestovat.

Až na pár překlepů a grafických drobností, na které jsem narazil během testování v operačním systému iOS, se průchody všemi testovacími scénáři podařily úspěšně dokončit bez závažnějších problémů.

Kapitola 7

Zhodnocení přínosů pro podnikatele ČR

Tato kapitola se zabývá ekonomickou analýzou a zhodnocením přínosů, které má vytvořená aplikace pro podnikatele podnikajícího na území České republiky.

7.1 Ekonomická analýza

Hodnotu vyvíjeného SW lze spočítat z časové náročnosti jednotlivých vývojových částí x hodinová sazba. Ale jelikož jsem byl roli analytika, designéra, vývojáře i testera a jednotlivé etapy probíhaly paralelně, nemůžu objektivně vyjádřit časovou náročnost jednotlivých částí. Celkovou dobu strávenou vývojem softwaru odhaduji přibližně na 500 hodin (v přepočtu zaokrouhleno 63 MD), během kterých jsem získával nové zkušenosti a možnosti frameworku Symfony2. Hodinová sazba začínajícího vývojáře je přibližně kolem 1 500 Kč za MD. Celková částka potřebná na vývoj aplikace je po zaokrouhlení **95 000 Kč**. Tady bych chtěl upozornit, že SW není vyvíjen pro jednotlivce, tedy tato cena není vstupním nákladem pro podnikatele využívající tuto aplikace, která bude zlomková.

K použití aplikace je nutný webový a databázový server, který si podnikatel může pronajmout, či jednorázově dokoupit.

7.2 Přínosy pro podnikatele

Hlavní výhody této aplikace jsou především v použití na jakémkoliv zařízením s moderním webovým prohlížečem. Další výhodou je vyšší přehlednost mezi zaevidovanými doklady na rozdíl od tabulkových editorů viz. sekce 3.2.

Tento SW je vhodný pro podnikatele, kteří potřebují evidovat položky, které tento SW nabízí:

- Peněžní deník
- Faktury vystavené a přijaté
- Pokladní doklady výdajové a příjmové vystavené podnikatelem
- Skladové položky
- Odběratelé/dodavatelé
- Bankovní výpisy
- Dlouhodobý majetek ve firmě

Rozšíření je možné na základě zákaznických požadavků.

Závěr

Cílem této práce bylo analyzovat, navrhnout, implementovat a otestovat systém pro vedení daňové evidence. V analytické části jsem se snažil zaměřit na funkčnost, kterou by měl vytvořený systém poskytovat. V další fázi jsem vybral vhodné technologie pro implementaci, které jsem v další fázi použil a aplikaci vytvořil, kterou jsem nakonec otestoval vůči funkčním požadavkům.

Bohužel se v době, kdy probíhala implementace, přibyla zákonná povinnost podávat tzv. kontrolní hlášení k dani z přidané hodnoty, kterou současná verze aplikace nepodporuje, ale to je jedna z možností budoucí verze aplikace.

Literatura

[1] ŠTOHL, P.: *Daňová evidence 2015: praktický průvodce.* Znojmo: Ing. Pavel Štohl, s.r.o., páté vydání, ISBN 978-80-87237-80-9.

[2] Zákon 586/1992 Sb., o daních z příjmů . 1992.

[3] Zákon č. 235/2004 Sb., o dani z přidané hodnoty. 2004.

[4] Microsoft Corporation: *Microsoft.* 1975, [cit. 2016-06-07]. Dostupné z: www.microsoft.com/

[5] The Document Foundation: *LibreOffice.* 1975, [cit. 2016-06-07]. Dostupné z: http://www.openoffice.cz/album/libreoffice

[6] STORMWARE, s.r.o.: *POHODA - Účetní program [online].* [cit. 2016-06-07]. Dostupné z: http://www.stormware.cz/pohoda/

[7] KASTNER software, s.r.o.: *Stereo - ekonomický software [online].* [cit. 2016-06-07]. Dostupné z: http://www.kastnersw.cz/stereo/

[8] Cígler Software, a.s.: *iDoklad [online].* [cit. 2016-06-07]. Dostupné z: www.idoklad.cz/

[9] STORMWARE, s.r.o.: *mPOHODA - Mobilní fakturace [online].* [cit. 2016-06-07]. Dostupné z: http://www.mpohoda.cz/

[10] Direct Accounting s.r.o.: *iUcto - chytré online účetnictví, daňová evidence[online].* [cit. 2016-06-07]. Dostupné z: http://www.iucto.cz/

[11] Philip Olson: *PHP: Hypertext Preprocessor [online].* [cit. 2016-06-07]. Dostupné z: http://php.net/

[12] Oracle Corporation: *MySQL [online].* [cit. 2016-06-07]. Dostupné z: https://www.mysql.com/

[13] THE APACHE SOFTWARE FOUNDATION: *The Apache HTTP server[online].* [cit. 2016-06-07]. Dostupné z: https://httpd.apache.org/

[14] SensioLabs: *Symfony[online].* [cit. 2016-06-07]. Dostupné z: http://symfony.com/

[15] Jonathan Wage: *Doctrine [online].* [cit. 2016-06-07]. Dostupné z: http://doctrine-orm.readthedocs.io/en/latest/

[16] Oracle Corporation: *Oracle [online].* [cit. 2016-06-07]. Dostupné z: https://www.oracle.com/index.html

[17] John Resig: *jQuery [online].* [cit. 2016-06-07]. Dostupné z: https://jquery.com/

[18] Boostrap - Core team: *The most popular HTML, CSS, and JS framework [online].* [cit. 2016-06-07]. Dostupné z: http://getbootstrap.com/

[19] Halil İbrahim Kalkan: *jTable - jQuery plugin [online].* [cit. 2016-06-07]. Dostupné z: http://www.jtable.org/

[20] Ashish Kulkarni: *wkHTMLtoPDF [online].* [cit. 2016-06-07]. Dostupné z: http://wkhtmltopdf.org/

[21] Sbírka zákonů č. 169 / 2011, o stanovení pravidel tvorby čísla účtu v platebním styku . 2011.

[22] JetBrains s.r.o.: *JetBrains PhpStorm IDE [online].* [cit. 2016-06-07]. Dostupné z: https://www.jetbrains.com/phpstorm/

[23] Sparx Systems: *Enterprise Architect - UML design tool [online].* [cit. 2016-06-07]. Dostupné z: http://www.sparxsystems.com.au/

[24] Canonical: *Ubuntu: The leading OS for PC, tablet, phone and cloud [online].* [cit. 2016-06-07]. Dostupné z: http://www.ubuntu.com/

[25] SensioLabs: *Twig - The flexible, fast, and secure PHP template engine [online].* [cit. 2016-06-07]. Dostupné z: http://twig.sensiolabs.org/

Příloha **A**

Seznam použitých zkratek

UC Use case (případ užití)

SW Software

SQL Structured Query Language (dotazovací jazyk)

PDF Portable Document Format (Přenosný formát dokumentů)

MD Manday (pracovní den – 8 hodin)

Příloha **B**

Obsah přiložené SD karty

- `instalace.txt` ..... instalační příručka
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis.tex` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `thesis.ps` ..... text práce ve formátu PS