FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Webový frontend informačního systému
Student:	Jakub Doležal
Vedoucí:	Ing. Jiří Chludil
Studijní program:	Informatika
Studijní obor:	Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2018/19

Pokyny pro vypracování

MAHALUX s.r.o. je firma zaměřující se na vývoj, výrobu a prodej elektronických zařízení. Cílem práce je vytvořit komponentový webový frontend pro informační systém, který centralizuje správu provozních normalizovaných dat firmy.
1. Analyzujte funkční a nefunkční požadavky zákazníka. Zaměřte se na složitější případy užití.
2. Na základě získaných požadavků navrhněte pomocí metod softwarového inženýrství webový frontend informačního systému zadavatele.
3. Navrhněte komponentový prototyp s ohledem na jeho znovu použitelnost.
4. Navrhněte přívětivé uživatelské rozhraní.
5. Implementujte prototyp v frameworku ReactJS.
6. Webový frontend podrobte vhodným testům.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

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

V Praze dne 21. prosince 2017

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Webový frontend informačního systému

Jakub Doležal

Katedra Softwarového inženýrství
Vedoucí práce: Ing. Jiří Chludil

13. května 2018

Poděkování

Chtěl bych poděkovat mojí rodině za podporu v době studií a při tvorbě této práce. Dále bych rád poděkoval mému vedoucímu Ing. Jiřímu Chludilovi za jeho příkladné přátelské vedení. A v neposlední řadě chci poděkovat Bc. Michaeli Voříškovi z firmy MAHALUX s.r.o. za jeho rady a vztřícnost.

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 13. května 2018 .....

Odkaz na tuto práci

Doležal, Jakub. *Webový frontend informačního systému.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.

Abstrakt

Cílem této bakalářské práce je návrh, implementace a otestování webového frontendu informačního systému pro firmu MAHALUX s.r.o. Hlavním účelem aplikace je zefektivnění a centralizování evidence provozních dat firmy.

Na základě sběru požadavků a analýzy firemních procesů je navržen informační systém s webovým frontendem, který je implementovaný pomocí frameworku ReactJS v jazyce JavaScript. Serverová část aplikace je realizována ve skriptovacím jazyce PHP s využitím frameworku Nette. Aplikace byla podrobena uživatelskému testování použitelnosti.

Výsledkem práce je informační systém spustilný v běžném webovém prohlížeči.

Klíčová slova webový frontend, informační systém, návrh, implementace, testování, hromadné zadávání dat, ReactJS, PHP, Nette framework

Abstract

The object of this bachelor thesis is to design, implement and test web frontend of information system for MAHALUX s.r.o. company. Application's main purpose is to make data storing more efficient and centralized.

Design of the web frontend is given by the system specification requirments from contracting company. It is implemented with usege of ReactJS framework in JavaScript language. Server-side application is made with PHP language and Nette framework. Usability testing was done on the application.

The outcome is the information system which you can run in every common web browser.

Keywords web frontend, information system, design, implemtation, test, bulk data input, ReactJS, PHP, Nette framework

Obsah

Seznam obrázků

Úvod

MAHALUX s.r.o. je mladá expandující firma zabývající vývojem, výrobou a prodejem elektronických zařízení. Typické zařízení obsahuje desítky až stovky různých komponent dodávaných z různých částí světa. To přináší velké nároky na evidenci rozličných dat o adresách, bankovních spojeních a dalších informacích. Proto se firma MAHALUX s.r.o. rozhodla pro zavedení informačního systému s webovým frontendem, který přinese potřebné centralizované ukládání dat a pohodlný přístup k těmto datům. Dalším hlavním účelem systému je zefektivnění procesu vyřizování objednávek.

V práci se zabývám analýzou, návrhem, implementací a testováním webového informačního systému. Úkolem úvodní části práce je sběr funkčních a nefunkčních požadavků zadavatele. Dále je také nutná analýza složitějších případů užití a firemních procesů. Následuje návrh struktury systému, komponentového prototypu a uživatelského rozhraní. Navazovat bude samotná implementace prototypu frontendu v knihovně ReactJS a vytvoření serverové části aplikace v jazyce PHP s pomocí frameworku Nette. V závěrečné kapitole je popsáno testování aplikace.

Kapitola 1

Cíl práce

Cílem úvodní části práce je analýza požadavků zákazníka, prozkoumání složitějších případů užití a pochopení firemních procesů. Kromě toho je potřeba seznámení s existujícími řešeními webových frontendů pro získání přehledu o best practicies.

Cílem praktické části práce je návrh, implementace a otestování webového frontendu tohoto informačního systému. Hlavní důraz bude kladen na dynamičnost uživatelského rozhraní. To by mělo uživateli napomáhat k co nejrychlejšímu a bezchybnému vkládání dat. K tomu budou využity šablony, které si uživatel nadefinuje společně s daty a ty následně ze šablony vloží do konkrétního formuláře. Dále by měla být zajištěna přehledná reprezentace větších záznamů a hromadné úpravy. Systém bude komponentový pro zaručení možnosti jednoduchého rozšíření systému.

Kapitola 2

Analýza

2.1 Specifikace systémových požadavků

Na základě diskuze s firmou MAHALUX s.r.o. byly stanoveny následující systémové požadavky.

2.1.1 Funkční požadavky

FN1 Agenda faktur a fakturovaných katalogových položek(fakturováno může být několik katalogových položek v počtu jeden a více).

- Přehled evidence vydaných faktur a fakturovaných položek a pro jednotlivé faktury:
  - vytvoření nové faktury,
  - editace faktury,
  - smazání faktury a
  - tisk faktury.
- Formulář pro vytvoření a úpravu jedné nebo více faktur na základě jedné nebo více objednávek. Pro tento formulář zajistit předvyplnění fakturačních položek na základě položek v objednávce či objednávkách.
- Přehled přijatých faktur a pro jednotlivé faktury:
  - uložení přijaté faktury včetně souborů,
  - změna stavu faktury (přijatá, částečně uhrazená, uhrazená),
  - smazání faktury a
  - tisk faktury.

FN2 Evidence nabízených produktů a s tím spojené vyhledání, přidání, úprava a smazání nabízeného produktu.

FN3 Agenda firemního skladu.

- Evidovat skladové pohyby u jednotlivých produktů.
- Přehled položek ve skladu.
- Editace položek skladu.

FN4 Agenda objednávek – přehled evidovaných objednávek a pro jednotlivé objednávky:

- vložení nové objednávky,
- editace objednávky,
- smazání objednávky,
- vytvoření faktur na základě jedné nebo více objednávek a
- evidence trackingu po expedování.

FN5 Formulář pro přípravu expedice položek objednávky.

- Formulář zobrazí objednané množství produktů a množství produktů na skladě.
- Ve formuláři se přednastaví množství k odeslání podle toho, zda je možné objednávku kompletně vybavit či ne. Pokud lze objednávky vybavit kompletně, nastavená hodnota bude rovna objednanému množství. Pokud nepůjde odbavit objednávku kompletní, bude nastavena hodnota 0.
- Expeditér bude dále moci upravit množství k odeslání na hodnoty v rozmezí 0 až objednané množství nebo množství na skladě podle toho, která hodnota je menší.
- Potvrzením vložení do expedice při nenulovém množství k odeslání dojde k vytvoření záznamu o balíčku a jeho zařazení do následující expedice.

FN6 Agenda dodávek.

- Přehled objednaných dodávek včetně trackingu.
- Formulář pro vložení nové dodávky včetně trackingu.
- Editace dodávky v případě špatně nebo nekompletně dodané dodávky.
- Označení dodávky jako dodané způsobí automatické naskladnění.

FN7 Agenda kontaktů.

- Na základě příznaku evidovat dodavatele a odběratele.
- Evidovaní budou dodavatelé – právnické osoby.

- Evidence odběratelů:
  - právnické osoby a
  - fyzické osoby.
- Evidovat kontakty tržních míst.

FN8 Hromadné úpravy.

- Více záznamů lze editovat jako jeden.
- Pokud jsou původní hodnoty stejné, zobrazí se k úpravě stejně, jako by se editoval jeden záznam.
- Pokud jsou původní hodnoty rozdílné, namísto pole k editaci se zobrazí tlačítko pro výběr jedné společné hodnoty (ideálně histogram), kterou lze potom opět editovat jako jednu hodnotu. Pokud se hodnota neupraví, zůstávají původní hodnoty.

FN9 Vyplnění formuláře šablonou.

- Uživatel si nadefinuje šablonu s konkrétními daty v konkrétních polích a šablonu uloží.
- Při dalším vyplňování téhož formuláře bude moci vybrat uloženou šablonu a ta mu formulář vyplní uloženými daty.
- Uživatel bude dále moci data upravit a případně šablonu upravit či uložit novou.

FN10 Správa uživatelů.

- O každém uživateli evidovat loginID, username, skutečné jméno a heslo.
- Každý uživatel má nějaké oprávnění. Oprávnění mohou být admin nebo expeditér.
- Založení uživatele.
- Upravení uživatele.
- Deaktivace uživatele.
- Přehled uživatelů.
- Administrátor bude moci vygenerovat uživateli nové heslo v případě zapomenutí hesla.

FN11 Evidence úprav dat (kdo data upravil a kdy).

- Pro každý záznam ve všech tabulkách bude evidováno kdo a kdy řádek vytvořil, upravil a smazal(technikou soft delete).
- Informace o tom, kým byla data upravena, bude zaznamenána z loginu uživatele.

FN12 Vytvořit API pro manipulaci s daty ve všech tabulkách.

2.1.2 Nefunkční požadavky

NF1 Odezvy v reálném čase (typicky maximálně 100 ms na 1000000 záznamů, u větších tabulek základní stránkování s použitím indexů).

NF2 Konzistence dat - na úrovni DB transakcí (server splňuje ACID).

NF3 Dynamicky reagující formuláře.

- Určená pole formuláře reagují na změnu v jiném poli. Například při výběru/zadání hodnoty v kolonce země se v návaznosti na to upraví nabídka v kolonce stát.
- Pokud je pro dané pole možné vyplnit pouze jednu hodnotu, bude tato hodnota automaticky doplněna.

NF4 Dynamické doplňování zapisovaného textu (našeptávač).

- Při zapisování údaje do určené kolonky systém automaticky napovídá zbylou část textu.
- Je potřeba, aby našeptávač mohl našeptávat z více sloupců.
- Našeptávač se bude využívat při odkazování na informace z jiné tabulky. Například při výběru položky z objednávky, našeptávač našeptává počet kusů z tabulky položky objednávky a název položky z tabulky číselníku. Vždy budou určeny konkrétní sloupce z odkazované tabulky.

NF5 Tabulka pro zobrazení záznamů z databázové tabulky.

- Editace řádků v modálním okně. Zobrazení upravených dat až po odeslání na server a jejich opětovném načtení.
- Možnost zobrazit subtabulku pro jednotlivé řádky k vypsání souvisejích položek.
- Globální filtrování záznamů v tabulce nad všemi sloupci a textové filtrování nad všemi sloupci.

2.1.3 Požadavky na rozhraní

PR1 Integrace do MySQL databáze a souborového systému pro upload příloh.

PR2 Přívětivé uživatelské rozhraní v ReactJS napomáhající uživateli práci odvést rychle a bezchybně.

2.2 Doménový model

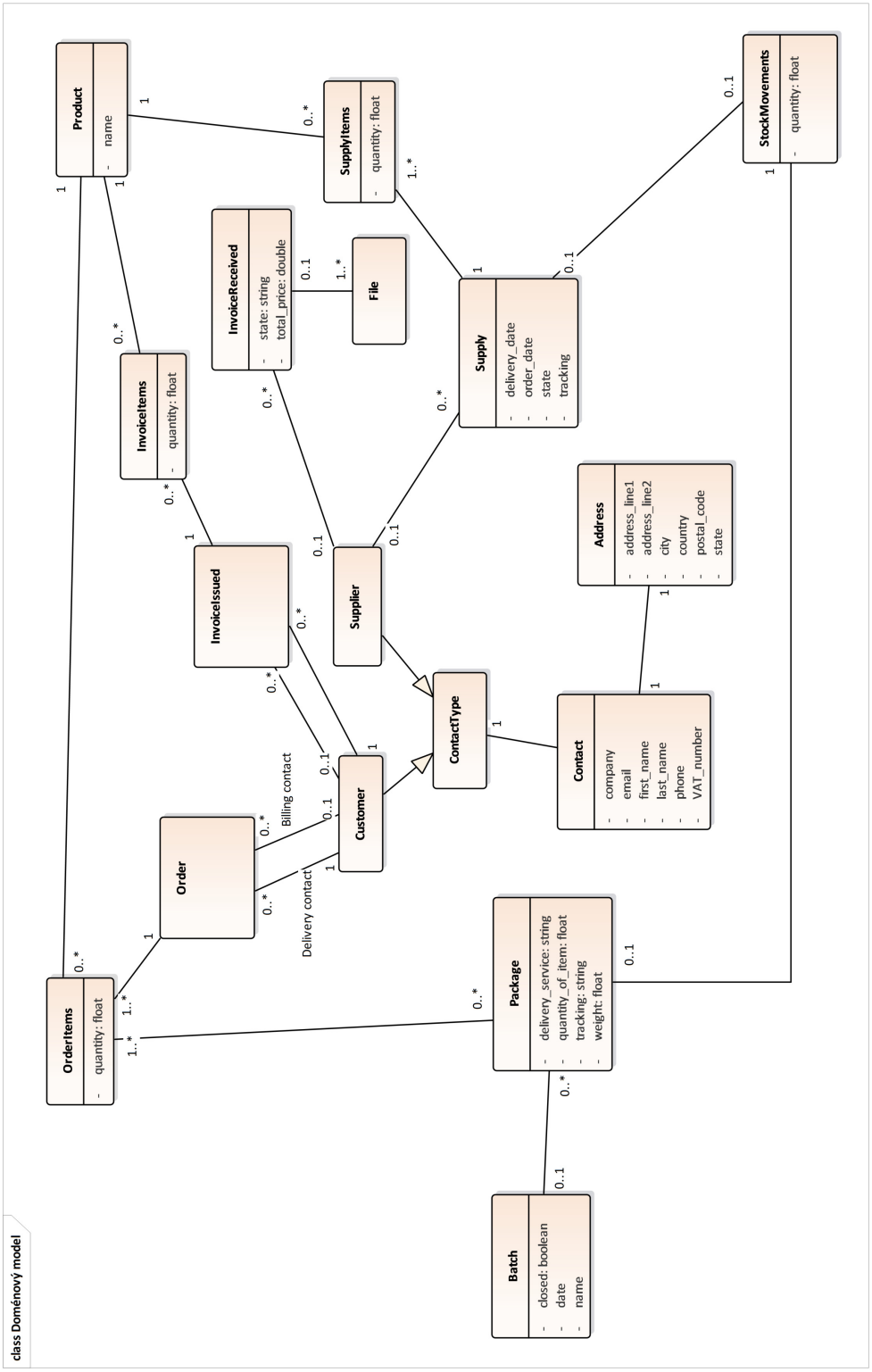
Doménový model2.1 popisuje entity a vazby mezi nimi v doméně internetového obchodu firmy MAHALUX s.r.o. Hlavní pozornost je zaměřena na objednávky a infrastrukturu procesu jejich vyřizování. Objednávka má vždy seznam položek objednaných produktů a jejich množství. Množstvím se rozumí počet kusů nebo číselnou hodnotu nějaké veličiny, například délky či váhy. U objednávky se dále evidují informace o zákazníkovi v závislosti na tom, zda jde o fyzickou či právnickou osobu. O fyzické osobě je zaznamenáno jméno a příjmení a o právnické osobě pak název společnosti a daňové ID. Dále se eviduje doručovací adresa a také, pokud je odlišná od doručovací, fakturační adresa.

V návaznosti na objednávky vznikají balíčky s objednanými produkty k odeslání zákazníkovi. O balíčcích jsou vedeny následující informace: hmotnost v kilogramech, množství objednaného produktu v balíčku, doručovací služba, která balíček doručuje, a pokud je k dispozici, tak také internetová adresa pro sledování balíčku, neboli tracking. Dále platí, že objednávka může být rozdělena do více balíčků a to i v rámci jednotlivé položky. Proto je nutné zaznamenávat množství konkrétního produktu v balíčku. Balíčky jsou pravidelně expedovány každý pracovní den ve várce pro jednotlivou doručovací společnost. Aktuálně se jedná pouze o Českou poštu, s.p. U těchto expedičních várek je evidováno datum provedení expedice a informace o tom, zda byla várka vyexpedována.

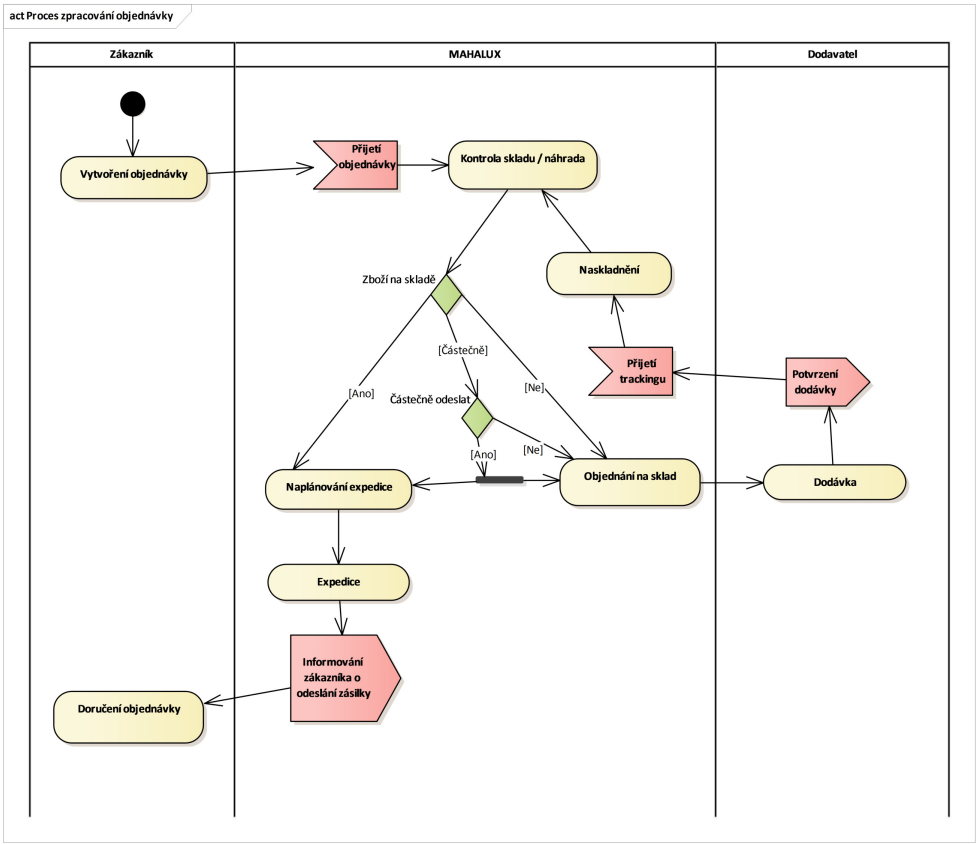
V případech, kdy není možné objednávku kompletně vyexpedovat ze zásob skladu, vznikají zásobovací dodávky. U těchto dodávek je opět nutné evidovat seznam položek s objednaným množstvím produktu, tak jak bylo definováno pro objednávky. Dále se zaznamenávají informace o dodavateli, datum objednání dodávky, stav dodávky a případně internetová adresa sledování dodávky. Dodavatelé bývají využíváni opakovaně.

Na základě přijatých zásobovacích dodávek(naskladnění) a vyexpedovaných objednávek(vyskladění) vznikají pohyby na skladě. Každý skladový pohyb je způsoben jednou z těchto dvou událostí. Je nutné evidovat množství produktu, znamenající pohyb skladu, stejně jako u položek objednávek a dodávek. Vyskladnění znamená skladový pohyb se záporným množstvím. Naskladnění je naopak pohyb s kladným množstvím.

Nezávisle na objednávkách je vedena evidence faktur, a to jak vydaných, tak i těch přijatých. Pro faktury vydané je znovu třeba ukládat seznam fakturovaných položek a jejich množství. Dále faktura obsahuje informace o zákazníkovi. Opět se jedná o různé údaje podle toho, zda se jde o fyzickou či právnickou osobu.



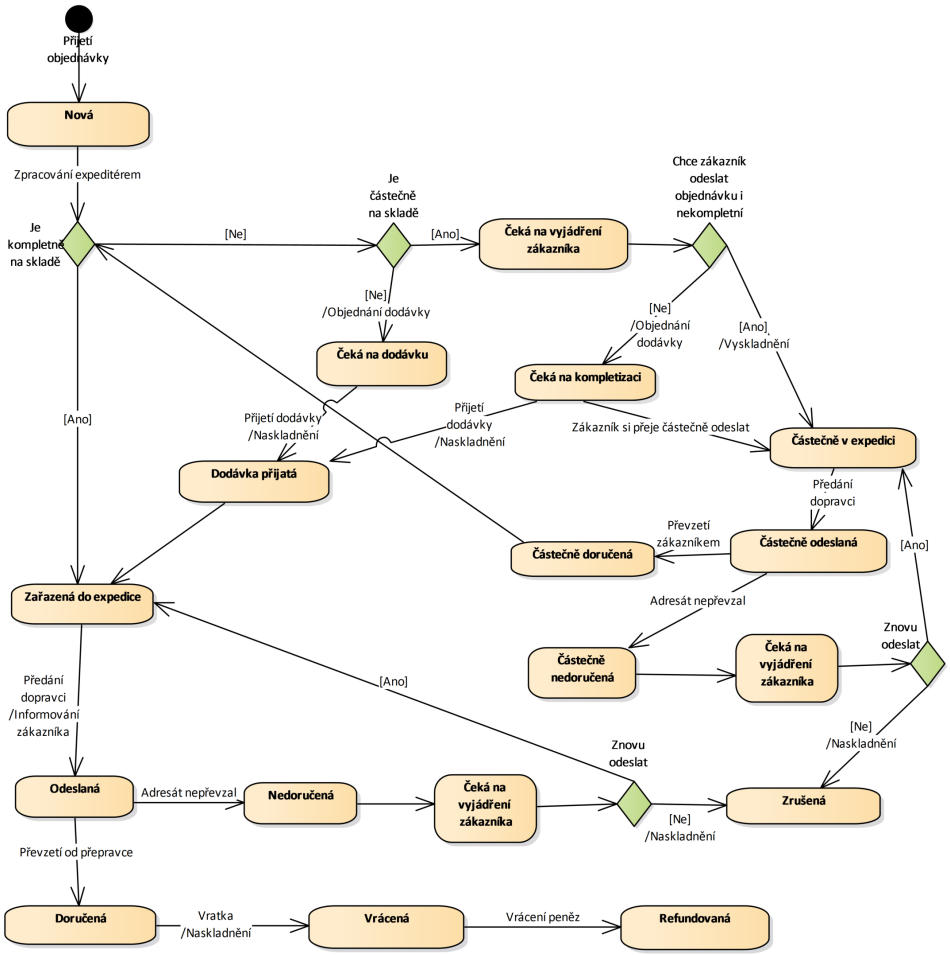
Obrázek 2.1: Doménový model



Obrázek 2.2: Diagram procesu zpracování objednávky

2.3 Diagram procesu zpracování objednávky

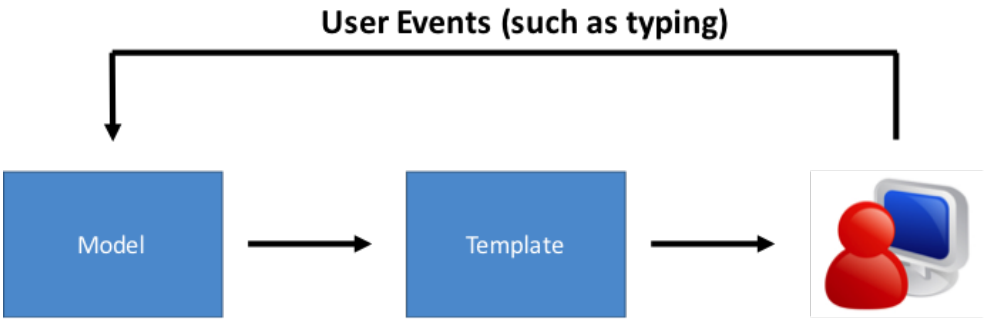
Tento diagram2.2 aktivit popisuje proces zpracování objednávky od jejího vytvoření až po doručení zákazníkovi. Objednávku vytváří zákazník a dále ji odesílá firmě MAHALUX. Ta ji po přijetí zpracovává. Kontroluje se především, zda jsou odjednané produkty na skladě. V závislosti na tom pak vznikají dodávky. Po kompletaci objednávky následuje její expedice. Ta může nastat i v případě objednávky, která není kompletně na skladě, ale zákazník si ji přeje odeslat nekompletní. Po vyexpedování je zákazník informován o předání zásilky dopravci.



Obrázek 2.3: Životní cyklus objednávky

2.4 Životní cyklus objednávky

Na obrázku 2.3 je znázorněn životní cyklus objednávky. Zajímavostí je například to, že pokud si zákazník přeje objednávku odeslat, i když není kompletně na skladu, objednávka se dostane do stavu „Částečně v expedici“.



Obrázek 2.4: Jednosměrné datové vázání

2.5 Použité technologie

2.5.1 ReactJS

ReactJS[1] je JavaScriptová knihovna zaměřená na vykreslování webových aplikací. Je založená na vizuálních komponentách, které při změně stavu aplikace efektivně překresluje. Tento proces je implementován pomocí virtualizace DOM, kterým je zaznamenáván stav skučného DOM. Při změně stavu je virtuální DOM porovnán se skutečným a následně je překreslena pouze dotčená část skutečného vzhledu stránky. [2]

React je příkladem tzv. jednosměrného vázání dat. To znamená, že události uživatelského rozhraní jsou přenášeny do stavu aplikace změnou modelu, která je provedena voláním metody `setState`. Jde o tzv. manuální detekci změny. React je tím informován o modifikaci modelu a provede již zmíněný proces propagace změn do DOM.[3] Tento proces je znázorněn na obrázku 2.4.

Pro definici toho, jak se má jednotlivá komponenta vykreslit, využívá React speciální jazyk JSX. Jde o syntaktické rozšíření jazyku JavaScript, v kterém je možné zapisovat HTML tagy. Může připomínat šablonovací systém, ale v JSX lze zapisovat přímo příkazy JavaSciptu, jejich vložením do složených závorek. Díky JSX lze zapisovat kód pro vykreslování a ostatní logiku uživatelského rozhraní společně a získat tak lepší přehled o aplikaci.[4]

2.5.2 Nette Framework

Nette je kompletní framework pro PHP, který výrazně zjednodušuje tvorbu webových aplikací. Jeho autorem je český vývojář David Grudl. Je založen na samostatných knihovnách, které dohromady tvoří celý framework. Jednou z hlavních předností Nette je vysoká míra zabezpečení za pomoci technologií, které eliminují všechny bezpečnostní mezery. Nette staví na architektuře MVC.[5]

Model-View-Controller je softwarová architektura, která vznikla z potřeby oddělit u aplikací s grafickým rozhraním kód obsluhy (controller) od kódu aplikační logiky (model) a od kódu zobrazujícího data (view). Tím jednak aplikaci zpřehledňuje, ale také usnadňuje budoucí vývoj a umožňuje testování jednotlivých části zvlášť.

Model je datový a zejména funkční základ celé aplikace. Je v něm obsažena aplikační logika. Jakákoliv akce uživatele (přihlášení, vložení zboží do košíku, změna hodnoty v databázi) představuje akci modelu. Model si spravuje svůj vnitřní stav a ven nabízí pevně dané rozhraní. Voláním funkcí tohoto rozhraní můžeme zjišťovat či měnit jeho stav. Model o existenci view nebo kontroleru neví.

View – neboli pohled, je vrstva aplikace, která má na starost zobrazení výsledku požadavku. Obvykle používá šablonovací systém a ví, jak se má zobrazit ta která komponenta nebo výsledek získaný z modelu.

Controller je řadič, který zpracovává požadavky uživatele a na jejich základě pak volá patřičnou aplikační logiku (tj. model). Poté požádá view o vykreslení dat. Obdobou kontrolerů v Nette Framework jsou presentery.[6]

Komunikace mezi jednotlivými částmi je znázorněna na obrázku 2.5.

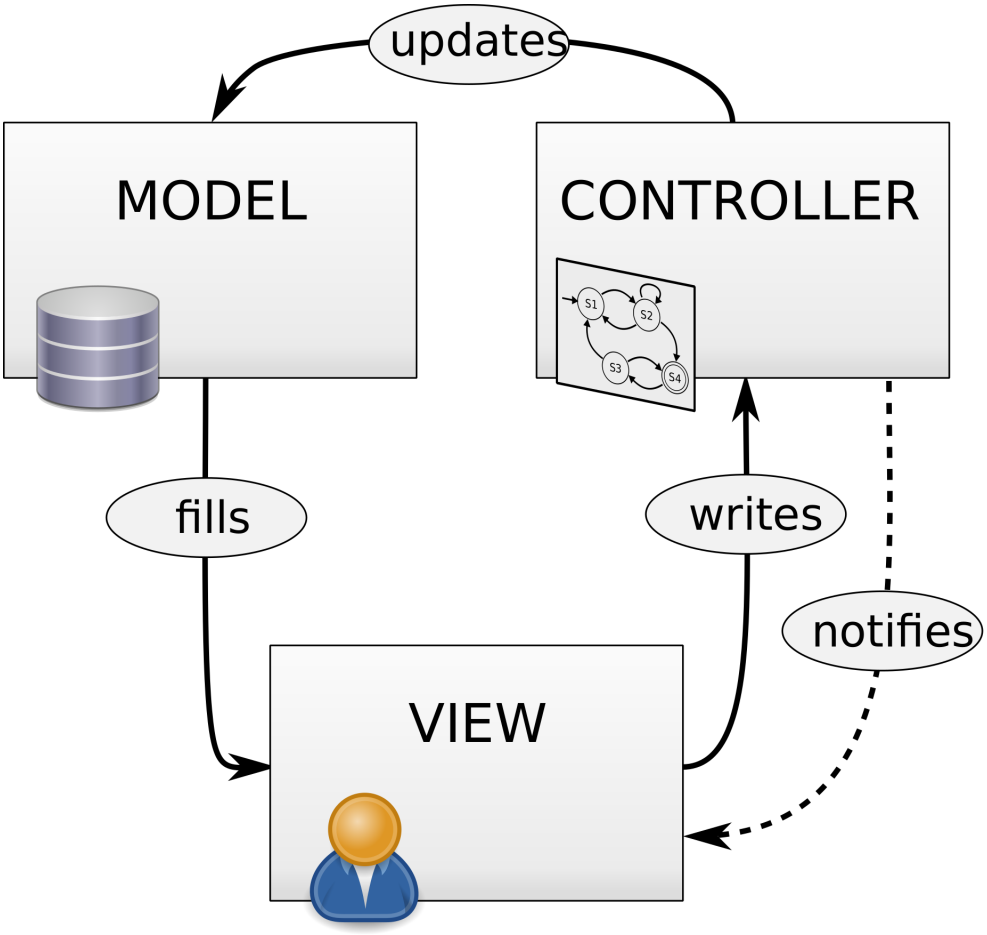
2.5.3 Doctrine 2

Doctrine 2 je ORM knihovna pro jazyk PHP. Zajišťuje mapování relační databáze na PHP objekty. Takové objekty se nazývají „Entity“. Doctrine 2 je postavena na návrhovém vzoru Data Mapper reprezentovaném vůči zbytku aplikace fasádou v podobě Entity Manageru. Pro definici vlastností jednotlivých entit se elegantně využívají komentářové anotace. Není tedy nutné entity dědit od nějakého společného předka a můžeme si hierarchii dědičností udělat zcela dle svých potřeb. Struktura databáze i veškeré její změny se generují automaticky z definic entit pomocí nástroje schema-tool. [7]

Schema-tool je nástroj pro manipulaci s databázovým modelem používaný z příkazové řádky. Na základě ORM anotací vytvoří databázové DDL skripty pro zavedení databázového schématu, které mohou být dále odladěny nebo automaticky provedeny.

Doctrine 2 je rozdělena do tří vrstev:

Common Definuje základní obecná rozhraní, třídy a knihovny. Například nástroje pro práci s kolekcemi, anotacemi, cachováním, událostmi apod. Ty jsou pak využívány oběma vyššími vrstvami. Common je sám o sobě na zbylých vrstvách nezávislý, takže je ho teoreticky možné používat i samostatně. Vše, co sem patří, je definováno v namespace `Doctrine\Common`.



Obrázek 2.5: Architektura MVC

DBAL (DataBase Abstraction Layer) Abstrahuje zbytek aplikace od konkrétního typu databáze. Primárně rozšiřuje standardní PDO, ale umí pracovat i s jinými databázovými drivery. DBAL je závislý na Common, ale může být opět teoreticky používán i samostatně bez poslední ORM vrstvy. Vše, co je jeho součástí, se nachází v namespace `Doctrine\DBAL`.

ORM (Object-Relational Mapping) je nejvyšší vrstva, která zajišťuje mapování aplikačních objektů na relační databázi, jejich persistování a načítání. ORM je závislá na DBAL i Common. Namespace této vrstvy je `Doctrine\ORM`.[7]

2.6 Výběr nástroje pro modelování tříd

Hlavním cílem bylo najít nástroj, který by uměl z namodelovaného schématu vygenerovat třídní model včetně ORM notací pro Doctrine 2 a metod get a set. Takový nástroj by velice pomohl při vývoji. V první řadě je potřeba pro vizualizaci problémové domény. Za druhé poskytne zdrojový kód modelu zastřešující databázové schéma. Vybíral jsem z následujících dvou nástrojů.

Enterprise Architect je poměrně robustní UML nástroj pro modelování celého vývojového cyklu softwaru. Pokrývá vše od sběru požadavků, modelování business procesů, architektury, domény až po class diagramy, či databázové modely. Dále umožňuje generování zdrojových kódů modelu včetně getterů a setterů pro jazyky Java, C++, C#, Python, PHP aj., které provádí na základě modifikovatelných šablon.

Tyto šablony využívají vlastní jazyk nazývány Transformation Template language. Enterprise Architect také podporuje generování DDL skriptů pro mnoho databázových enginů jako jsou Oracle, MySql, SQL Server a další.[8]

Skipper[9] je nástroj pro modelování E-R schémat zaměřený na programovací jazyk PHP. Umožňuje generovat zdrojový kód modelu s ohledem na použitý MVC framework a také dle zvolené ORM knihovny. Tím například u Doctrine 2 nepřímo zajistí i databázové schéma. To je vygenerováno pomocí schema-tool. Skipper podporuje ORM knihovny Doctrine, Doctrine 2, Propel a CakePHP.[10]

Skipper je bezesporu lépe připravený na ORM. Generování notace pro Doctrine 2 zvládá bezchybně. Enterprise Architect ORM neřeší. Na druhou stranu lze v Enterprise Architectu generování kódu doupravit pomocí šablon a plně nakonfigurovat. To může být velmi užitečné, například při použití traitu společného pro všechny třídy reprezentující databázové entity. Programový kód pro použití traitu se jednoduše vloží do šablony. Skipper toto neumožňuje a přidání je nutné dělat ručně. Nicméně vyladění šablon v nástroji Entreprise Architect pro generování zdrojového kódu včetně notace pro Doctrine 2 by bylo poměrně obtížné a s nejistým výsledkem. Proto jsem se rozhodl pro využití Skipperu i přes jeho slabší konfigurovatelnost.

Kapitola 3

Návrh

3.1 Architektura

Pro architekturu systému bude využita struktura Nette. Půjde tedy o architekturu MVC(v případě Nette pojmenovanou MVP). Část uživatelského rozhraní bude generována pomocí Latte šablon. Zbytek webového frontendu bude sestaven z vytvořených komponent React. Nette presentery budou taktéž využity pro vytvoření API.

3.2 Model tříd

Návrhový model tříd si klade za cíl správně přiřadit zodpovědnost třídám. Dále také může popisovat použité návrhové vzory a programovací techniky.[11] Pro účely navrhovaného systému jsem model tříd rozdělil do dvou částí. V první části, popisující reactové komponenty na klientské straně aplikace, jde spíše o popis prototypů JavaScriptových objektů. V drůhé části jsou poté popsány třídy pro jazyk PHP.

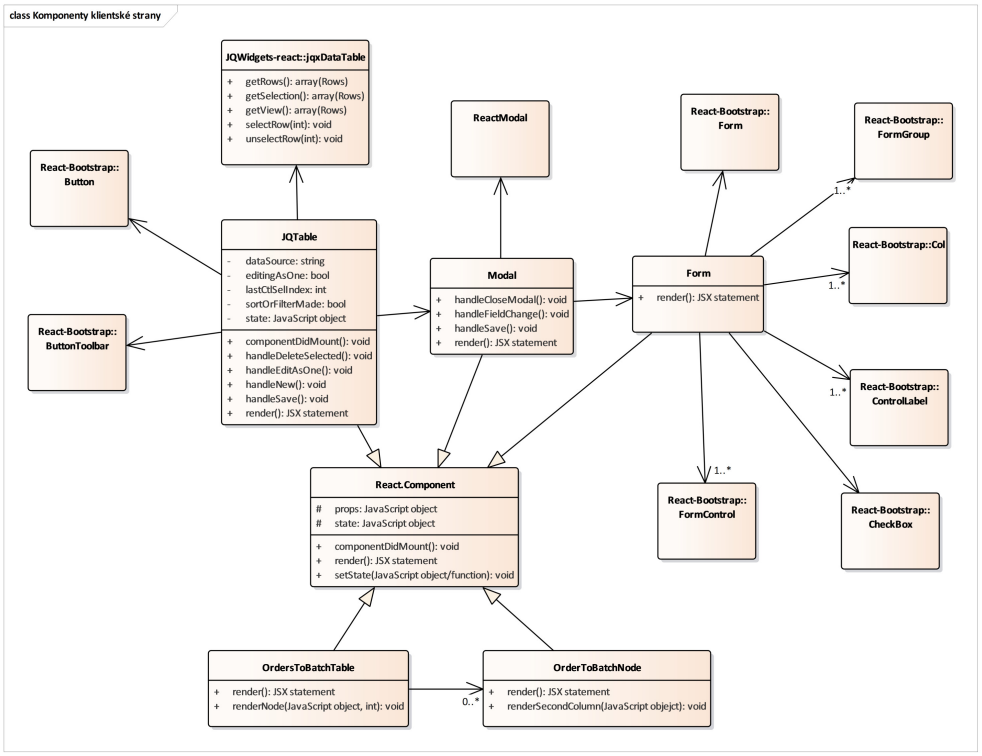
3.2.1 Klientská strana

Komponenty Reactu jsou vytvářeny děděním od třídy React.Component, to lze pozorovat na obrázku 3.1. Skládáním komponent jsou utvořeny jednotlivé webové stránky.

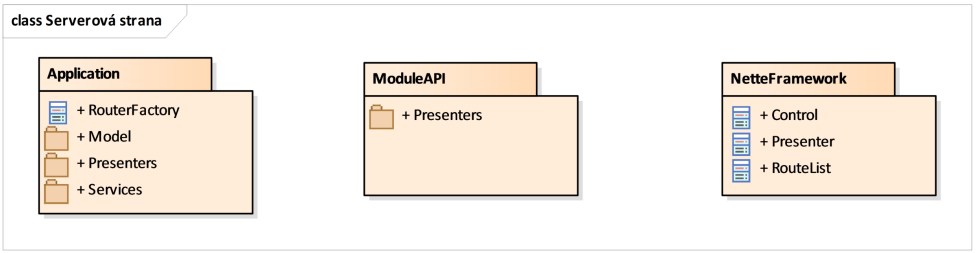
3.2.2 Serverová strana

Na serverové straně jsou využity knihovny frameworku Nette, a to především presentery a formuláře. Od samotné aplikace je pak oddělen modul API. Vše je znázorněno na obrázku 3.2.

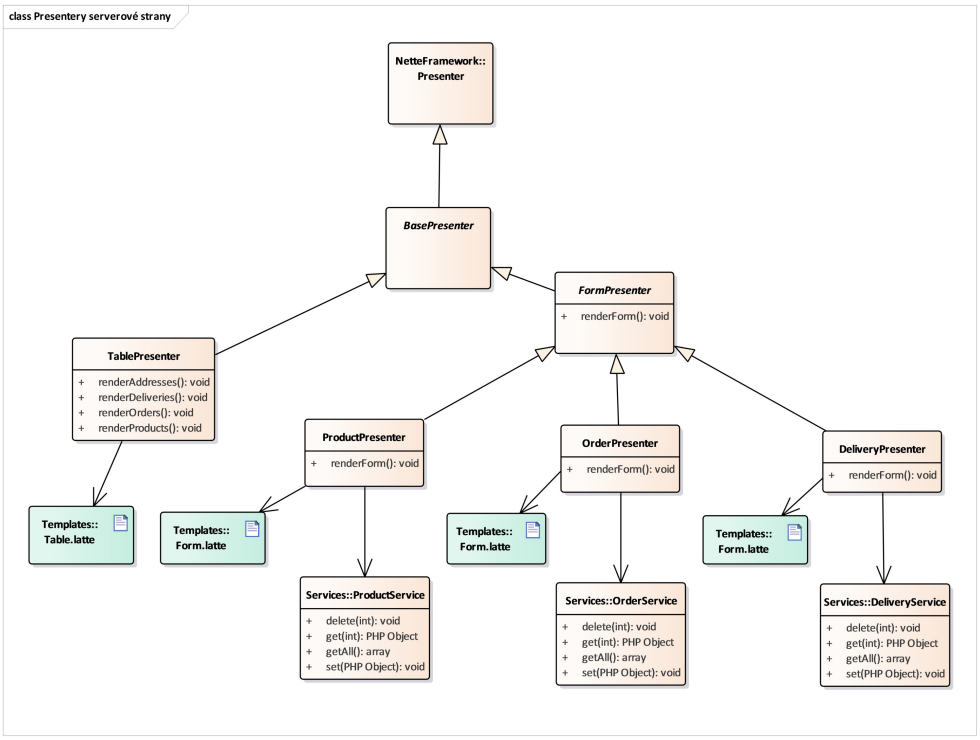
Z aplikace je nejzajímavější struktura presenterů, která je vyobrazena na snímku 3.3. Lze zde pozorovat architekturu MVP, kde model zastupují třídy



Obrázek 3.1: Komponenty uživatelského rozhraní



Obrázek 3.2: Balíčky serverové strany



Obrázek 3.3: Presentery aplikace

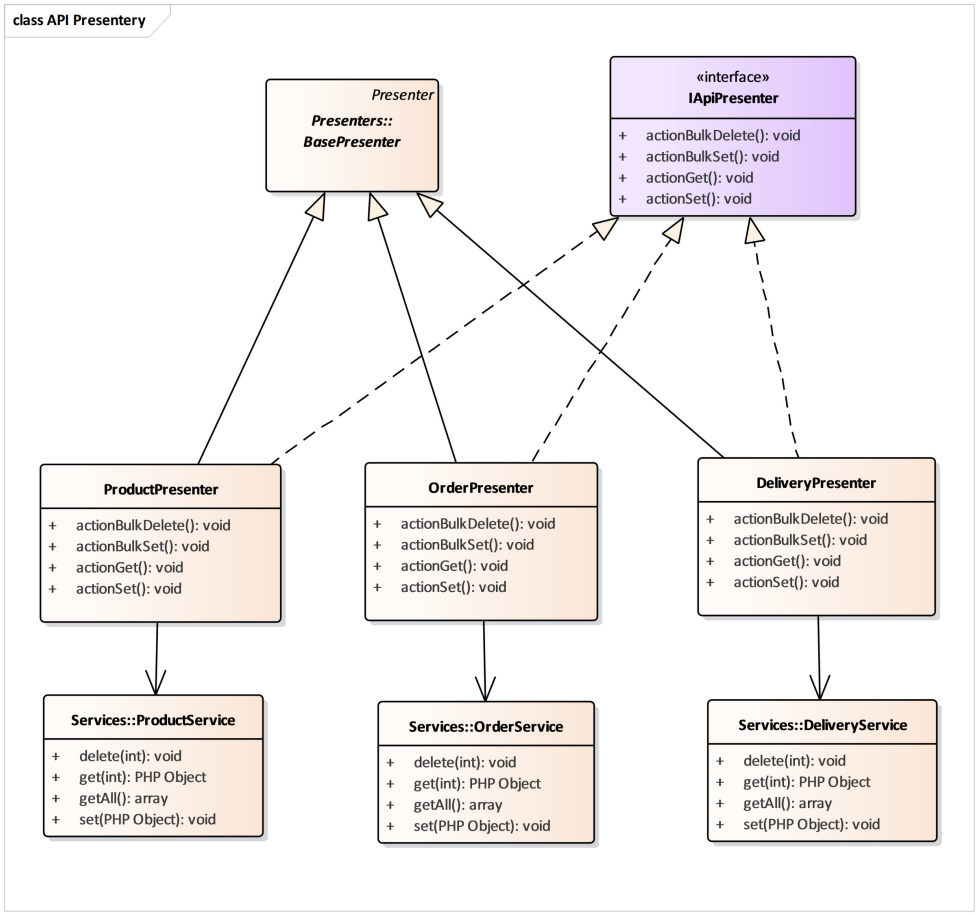
*Service, view reprezentují šablony Latté a presentery představují samotné třídy *Presenter, které jsou odděděny od Nette Preseneteru.

Podobně vypadá model presenterů API. Ačkoli zde ze zjevného důvodu chybí šablony pro vykreslování. Struktura prezenterů API je zachycena na diagramu 3.4. Dostupnost API je navržena přes URL <adresa-systému>/api.

3.3 E-R model

E-R model3.5 popisuje relační databázový model, navržený pro vyvíjený systém. Model popisuje entity se všemi vlastnostmi a vazby mezi nimi. Je vytvořen v programu Skipper, který vygeneruje zdrojový kód zastřešujících PHP datových objektů včetně notací pro ORM knihovnu Doctrine 2. Pomocí schematool je ze vzniklých tříd zavedeno databázové schéma. Vzhledem k jeho rozsáhlosti byl model rozdělen do několika částí.

První, tmavěji zelená sekce, popisuje evidované informace o uživateli systému. Každý uživatel (User) může mít více rolí (UserRole) a zároveň každou roli může zaujímat více uživatelů. Jde tedy o vazbu m:n. Tu realizuje vazební tabulka UserRoleUser. Vzhledem k tomu, že každou roli může mít každý uživa-



Obrázek 3.4: Presentery API

tel pouze jednou, tvoří primární klíč této tabulky dvojice cizích klíčů z tabulek User a UserRole.

Světle fialová oblast nastiňuje strukturu informací o kontaktních adresách. Rozlišují se kontakty zákazníků, zaměstnanců firmy, firemní kontakty, kontakty dodavatelů a tržních míst. Všechny společné informace jsou vedeny ve společné tabulce Address. Typ kontaktu je rozlišen sloupcem diskriminátoru addressType. Všechny typy kontaktů dále mají, vlastní tabulky s jejich specifickými informacemi. Tyto tabulky jsou vázány ke společné tabulce vazbou 1:1. Tato vazba je zajištěna cizími klíči v tabulkách potomků, které jsou zároveň i klíči primárními. Dále je zaveden rejstřík zemí včetně jejich států (krajů). Země typicky mají několik států a stát náleží pouze jedné zemi. O zemích je dále vedeno to, na kterém kontinentě se nacházejí.

Tabulky popisují strukturu informací o fakturách jsou v bílé části, nacházející se vlevo dole. Evidence přijatých a vydaných faktur je vzájemně oddělena. U vydaných faktur je důležité ukládat jednotlivé fakturované položky produktů a jejich množství. Fakturováno může být několik položek. Dále je pro každou vydanou fakturu nutné evidovat fakturační adresu zákazníka. K přijatým fakturám jsou uloženy soubory reprezentující tyto faktury. Platí, že každý takový soubor může být pouze u jedné evidované faktury. Dále je zaznamenána fakturovaná částka a stav zaplacení faktury.

Pro popis dodávek je použito žluté pole. U dodávek je opět evidováno, jaké produkty dodávka obsahuje a v jakém množství. K dodávkám je pomocí cizího klíče uveden dodavatel, přepravní služba a volitelně i uživatel, který dodávku vytvořil. Přijetím dodávky vždy vzniká skladový pohyb s kladným množstvím.

V tmavě růžové sekci je popsán sklad. Pro jednotlivé sklady je pomocí cizího klíče z tabulky AddressOur evidováno to, na které adrese se sklad nachází. Pro každý sklad jsou dále evidovány skladové pohyby. Ty bývají inicovány již zmíněnými dodávkami nebo expedováním balíčku. Jedná se o dva výlučné stavy. Nemůže nastat stav, že by skladový pohyb odkazoval na přijatou dodávku a zároveň na expedovaný balíček. Toto bude ošetřeno databázovým omezením Constraint, který zmíněné omezení bude kontrolovat.

Expedované balíčky jsou znázorněny ve světle zeleném obdélníku. Balíček bývá zařazen do konkrétní expediční várky. V této várce bývá více balíčků. U balíčků je také nutné evidovat jejich soubory, typicky to jsou jejich fotografie. Podobně je tomu tak i u kompletních várek. U balíčku je také nutné znát doručovací službu. Tato služba bývá použita vícekrát.

Zmiňované soubory jsou popsány v bílé sekci. Soubory budou uloženy v jejich binární podobě. U fotografií bude vytvořen a uložen datově menší náhled fotografie. Pro textové dokumenty bude volitelně uložen jejich text získaný optickým čtením.

Struktura schématu pro objednávky je zobrazena ve světle růžovém poli. Pro každou objednávku jsou evidovány jednotlivé řádky objednávky. Dále platí, že na každém řádku objednávky může být několik produktů v libovolném množství. U objednávky je dále nutné zaznamenat odesílací a fakturační adresu zákazníka. Volitelně pak adresu tržního místa a vlastní firemní adresu.

V tmavě modré oblasti je popsán model informací o nabízených produktech. O produktech jsou například evidovány jednotlivé jeho atributy. Každý definovaný atribut může být použit pro více produktů. U produktů je také rozlišován jejich typ. U typu produktu je určeno, zda jde o fyzický produkt, službu, interní produkt nebo kombinace předchozích.

Pro indexaci, a s tím spojené rychlejší vyhledávání podle hodnot sloupců, jsem zvolil následující sloupce:

- name z tabulky Product,
- ourOrderUID z tablky Order,

- lastName a email z tabluky Address,
- name z tabulky State,
- name z tabulky Country a
- alias z tabulky User.

Pro všechny vazby ve schématu je voleno výchozí nastavení pro operace delete a update a to akce restrict, která nepovoluje operace update a delete pro záznamy v rodičovské tabulce mající potomky v podřízené tabulce.

3.4 Uživatelské rozhraní

3.4.1 Formulář pro naplánování expedice

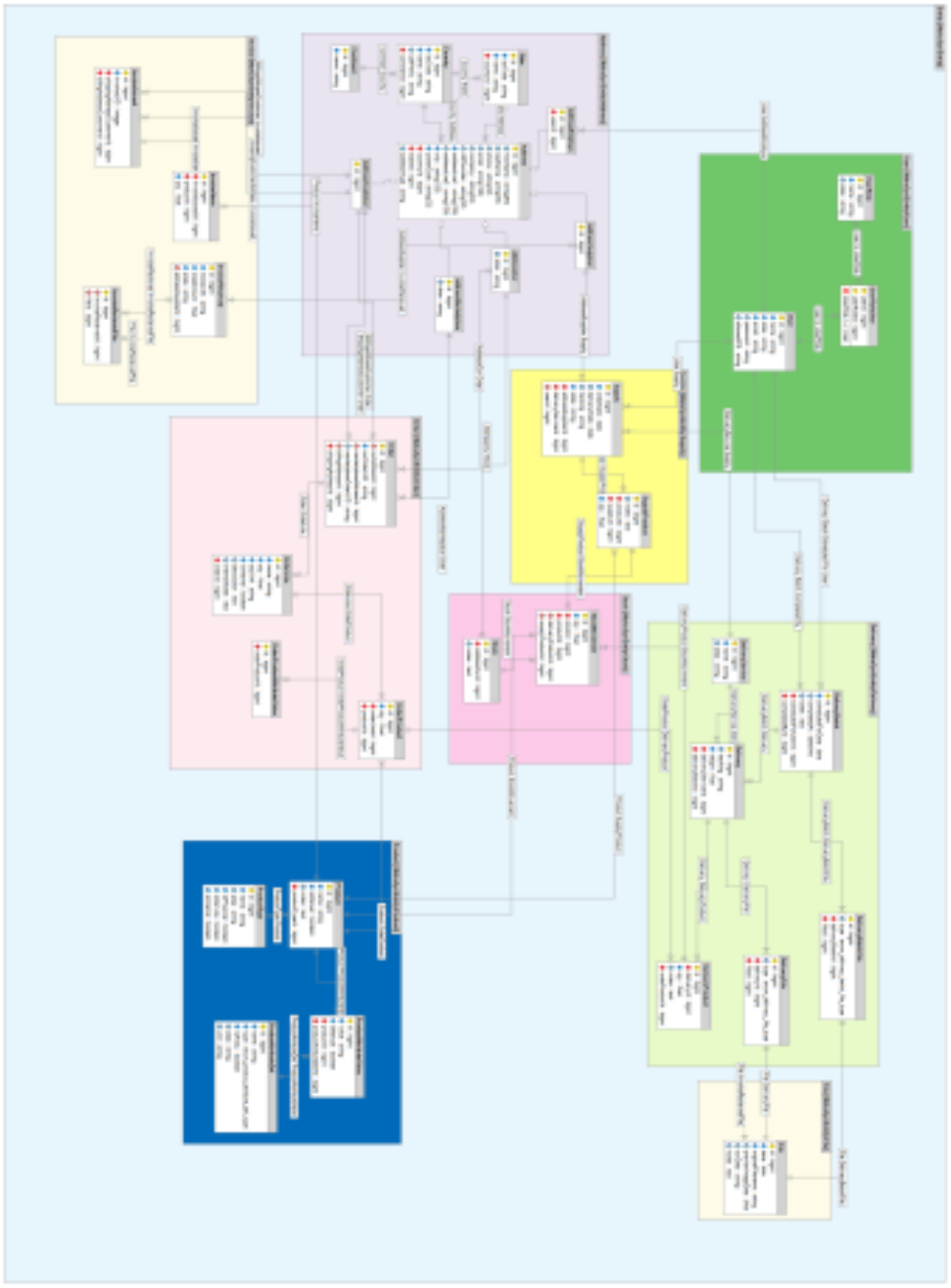
Na obrázku 3.6 je navržený formulář pro naplánování expedice definovaný ve funkčním požadavku FN5. Jednotlivé řádky tabulky tvoří konkrétní objednávky. Řádky jsou dále rozděleny dle jednotlivých produktů v objednávce. Textová pole s plným okrajem a bílým tělem naznačují upravitelné atributy. Pole s uživatelsky neupravitelným obsahem pak představují obdélníky se světle růžovým tělem a čárkovaným okrajem. V jednotlivých sloupcích jsou zleva informace o množství objednaného produktu, množství produktu, které zbývá odeslat, množství ke vložení do balíčku, množství na skladě, poznámka a váha zadaného množství produktu. V posledním sloupci se pak nachází pole pro výběr doručovací služby a tlačítko **Schedule dispatch**, pomocí kterého dojde k vytvoření balíčku se zadaným množstvím produktů v systému. Tlačítko **Bulk: Schedule dispatch** poté simuluje stisknutí všech tlačítek **Schedule dispatch**.

3.4.2 Formulář pro vytvoření faktury

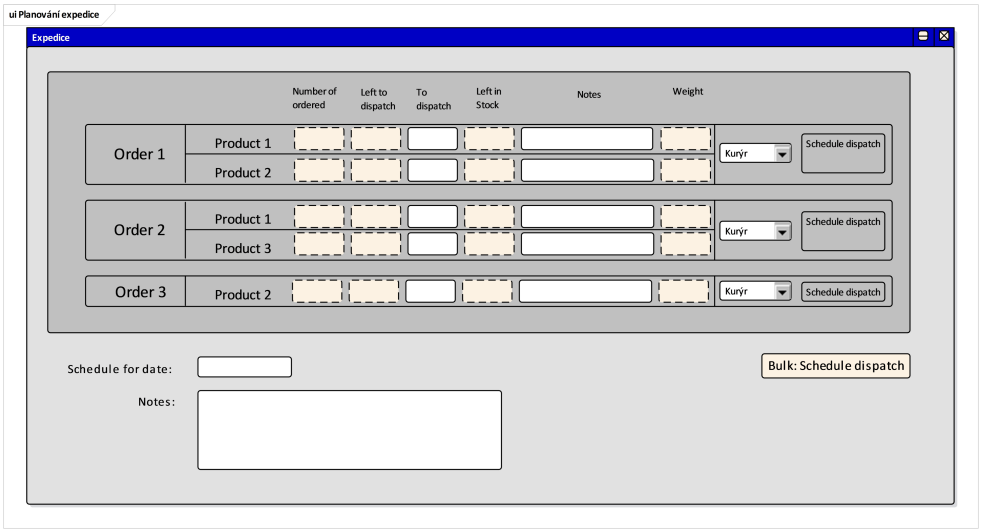
Formulář pro vytvoření faktury definovaný ve funkčním požadavku FN1 je nastíněn na obrázku 3.7. V levé části se nachází seznam objednávek včetně objednaných položek. Položky půjde nezávisle na objednávce vkládat do fakturačních položek pomocí tlačítka **Billing**. Finální podoba fakturovaných položek bude zobrazena v tabulce v pravé dolní části formuláře. Fakturované množství produktů půjde upravit ve sloupci „Qty“ nebo položky z faktury kompletně odstranit červeným tlačítkem **X**. Informace o zákazníkovi budou vyplněny dle první vybrané objednávky nebo půjde zákazníka dohledat přes textové pole s dymickým doplněním textu.

3.4.3 Tabulka

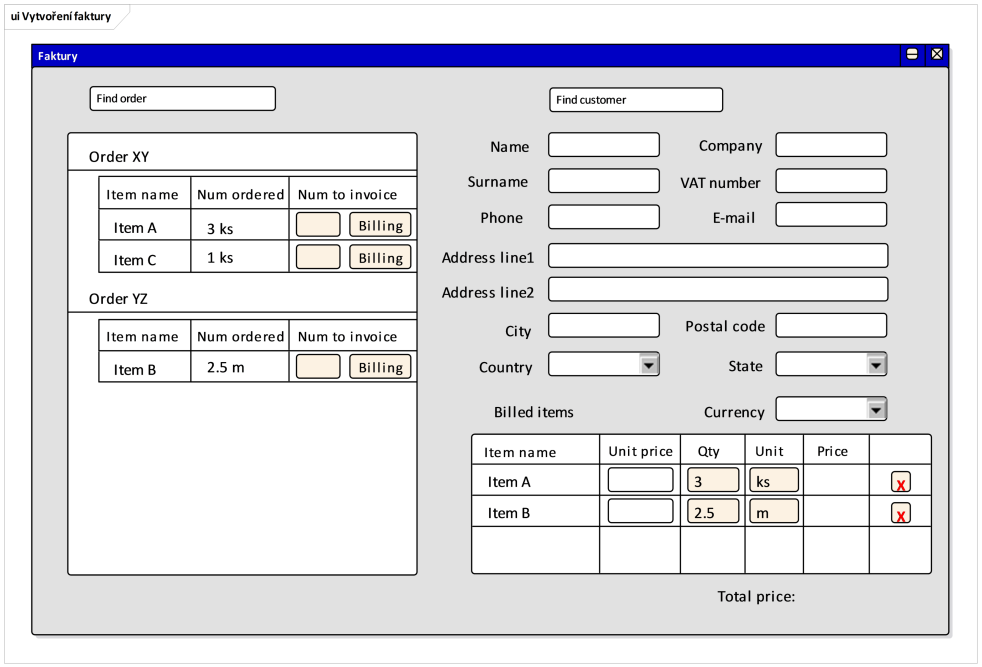
Obrázek 3.8 zobrazuje navrženou tabulku pro zobrazení hodnot z databáze. Základem tabulky je jqxDataTable[12] od jQWidgets. JqxDataTable posky-



Obrázek 3.5: E-R model



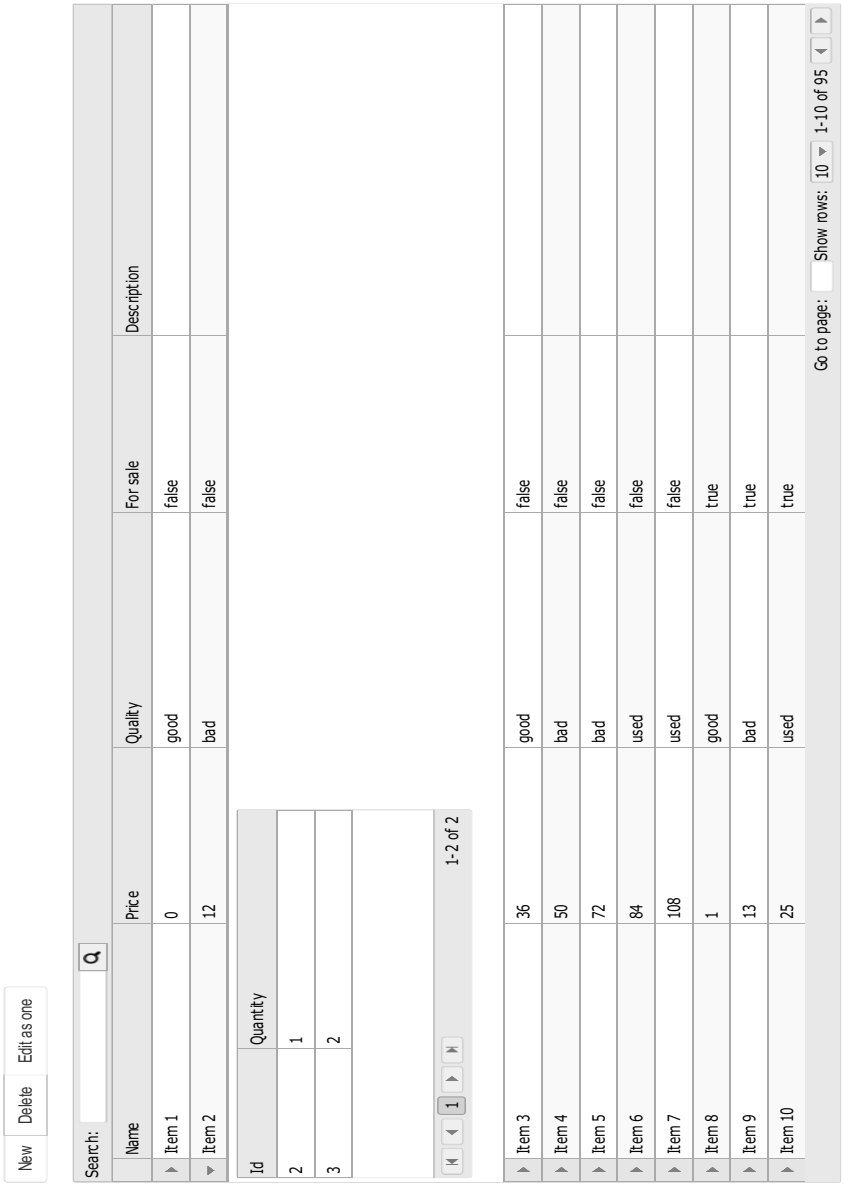
Obrázek 3.6: Formulář pro naplánování expedice



Obrázek 3.7: Formulář pro vytvoření faktury

tuje již hotové stránkování, textové řazení podle jednotlivých sloupců, textové pole pro filtrování a možnost zobrazit vloženou tabulku. Díky rozsáhlému API umožňuje JqxDataTable vytvoření vlastního způsobu vybírání řádek. Komponenty od jQWidgets mají mnoho vlastních stylů. V návrhové ukázce je použit styl „Light“.

K prvkům jqxDataTable jsou doplněna tlačítka pro přidání nového záznamu, smazání záznamu nebo záznamů na základě výběru řádků a tlačítko pro hromadnou editaci řádků.



Obrázek 3.8: Tabulka

Kapitola 4

Realizace

4.1 Instalační příručka

V této kapitole je popsána instalace aplikace na Linuxové distribuci Debian 9.

4.1.1 Instalace systému

Jako první provedeme instalaci Debianu 9 na náš stroj. Instalační obrazy jsou dostupné na stránce `https://www.debian.org/distrib/`. Obraz Debianu zahrnuje i webový server Apache2, jeho instalaci zajistíme zaškrtnutím volby „web server“ u výběru komponent systému. Při instalaci systému taktéž vytvoříme uživatele „mahasys“.

4.1.2 Nastavení uživatele

Po instalaci systému bude dalším krokem nastavení vytvořeného uživatele. Nejprve za pomoci uživatele „root“ doinstalujeme nástroj sudo, a to příkazem `apt-get intsall sudo`. Dále přidáme uživatele mahasys do skupiny uživatelů sudo pokynem `adduser mahasys sudo`. Poté se pomocí `su mahasys` přepneme na uživatele mahasys.

4.1.3 Instalace PHP

Aplikace používá skriptovací jazyk PHP verze 7.2. Spuštěním následujících příkazů nainstalujeme tuto verzi PHP a její rozšíření.

```
sudo apt-get install apt-transport-https ca-certificates
wget https://packages.sury.org/php/apt.gpg
sudo mv apt.gpg /etc/apt/trusted.gpg.d/php.gpg
echo "deb https://packages.sury.org/php/ stretch main" |
sudo tee /etc/apt/sources.list.d/php.list
sudo apt-get update
```

```
sudo apt-get install php7.2-cli libapache2-mod-php7.2
sudo apt-get install php7.2-sqlite3 php7.2-mysql
```

4.1.4 Nastavení PHP a Apache

Rozšíření naistalovaná v předchozím kroku musíme povolit. Docíleme toho následujícími příkazy:

```
sudo phpenmod pdo_mysql
sudo phpenmod pdo_sqlite
sudo phpenmod sqlite3
```

Nyní v souboru /etc/apache2/apache2.conf upravíme nastavení složky /var/www/ do nasledujícího tvaru.

```
<Directory "/var/www/">
Options Indexes FollowSymLinks
AllowOverride All
Require all granted
</Directory>
```

Dále povolíme modul rewrite příkazem `sudo a2enmod rewrite` a restartujeme Apache: `sudo apache2ctl -k restart`.

4.1.5 Stažení aplikace

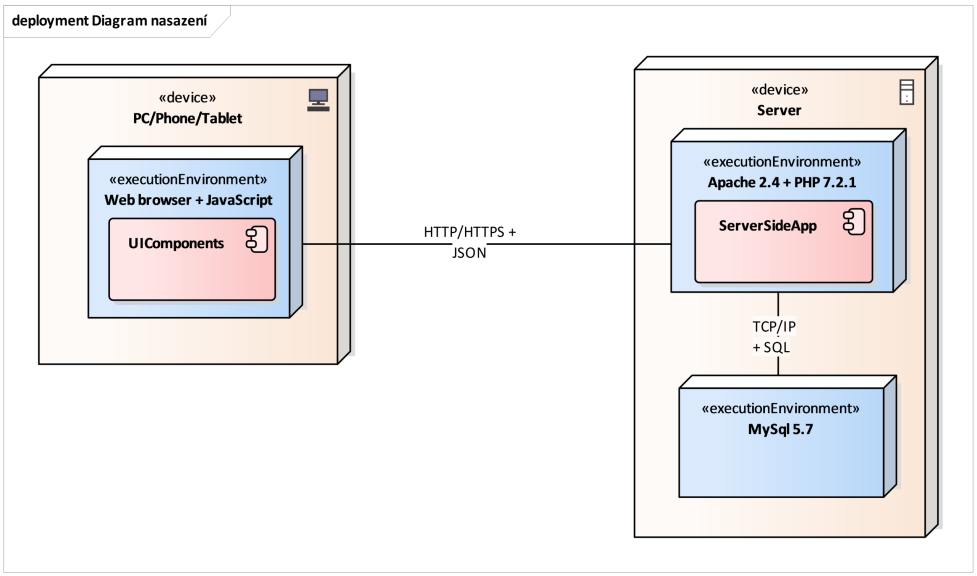
Aplikaci nainstalujeme do složky /var/www/html. Přesuneme se do ní příkazem `cd /var/www/html`. Samotnou aplikaci stáhneme pomocí `wget http://147.251.253.183:1080/dolezj24/mahasys/repository/production/archive.zip`. Stažený archiv rozbalíme příkazem `sudo unzip archive.zip` a vzniklou složku přejmenujeme – `sudo mv mahasys* mahasys`. Do této složky se přesuneme a pomocí `sudo chmod -R a+rw temp log` přidáme práva čtení a zápisu všem uživatelům pro složky temp a log. A na závěr nainstalujeme všechny balíčky, na kterých aplikace závisí příkazem `sudo php composer.phar install`.

4.1.6 Diagram nasazení

Diagram nasazení4.1 popisuje fyzické rozmístění komponent a hardwarových zdrojů systému včetně jejich propojení.[13]

4.2 Administrátorská příručka

V této kapitole jsou popsány postupy pro úpravy aplikace v návaznosti na změnu databázového modelu.



Obrázek 4.1: Diagram nasazení

4.2.1 Změna modelu v nástroji Skipper

Úpravu modelu provedeme pomocí formulářů v nástroji Skipper. Pro nové entity je vhodné nastavit jméno vygenerovaného souboru s definicí zastřešující třídy. Tvar jména bude typicky „<jméno entity>.php“. Do nové entity je také vhodné vložit atribut primárního klíče nazývaný „id“, ačkoliv ve vygenerovaném souboru entity bude nahrazen použitím identifikačního traitu. Důležité je to především u entit, kde primární klíč bude sloužit jako cizí klíč v jiné tabulce. Díky tomu budou správně vygenerovány vazební vlastnosti entit.

4.2.2 Vytvoření datového typu enumerate

Pokud potřebujeme použít výčtový datový typ, je nutné vytvořit vlastní datový typ, protože Doctrine 2 enumerate nepodporuje. Výčtový typ vytvoříme pomocí PHP třídy, kterou oddědíme od třídy `MahaSys\Entity\EnumType`. Dále už jen nastavíme jméno datového typu a pole s hodnotami výčtu. Výsledná třída může vypadat následovně.

```
namespace MahaSys\Entity;
class EnumVisibility extends Entity\EnumType
{
     protected $name = 'enum_visibility';
     protected $values = array('visible', 'invisible');
}
```

Následně je nezbytné nový datový typ Doctrine předat. Zaregistrujeme ho v souboru app/config/config.neon.

```
doctrine:
types:
enum_visibility: MahaSys\Entity\EnumVisibility
```

A také v souboru /db-managment-proj/bootstrap.php přidáním řádky:
`Type::addType('enum_visibility','MahaSys\Entity\EnumVisibility');`
Pro možnost nastavení nově vytvořeného datového typu pro sloupec entity v nástroji Skipper vložíme řádek `<data-type name="enum_visibility"/>` do tagu `<data-types>` v souboru MahaSys_Doctrine2.skipper.cfg.xml, kde je uloženo přídavné nastavení shématu pro soubor MahaSys.skipper.

4.2.3 Zavedení úprav schématu

Po vygenerování tříd pomocí programu Skipper zaneseme změněné třídy do složek db-managment-proj/src/MahaSys/Entity a app/Entity. Do upravených tříd přidáme metody get a set a zároveň nahradíme atribut id za použití identifikačního traitu příkazem `use \MahaSys\Entity\BaseEntityTrait;`. Nyní pomocí příkazu
`.\doctrine.bat orm:schema-tool:update --force --dump-sql`
vygenerujeme příslušné SQL příkazy. Ve těchto příkazech nahradíme názvy tabulek a sloupců z tvarů lower-CamelCase za zápisy s podtržítky. Výsledné příkazy spustíme v databázi.

4.2.4 Vytvoření formuláře

Formuláře se vkládají do složky app/components. Využívají Nette formuláře, kompletní popis tvorby formuláře je k dispozici na `https://doc.nette.org/cs/2.4/forms`. Pro formulář taktéž vytvoříme továrničku ve složce app/factories, kterou zaregistrujeme v konfiguračním souboru config.neon. Díky tomu nám bude formulář dodán pomocí Dependency Injection kdykoli ho budeme potřebovat.

Dále do složky app/services vytvoříme korespondující třídu pro manipulaci s daty. Tato třída bude oděděna od abstraktní třídy `AService` a bude implementovat všechny její abstraktní metody. Poslední krok je přidání třídy prezenteru a šablony pro vykreslení formuláře. Prezenter bude oděděn od abstraktní třídy `AFormPresenter`.

4.2.5 Nastavení tabulky zobrazující data

Pro manipulaci s daty nejprve upravíme API, které tabulka využívá. API je soustředěno do samostatného modulu ve složce app/ApiModule, v které je

složka presenters s prezenetery API. Ty jsou následně dostupné přes <adresa-systému>/api. Pro nové tabulky vytvoříme korespondující API prezenter pomocí nové třídy oddděné od třídy `BasePresenter` a implementující interface `IApiPresenter` tak, jak je to znázorněno na obrázku 3.4.

Následně připravíme nastavení pro tabulku. K tomu slouží továrnička `JQTableSetupFactory` a její statická metoda `create`, které jako první parametr předáme seznam sloupců s jejich označením a datovým typem sloupce. Pokud bude u tabulky využita subtabulka, jako druhý parametr metody poslouží název JSON atributu, který ponese data pro subtabulku. Třetí parametr bude v takovém případě obsahovat informace o subtabulce ve stejném formátu, v jakém tomu bylo u tabulky primární.

Nyní vytvoříme novou metodu render<NázevTabulky>, kde šabloně do atributu `setup` předáme nastavení serializované do formátu JSON. Dále také do šablony nastavíme odkazy na API, které jsme předtím vytvořili.

4.3 Uživatelská příručka

4.3.1 Tabulka

Tabulka3.8 poskytuje možnosti pro manipulaci, procházení a vyhledávání tabulovaných dat. Jde o následující funkce.

Výběr 1 řádky je uskutečněn kliknutím na požadovanou řádku zároveň se stisknutím klávesy Ctrl. Opětovným kliknutím na tu samou řádku při zmáčkuté klávese Ctrl dojde ke zrušení výběru. Opakováním postupu lze vybrat více řádek.

Výběr více řádků je docílen dvěma kroky. Nejprve označíme nebo odznačíme první krajní řádek výběru, tak jak bylo popsáno v předchozím bodě. Poté klikneme zároveň se stlačenou klávesou Shift na druhou krajní řádku výběru. Tím dojde k výběru všech řádek mezi krajními, včetně krajních.

Manipulace s daty – úprava dat probíhá ve formulářích v rámci modálního okna tabulky. Úpravy dat jsou potvrzené stisknutím tlačítka Save v modálním. Zrušení změn pak tlačítkem Cancel. K dispozici jsou tyto operace.

- **Nový záznam** – stisknutím tlačítka **New** bude otevřen nevyplněný formulář pro zadání nového záznamu.
- **Editace záznamu** – kliknutím na řádku s požadovaným záznamem a následnou úpravou dat ve formuláři modálního okna.
- **Smazání záznamu(ů)** – označením jednoho nebo více záznamů a následným stisknutím tlačítka **Delete**.

Hromadná editace – výběrem několika řádků a následným stisknutím tlačítka **Edit as one** dojde k otevření formuláře pro zadání společných hodnot pro každý vybraný řádek.

Filtrování – vložením hledaného výrazu do textového vstupu v hlavičce tabulky a jeho potvrzení stisknutím klávesy Enter nebo kliknutím na tlačítko s lupou. Výraz je vyhledán nad všemi sloupci.

Řazení – kliknutím na hlavičku sloupce budou záznamy seřazeny vzestupně dle hodnot v daném sloupci. Opětovným stiskem se záznamy seřadí sestupně a třetím kliknutím bude řazení zrušeno.

Stránkování – pro procházení stránek se záznamy jsou určena tlačítka v pravém dolním rohu tabulky. Kromě toho lze zvolit množství zobrazených záznamů na jedné stránce výběrem hodnoty v sousedním select boxu.

Subtabulka – pro tabulky s daty, které zahrnují i nějaká pomocná data, je možné zobrazit subtabulku kliknutím na malý trohúhelník na levém kraji řádky. Opětovným stisknutím tohoto trojúhelníku dojde ke skrytí subtabulky.

Kapitola 5

Testování

Testování aplikace primárně zajišťují jednotkové testy. Na úrovni jednotkových testů je také kontrolována správnost úprav dat, které aplikace zapisuje do databáze.

5.1 Uživatelské testování

Na aplikaci bylo také provedeno uživatelské testování, které hodnotí míru použitelnosti systému.

5.1.1 Testovací scénáře

V rámci uživatelského testování byly využity následující testovací scénáře (pro každý testovací scénář jsou taktéž uvedeny otázky, kterými se vyhodnotí úspěšnost průchodu scénářem).

1. Systém přijal objednávku na několik produktů. Zákazník v objednávce požaduje odeslání objednávky, i když nebude kompletně skladem. Naplánujte expedici objednávky se zbožím, které je skladem.

    - Bude zjevné, že všechno zboží v objednávce není skladem?
    - Podaří se testerovi vytvořit balíček u objednávky, která není kompletně skladem?

2. Po naplánování expedice objednávky jste zjistili, že rozložení produktů do dvou balíčků nebylo vhodné (zboží se do balíčků nevejde). Upravte expedici objednávky rozdělením produktů do tří balíčků.

    - Nalezne tester inkriminované balíčky?
    - Bude schopen rozdělení produktů upravit?

3. Do systému byly přijaty dvě samostatné objednávky od jednoho zákazníka. Zákazník u druhé objednávky požaduje společné vyfakturování obou objednávek. Vytvořte fakturu na základě těchto dvou objednávek.
   - Podaří se testerovi najít objednávky?
   - Bude testerovi zjevné, že se jedná o stejného zákazníka u obou objednávek.
   - Zvládne vytvořit požadovanou fakturu?
4. Uživatel Novák Jan zapomněl heslo pro přístup do systému. Resetujte heslo tomuto uživateli.
   - Bude tester schopný najít uživatele a ověřit resetování hesla?
5. Vyfotografovali jste balíček, který je připravený k expedici. Zaneste fotografii balíčku do systému.
   - Bude uživateli jasné jak soubor nahrát?
6. Přijali jste tracking pro sledování odeslaného balíčku. Přiřaďte tracking k balíčku v systému.
   - Podaří se testerovi spojit tracking se správným balíčkem?
7. Dodavatelská firma Abc s.r.o. změnila název společnosti na Def a.s. S využitím hromadných úprav zaneste změnu názvu do všech kontaktů na tuto firmu.
   - Bude testerovi jasná forma hromadné úpravy včetně nalezení a označení všech dotčených záznamů?
8. Přijali jste elektronický dokument s fakturou za dodávku. Zadejte informace z faktury do systému včetně vložení samotného dokumentu.
   - Nalezne a použije tester správný formulář pro vložení přijaté faktury?
9. Firma přestala v internetovém obchodě nabízet produkty typu RAM. Odstraňte všechny tyto výrobky ze seznamu nabízených produktů.
   - Podaří se testerovi vyfiltrovat a označit všechny záznamy, které je nutné odstranit?
10. Firma přijala nového zaměstnance na pozici expeditér. Vytvořte nového zaměstnance s tímto oprávněním, jménem Karel Novotný, přezdívkou Karl a emailem karel.novotny@mahalux.cz.
    - Bude možné vyplnit všechny údaje včetně přiřazení oprávnění?

11. Přijali jste dodávku od dodavatele Digi-Key Electronics, která obsahovala o 5 kusů produktu rezistor Samsung méně než měla. Upravte množství produktu v dodávce a označte ji za přijatou.

    - Podaří se testerovi dodávku najít, upravit a označit jako přijatou?

5.1.2 Výstupní dotazník

Po provedení testu jednotlivý testeři zodpoví následující dotazník pro získání zpětné vazby. Pro některé otázky jsou k dispozici 3 odpovědi, které bude možné dále rozvést.

1. Jaká byla srozumitelnost navigace?

    - Špatná – často jsem nevěděl kde jsem a jak se kam dostat.
    - Průměrná – většinou jsem měl přehled, ale byli případy, kdy jsem se v systému ztratil.
    - Dobrá – velmi přehledná.

2. Jak hodnotíte jednoznačnost pojmenování datových polí?

    - Špatná – často nejednoznačné.
    - Průměrná – jednoznačné s výjímkami.
    - Dobrá – nezaznamenal jsem nejednoznačnost.

3. Bylo vždy zřejmé jaký datový typ vyplnit (číslo, volba ano/ne, text atp.) a zda je pole povinné?

    - Ne – často nejasné.
    - Ne – občas nejasné.
    - Ano – vždy zřejmé.

4. Rozptylovali vás nějaké vizuální vlastnosti systému?

5. Chyběli vám určité jednoduché funkce (klávesové zkratky apod.)?

6. Jak hodnotíte rychlost odezvy aplikace?

    - Pomalá – časové odezvy na akce velmi pomalé.
    - Průměrná – reakce systému s nepatrným zpožděním.
    - Rychlé – okamžité odezvy.

7. Jak jste spokojen s vizuální stránkou systému?
   - Ošklivá – vzhled aplikace se mně nelíbí.
   - Průměrná – vcelku v pořádku, ale nějaká vylepšení bych uvítal.
   - Pěkná – se vzhledem jsem spokojen.
8. Stěžovalo vám používání aplikace ještě něco jiného (chybějící tlačítka, špatné reakce prvků uživatelského rozhraní na vaše pokyny atp.)?
9. Jak jste obecně spokojen s fungováním aplikace?
   - Nespokojen – s aplikací se pracuje špatně.
   - Spokejen s výhradami – aplikace je použitelná, ale menší překážky se vyskytly.
   - Spokojen – s aplikací se pracuje velmi dobře.

Závěr

Cílem práce bylo navrhnout, implementovat a otestovat webový frontend informačního systému pro firmu MAHALUX s.r.o. na základě získaných funkčních a nefunkčních požadavků. Důraz byl kladen na použitelnost aplikace tak, aby napomáhala rychle a bezchybně zadávat data do databáze.

U vytvořeného prototypu se podařilo zajistit základní funkčnost. Aplikace zvládá obsluhu procesu vyřízení objednávky, včetně plánování expedice a fakturování objednávek. Je možné zobrazovat veškerá data z databáze pomocí implementované tabulky, a dále je upravovat, mazat či přidávat nové záznamy. Pro vkládání a úpravy dat byly použity Nette formuláře. Zároveň se podařilo připravit API pro manipulaci se všemi tabulkami. Celá aplikace je postavena na návrhovém vzoru MVC, který je implementován spoluprací Nette prezenterů, Latté šablon a knihovny Doctrine 2. Nepodařilo se realizovat některé podpůrné prvky aplikace, jako je dynamické doplňování textu nebo vyplňování dat ze šablon. Do budoucna by měly být tyto funkce doplněny.

Aplikace je taktéž velice snadno rozšiřitelná a dobře konfigurovatelná. Díky tomu je možné provádět i poměrně velké změny v databázovém modelu a v návaznosti na to aplikaci jednoduše upravit. I proto by bylo vhodné vytvořit proces Continous Deployment pro automatizované nasazování nových verzí systému včetně kvalifikačních testů.

Literatura

[1] Facebook Inc.: *React*. [cit. 2018-03-07]. Dostupné z: https://reactjs.org/

[2] DeBenedetto, S.: Building a Simple CRUD App with React + Redux: Part I Intro [online]. 2016, [cit. 2018-03-07]. Dostupné z: http://www.thegreatcodeadventure.com/building-a-simple-crud-app-with-react-redux-part-1/

[3] Greene, E.: Two-Way Data Binding: Angular 2 and React [online]. 2018, [cit. 2018-03-07]. Dostupné z: https://www.accelebrate.com/blog/two-way-data-binding-angular-2-and-react/

[4] Facebook Inc.: *Introducing JSX [online]*. 2018, [cit. 2018-03-07]. Dostupné z: https://reactjs.org/docs/introducing-jsx.html

[5] Nette Foundation: *Nette framework*. Dostupné z: https://nette.org/cs/

[6] Nette Foundation: *MVC aplikace a presentery [online]*. [cit. 2018-04-11]. Dostupné z: https://doc.nette.org/cs/2.4/presenters

[7] Tichý, J.: Doctrine 2: úvod do systému [online]. 2010, [cit. 2018-03-11]. Dostupné z: https://www.zdrojak.cz/clanky/doctrine-2-uvod-do-systemu/

[8] Sparx Systems Pty Ltd.: *Enterprise Architect*. Dostupné z: http://sparxsystems.com/products/ea/

[9] Inventic s.r.o.: *Skipper*. Dostupné z: https://www.skipper18.com

[10] Inventic s.r.o.: *Skipper features [online]*. [cit. 2018-04-11]. Dostupné z: https://www.skipper18.com/en/features

[11] Mlejnek, J.: Návrhové třídy a přiřazení zodpovědností [online]. 2018, [cit. 2018-04-25]. Dostupné z: https://edux.fit.cvut.cz/courses/BI-SI1/_media/lectures/06/06.prednaska.pdf

[12] jQWidgets: *jqxDataTable [online]*. Dostupné z: https://www.jqwidgets.com/react/react-datatable/index.htm

[13] Mlejnek, J.: Návrh – rozhraní a komponenty [online]. 2018, [cit. 2018-04-25]. Dostupné z: https://edux.fit.cvut.cz/courses/BI-SI1/_media/lectures/07/07.prednaska.pdf

Příloha **A**

Seznam použitých zkratek

GUI Graphical user interface

DDL Data definition language

UML Unified Modeling language

ORM Object relation mapping

MVC Model-view-controller

MVP Model-view-presenter

DOM Document Object Model

JSX JavaScript Syntax eXtension

PDO PHP Data Objects

API Application interface

SQL Structured Query Language

Příloha B

Obsah přiloženého CD

- `src`
  - `impl` ..... zdrojové kódy implementace
    - `app` ..... aplikace serverové strany
      - `api`
        - `presenters` ..... prezentery modulu API
      - `components` ..... Nette formuláře
      - `config`
      - `factories` ..... továrničky
      - `model` ..... třídy obsluhující akce na databázovým modelem
      - `presenters` ..... prezentery aplikace
        - `templates` ..... vykreslovací šablony Latte
      - `router`
      - `.htaccess`
      - `bootstrap.php` ..... spouštěcí soubor Nette
    - `db-mangment-proj` ..... projekt pro manipulaci s databází
    - `client` ..... aplikace klientské strany
      - `components` ..... komponenty uživatelského rozhraní
      - `index.js` ..... vstupní soubor pro uživatelské komponenty
      - `package.json` ..... závislost komponent
    - `www` ..... viditelné soubory webu
      - `jqwidgets`
      - `static` ..... minifikovaná verze klientské aplikace
    - `.htaccess` ..... přesměrování do složky www pro Apache
    - `composer.json` ..... závislosti projektu
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
    - `BP_Dolezal_Jakub_2018.tex`
    - `mybibliographyfile.bib`
- `text` ..... text práce
  - `BP_Dolezal_Jakub_2018.pdf` ..... text práce ve formátu PDF
  - `zadani.pdf` ..... zadání práce ve formátu PDF