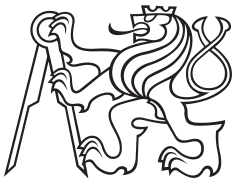
Diplomová práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Vylepšení a přesun restauračního systému na platformu ASP.NET Core

Pavel Zářecký

Školitel: Ing. Martin Komárek
Obor: Softwarové inženýrství
Zaměření: Otevřená informatika
Květen 2019

Poděkování

Chtěl bych zde poděkovat vedoucímu své práce, Ing. Martinu Komárkovi, za jeho vedení a rady. Rád bych zde také poděkoval svým rodičům a příbuzným za jejich dlouhodobou podporu při mém studiu a mé díky patří i všem ostatním, kteří mě při studiu podporovali.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze, 24. května 2019

Abstrakt

Cílem diplomové práce je analýza, vylepšení a přesun restauračního systému CashBob na platformu ASP.NET Core. Systém CashBob vznikal v předchozích letech iterativně, především v rámci závěrečných prací studentů ČVUT.

Klíčová slova: ASP.NET Core, Cloud, REST, EET, Iterativní vývoj

Školitel: Ing. Martin Komárek

Abstract

The main goal of this thesis is the analysis, improvement and transfer of the restaurent system CashBob to the ASP.NET Core platform. The Cashbob system was created iteratively over the last years, mainly in the form of theses of the ČVUT students.

Keywords: ASP.NET Core, Cloud, REST, EET, Iteration based development

Title translation: Improvement and transfer of a restaurant payment system to the ASP.NET Core platform

Obsah

Obrázky

Tabulky

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Zářecký** Jméno: **Pavel** Osobní číslo: **384532**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Studijní obor: **Softwarové inženýrství**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Vylepšení a přesun restauračního systému na platformu ASP.NET Core

Název diplomové práce anglicky:

Restaurant System Upgrade and Migration to ASP.NET Core Platform

Pokyny pro vypracování:

Analyzujte technologická omezení existující implementace serverové části restauračního systému CashBob[2] postaveného na JAVA technologiích. Nastudujte problematiku vývoje aplikací pro platformu ASP.NET [4] a analyzujte výhody/nevýhody přesunu stávající aplikace na tuto platformu.
Nově navrhněte a naprogramujte serverovou část tak, aby odstraňovala zásadní nedostatky stávajícího řešení. Práci provádějte iterativně, průběžně dokumentujte, testujte a nasazujte.
Do systému doplňte podporu elektronické evidence tržeb EET a podporu tisku účtenek.
Testování kompatibility přeprogramovaných rozhraní provádějte především pomocí existující frontendové Android aplikace[3]. Výsledný systém porovnejte s existujícími zavedenými systémy.

Seznam doporučené literatury:

[1] LARMAN, Craig a Chris RUPP. Applying UML and patterns: introduction to object-oriented analysis and design and iteractive development. 3rd ed. New Jersey: Prentice-Hall, 2005, xviii, ISBN 01-314-8906-2.
[2] ČERVENKA, Tomáš. Webové rozhraní restauračního systému, Praha, 2016. Dostupné také z: http://hdl.handle.net/10467/64854
[3] ZÁŘECKÝ, Pavel. Vývoj aplikací na platformě Android, Praha, 2016. Dostupné také z: http://hdl.handle.net/10467/64634
[4] ESPOSITO, Dino. Programming asp.net core. Indianapolis, IN: Microsoft Press, 2018. ISBN 9781509304417.

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Martin Komárek, kabinet výuky informatiky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **26.02.2019** Termín odevzdání diplomové práce: **24.05.2019**

Platnost zadání diplomové práce: **19.02.2021**

Ing. Martin Komárek podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Pavel Ripka, CSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. _____
Datum převzetí zadání

_____
Podpis studenta

Zkratky

Zkratka	Význam
REST	Representational state transfer
API	Application program interface
XML	Extensible markup language
ASP	Active server pages
EET	Elektronická evidence tržeb
RMI	Remote method invocation
JAVA SE	Java standard edition
FIK	Diskální identifikační symbol
SOAP	Simple object access protocol
HTTP	Hypertext transfer protocol
MVC	Model, view, controller
DTO	Data transfer object
SQL	Structured query language
CRUD	Create, Read, Update, Delete
JSON	JavaScript object notation

Kapitola 1
Úvod

Ve své bakalářské práci jsem se, mimo jiné, věnoval vývoji Android aplikace pro číšníky, komunikující přes REST rozhraní s pokladním systémem CashBob[Zá16]. Navazoval jsem převážně na práci jiného studenta[Hog16], který již vytvořil funkční základ aplikace. Vzhledem k tomu, že v nedávné době vstoupila v platnost povinnost elektronické evidence tržeb – EET[1], bylo by nutné k zajištění aktuálnosti upravit nejen tento funkční základ aplikace, ale i serverovou část tak, aby EET podporovaly. A dále také proto, že se v následujících kapitolách ukáže, že pro současný kód (až na samotný základ pokladního modulu) již nadále efektivně neexistuje dokumentace, bylo v rámci zadání rozhodnuto, že bude aplikace přepsána a znovu navržena tak, aby byly zohledněny nedostatky současné verze a aplikace tak byla vylepšena. Přepsání aplikace jako takové otevřelo také možnost převést aplikaci na platformu jinou než Java, pokud by toto bylo výhodné - vybrán byl framework ASP.NET Core[cor], který navazuje na původní ASP.NET, ale na rozdíl od něj je nejen multiplatformní, ale hlavně také open-source. v Následujících kapitolách tedy bude tento framework porovnán s technologiemi stojícími na Javě, ve kterých je současná aplikace psána.

1.1 Pokladní systém Cashbob

CashBob jako takový je studentský projekt, který již za dobu svého vývoje doznal spousty změn, jmenovitě například přesun od lokální desktop aplikace postupně ke kombinaci více modulů komunikujících přes rozhraní Java RMI[rmi] až k zavedení standardizovaného multiplatformního API REST[res]. Ve všech svých iteracích se ale primárně vždy jednalo o desktopovou aplikaci, a to i v případě své poslední verze vytvořené Tomášem Červenkou v rámci jeho bakalářské práce[Če16].

[1] Zákon vstoupil pro pohostinství v platnost 1. prosince 2016 [eetb]

1.2 Motivace

V zadání této diplomové práce mám stanoveny mimo jiné tyto hlavní body:

- Analýza technologických omezení existující implementace
- Analýza výhod/nevýhod přesunu aplikace na platformu ASP.NET Core
- Návrh nové aplikace odstraňující omezení aplikace současné

V současné době také panuje trend, při kterém se aplikace a systémy, historicky provozované lokálně, stěhují na Cloud[clo], za účelem co největšího zjednodušení přístupu uživateli. Toto je možné především proto, že stabilní, vysokorychlostní a především neomezený internet je v současné době (v České republice) již téměř samozřejmostí[int]. Tento trend je již možné pozorovat i v oblasti pokladních systémů pro malé podniky - existuje totiž hned několik takovýchto aplikací s cloudovou funkcionalitou[pok]. Využil bych tedy rád tuto příležitost a aplikaci navrhl tak, aby uživateli nabízela cloudovou funkcionalitu, obdobnou té, kterou již nabízejí konkurenční aplikace.

1.3 Struktura dokumentu

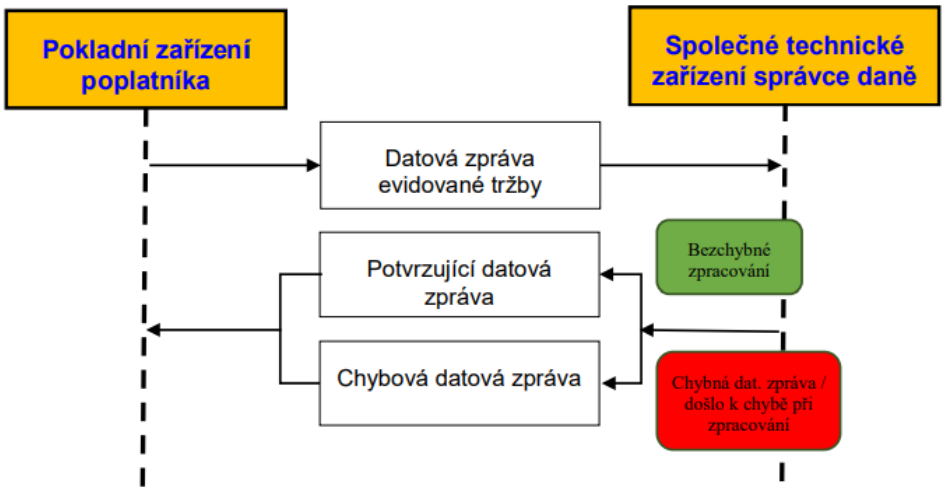
Dokument je standardně dělen na analýzu, prvotní návrh, jednotlivé iterace vývoje a zhodnocení práce. Součástí analýzy je seznam důvodů pro přechod na platformu ASP.NET Core, prvotní návrh pak zohledňuje celkovou funkcionalitu, kterou bych si přál v průběhu vývoje implementovat. Pokud by se v průběhu iterací ukázalo, že některá navržená funkcionalita by měla být změněna, budou jednotlivé iterace také obsahovat pozměněný návrh. Testování funkcionality pak bude probíhat průběžně v každé z iterací. Závěrem aplikaci porovnám s konkurenčními produkty na českém trhu a shrnu celkový výsledek práce.

Kapitola 2
Analýza

V této kapitole podrobně přiblížím řešený problém a do hloubky představím platformu ASP.NET Core, včetně srovnání s platformou Java SE + Spring, ve které je původní aplikace implementována. Závěrem bude analýza výhod a nevýhod, které by případný přesun aplikace na novou architekturu představoval.

2.1 Elektronická evidence tržeb

Komunikace probíha v režimu požadavek/odpověď a pokud zpráva vyhovuje kontrole všech náležitostí, je po zaslání evidované tržby vrácena odpověď s tzv. fiskálním identifikačním symbolem - FIK, který je nutné vystavit zákazníkovi [eeta].



Obrázek 2.1: Schema komunikace se serverem finanční správy[eeta]

2.1.1 SOAP zprávy

Pokladní systémy si se serverem finanční správy vyměňují zprávy pomocí protokolu SOAP, který je založený na formátu XML. Soap je vlastně předchůdce v dnešní době převážně používaných webových API, jako např. REST a stejně jako tyto služby používa přo přenost vrstvu HTTP, nikoliv ale metody GET a POST, proto k webovým API nepatří [soa].

2.1.2 Vývoj

Existují dvě prostředí EET - produkční a neprodukční. Testování rozhraní probíhá v neprodukčním prostředí, jelikož "zaslání zprávy do neprodukčního prostředí není zasláním údajů o evidované tržbě ve smyslu §18 Zákona o Evidenci Tržeb"a FIK takto vrácený není platný.

2.2 Popis problému a analýza stávající aplikace

2.2.1 Analýza serverové části

Existují dvě verze aplikace Cashbob, analyzoval jsem tedy obě - původní (kompletní) verzi psanou Lukášem Vyhlídkou[Zá16] a také verzi Tomáše Červenky, která ale postrádá některou funkcionalitu jako např. správu skladu, ale především se až na některé lehké úpravy nezaobírá backendem (který, až na drobnosti, zůstává nezměněn), pouze nahrazuje frontend (který jsem se později rozhodl prozatím nepřevádět, o tom ale detailněji až v následujících sekcích)[Če16]. Jak již bylo zmíněno v úvodu, stávající aplikace CashBob je psána v jazyce Java a s ostatními moduly komunikuje pomocí REST rozhraní. V základní serverové části je pomocí uživatelského rozhraní možné přistupovat k datům aplikace, mimo jiné jsou to následující položky v backendové části:

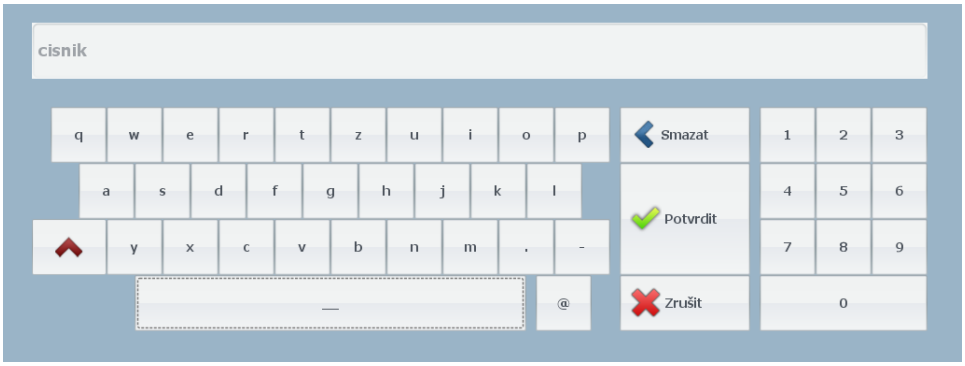
- Naskladnění/vyskladnění/odpis surovin/produktů ve skladu
- Položky menu
- Uživatelské účty
- Správa stolů

Děje se tak pomocí uživatelského rozhraní v podobě vyskakovacích oken, které na rozdíl od zbytku aplikace nelze ovládat pomocí dotykového displaye, pro který je aplikace zjevně určena, jelikož obsahuje i vestavěnou dotykovou klávesnici.

Použité technologie

Serverová část je psána v jazyce Java 1.7. Pro uživatelské rozhraní jsou použity pro Javu standardní grafické knihovny AWT a Swing. Dle analýzy Tomáše Červenky byl backend původní aplikace realizován za pomocí frameworku

Obrázek 2.2: Uživatelské rozhraní aplikace po přihlášení



Obrázek 2.3: Vestavěná dotyková klávesnice

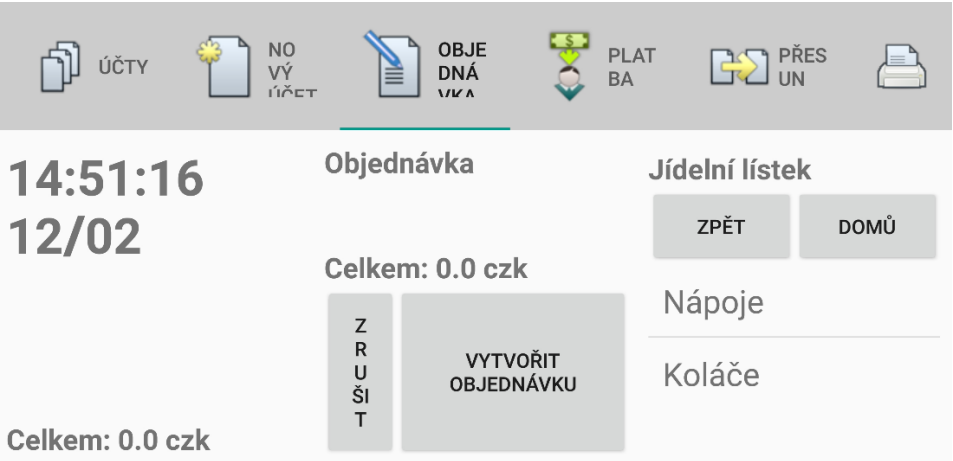
Spring a přístup k databázi realizován pomocí knihovny Hibernate, jak je pro Spring typické[jav]. Sám jsem bohužel do zdrojového kódu původní aplikace nahlédnout nemohl, není totiž již k dispozici - veškeré vlastní testy jsem prováděl na zkompilované aplikaci.

2.2.2 Analýza Android aplikace

Aplikaci původně psanou Tomášem Hogenauerem jsem již analyzoval[Zá16], včetně testů funkčnosti rozhraní REST, ve své bakalářské práci. Pro úplnost zde tedy shrnu výsledky této původní analýzy. Ve své bakalářské práci jsem také již analyzoval i platformu Android, pro kterou je aplikace určena - budu se tedy na ni případně odkazovat.

Určení

Jedná se o přenosnou aplikaci pro číšníky, která slouží jako tenký klient. Na tomto klientovi je možné vytvořit nový zákaznický účet, přiřadit mu číslo stolu, přidat objednávky, zaplatit objednávky, uzavřít účet, či přesouvat objednávky mezi účty. Klient je bezstavový a při každé změně aktivity se data načítají znovu přes rozhraní REST ze serveru.



Obrázek 2.4: Aplikace pro Android

REST

Aplikace ke komunikaci používá rozhraní REST (Representational state transfer) - jedná se o architekturu, která staví na HTTP volání GET a POST a přidává ještě DELETE a PUT, tedy kompletní CRUD model. Přenášená data jsou reprezentována pomocí zdrojů, kde každý má svou HTTP adresu (URI)[res].

REST	CRUD	Popis
POST	Create	Vytvoření nového zdroje
PUT	Update	Úprava existujícího zdroje
GET	Retrieve	Získání zdroje
DELETE	Delete	Smazání zdroje

Tabulka 2.1: Popis REST metod[Zá16]

2.3 Platforma ASP.NET Core

ASP.NET Core (dále jen framework Core) je multiplatformní open-source framework který kombinuje ASP.NET MVC (pro webové aplikace) a ASP.NET Web API (REST). Nejedná se tedy pouze o programovací jazyk - toto je tedy největší rozdíl oproti čisté Javě, ve které je psána původní aplikace. ASP.NET Core totiž nespecifikuje ani použitý jazyk, aplikace je možné kromě ovyklého C# psát například i v C++, nicméně pro tento jazyk nejsou k dispozici žádné automatizované nástroje[cor].

2.3.1 Modely entit

Framework Core využívá jako základní stavební kameny aplikace tzv. modely, které, jak již název napovídá, modelují jednotlivé entity (objekty), se kterými

aplikace pracuje. Jednotlivé modely mají přiřazené svoje atributy, jako například ID, název a podobně. Aplikace .NET standardně nepoužívají DTO objekty, a mapování na databázi probíhá plně automaticky dle použitého typu. Toto je dále samozřejmě možné specifikovat - jako např použití data v databázi místo stringu[cor].

```
namespace CashBobRest.Models {
    public class AccountItem {
        public long Id { get; set; }
        public int Table { get; set; }
        public string Name { get; set; }
        public bool Opened { get; set; }
        public long Createdate { get; set; }
        public List<OrderItem> Orders { get; set; }
    }
}
```

Obrázek 2.5: Entita Account

Jak je z obrázku patrné, entita může obsahovat nejenom primitivní typy, ale parametrem může být i jiná entita a lze takto jednoduše modelovat relace mezi nimi. Dále je zajímavé, že na rozdíl od Javy, get a set metody není nutné explicitně vypisovat, což se významně podílí na zkrácení ručně psaného kódu a také celkové přehlednosti.

LinQ

Další výhodou .NET frameworku je možnost použití LinQ dotazů, které simulují jazyk SQL, ale místo dotazování se nad tabulkami se odkazujeme pomocí tzv. lambda výrazů přímo pomocí atributů entit - toto zjednodušuje přístup k datům, jelikož není nutné rozlišovat jejich původ, jak jsou uložena[lin].

2.3.2 Databázový kontext

Dle relací modelů je automaticky vytvořen databázový kontext, který je spravovaný tzv. Entity frameworkem. Databázový kontext může vypadat takto:

Jak je na obrázku vidět, jednotlivé entity jsou lehce přístupně pomocí databázových setů, nad kterými je možné provádět filtrovací operace (where, order by, atd.).

2.3.3 Controller

REST adresy jsou v Core frameworku routovány pomocí controllerů, přičemž jednotlivým metodám jsou přiřazeny adresy pomocí tokenů v hranatých závorkách:

```
namespace CashBobRest.Models {
    public class DatabaseContext : DbContext {

        public DatabaseContext(DbContextOptions<DatabaseContext> options)
        }

        public DbSet<AccountItem> AccountItems { get; set; }
        public DbSet<OrderItem> OrderItems { get; set; }
        public DbSet<ItemItem> ItemItems { get; set; }
        public DbSet<TableItem> TableItems { get; set; }
    }
}
```

Obrázek 2.6: Databázový kontext aplikace

2.3.4 .NET Identity

Autentizace a autorizace jsou ve frameworku ASP.NET Core standardizovány a maximálně zjednodušeny za pomocí .NET identity - není nutné psát žádný vlastní kód, pouze v konfiguračním souboru aplikace specifikovat, které zdroje musí být být před přístupem autorizovány a to platí jak pro Razor stránky, tak i pro REST URI.

2.4 Závěrečné srovnání

Pokud uvažujeme srovnání ASP.NET Core a kombinaci Java + Spring, oba frameworky umožňují stavbu moderní aplikace s podporou REST a nasaditelnou na Cloudové platformě - Jelikož je ale nutné backend aplikace přepracovat tak, aby umožňoval provoz několika uživatelů zároveň a splňoval parametry pro podporu EET a také z důvodu toho, že původní aplikace je v současné podobě psána pomocí frameworku Play!, který je mnohem méně rozšířen, rozhodl jsem se aplikaci přesunout na platformu ASP.NET Core. Další důvody pro přesun k ASP.NET Core:

ASP.NET Core je také open-source. Z první analýzy frameworku jsem zjistil, že vývoj ASP.NET Core není uzavřený, ale jedná se o open-source projekt - narozdíl od původního ASP.NET.

Rychlost prototypování. Jak bylo popsáno v předchozí sekci, narozdíl od Javy a frameworku Spring je ke zprovoznění aplikace nutné jen minimum konfigurací - framework je optimalizován pro rychlé prototypování.

Autorizace. Implementace autentizace a autorizace je ve frameworku ASP.NET Core velmi snadná, díky .NET Identity

Vlastní zájem. CashBob je studentský projekt, který již za dobu svého vývoje doznal spousty změn, jmenovitě například přesun od lokální desktopové aplikace postupně ke kombinaci více modulů komunikujících přes rozhraní Java RMI[rmi] až k zavedení standardizovaného multiplatformního API REST.

Bohužel, nejnovější iterace aplikace je sestavena pomocí nástrojů, které nejsou běžně používány a při mých vlastních pokusech o překlad jsem se potýkal s problémy. Po konzultaci s vedoucím práce jsem se tedy rozhodl převést aplikaci k Microsoft platformě .NET - ještě v rámci semestrálního projektu, který předcházel tuto diplomovou práci. Dalším hlavním důvodem k tomuto kroku je z mé strany také zájem o to, naučit se pracovat s platformou .NET a také obecně s ekosystémem Microsoftu, se kterým jsem se v rámci výuky vůbec nesetkal.

Kapitola 3
Návrh

Aby bylo možné začít navrhovat strukturu modelů a REST URI aplikace, je nutné nejdříve stanovit funkční a nefunkční požadavky na nový server. Rozhodl jsem se, že se nejdříve omezím na tvorbu Backendu a proto se jej následující požadavky týkají téměř výhradně. U požadavků dále vycházím z již existujících aplikací a jelikož nemám k dispozici zdrojové kódy, snažil jsem se je určit podle vlastních poznatků z předchozí bakalářské práce a také z výsledků nového průzkumu zmíněné desktopové aplikace.

Vzhledem k tomu, že jsem začal na projektu pracovat bez jakýchkoliv znalostí platformy .NET, složitosti jednotlivých požadavků jsem nemohl určit, jelikož jsem nevěděl, kolik času a úsilí bude třeba k získání potřebných znalostí – vše jsem se musel nejdříve prostudovat v uživatelské dokumentaci. Dokonce i samotný programovací jazyk C# byl pro mě úplnou novinkou.

Aplikace bude koncipována jako cloudové řešení - tedy uživatel nebude mít aplikaci nainstalovanou na svém zařízení, ale bude k ní přistupovat přes webové, nebo REST rozhraní - to by bylo realizováno pomocí dodávané android aplikace pro číšníky, která obsahuje veškerou funkcionalitu pro chod podniku. Úskalí tohoto řešení je závislost na internetovém připojení. Jelikož ale samotný systém EET internetové připojení stejně vyžaduje[1], nevidím to osobně jako překážku. Následující

3.1 Funkční požadavky

RQ 1 – Perzistence

Zadavatel	Pavel Zářecký
Priorita	Vysoká

Aplikace bude automaticky ukládat veškeré potvrzené změny provedené v datech pomocí REST rozhraní, protože data nesmí být v nekonzistentním stavu (při výpadku proudu by došlo ke ztrátě pracovních dat)

[1] Existuje i tzv. offline režim[eetb]

RQ 1.1 – Ukládání všech objednávek

Zadavatel	Pavel Zářecký
Priorita	Vysoká

Aplikace bude ukládat historii objednávek pod jednoznačným ID za účelem automatizovaného účetnictví a pro potřeby EET (např. tzv. offline režim)

RQ 2 – Správa sortimentu

Zadavatel	Pavel Zářecký
Priorita	Vysoká

Aplikace bude umožňovat správu sortimentu, jelikož se jedná o nedílnou součást aplikací podobného zaměření.

RQ 2.1 – Správa položek menu

Zadavatel	Pavel Zářecký
Priorita	Vysoká

Systém bude umožňovat výběr aktuální nabídky a její kategorizaci do menu, aby byl umožněn co nejrychlejší výběr číšníkem za provozu podniku.

RQ 2.2 – Správa skladu

Zadavatel	Pavel Zářecký
Priorita	Nízká

Aplikace bude umožňovat správu skladových zásob k ulehčení inventury.

RQ 3 – Manipulace objednávek a účtů

Zadavatel	Pavel Zářecký
Priorita	Vysoká

Aplikace bude umožňovat vytváření nových objednávek, jelikož se jedná o nedílnou součást aplikací podobného zaměření.

RQ 3.1 – Platba zboží

Zadavatel	Pavel Zářecký
Priorita	Vysoká

Aplikace bude automaticky ukládat veškeré potvrzené změny provedené v datech pomocí REST rozhraní, protože data nesmí být v nekonzistentním stavu (při výpadku proudu by došlo ke ztrátě pracovních dat)

RQ 3.2 – Jednoznačné určení objednávek za provozu

Zadavatel	Pavel Zářecký
Priorita	Vysoká

Aplikace bude umožňovat číšníkovi označit objednávku tak, aby ji bylo možné za provozu jednoznačně určit (jméno, číslo stolu, ...).

RQ 3.3 – Přesun objednávek mezi účty

Zadavatel	Pavel Zářecký
Priorita	Vysoká

Bude možné přesouvat nezaplacené objednávky mezi aktivními účty, protože zákazník může chtít zaplatit část účtu samostatně.

RQ 4 – Granularita oprávnění

Zadavatel	Pavel Zářecký
Priorita	Vysoká

Aplikace bude rozlišovat několik úrovní oprávnění, aby číšníci nemohli měnit citlivé informace.

RQ 5 – Souběžnost přihlášených uživatelských účtů

Zadavatel	Pavel Zářecký
Priorita	Vysoká

Aplikace bude muset podporovat spoustu přihlášených uživatelských účtů, protože se bude jednat o centralizované cloudové řešení nabízené více zákazníkům výsledného produktu

3.2 Nefunkční požadavky

RQ 5 – Platforma ASP.NET

Zadavatel Martin Komárek

Aplikace bude implementována na platformě ASP.NET Core v jazyce C#, jelikož užití tohoto frameworku vyplynulo z analýzy.

RQ 6 – Rychlost operací

Zadavatel Pavel Zářecký

Aplikace musí být optimalizovaná pro rychlost, aby nedocházelo ke zdržování zákazníku pomalými reakcemi klientské aplikace

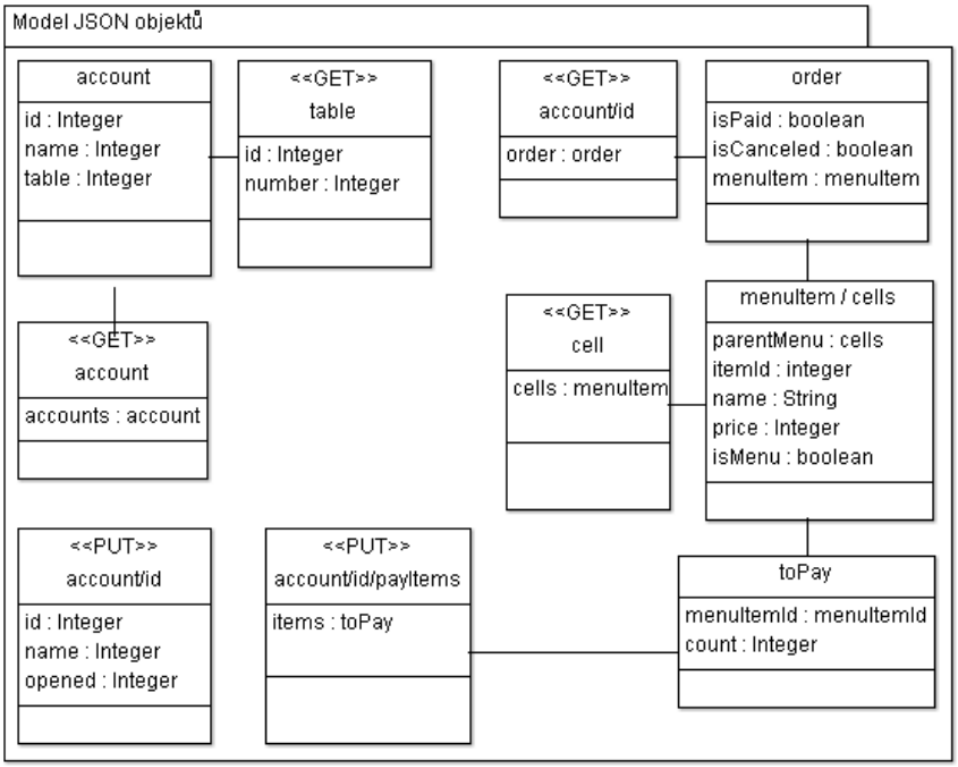
Kapitola 4

Implementace

4.1 Implementace základních entit

4.1.1 Návrh

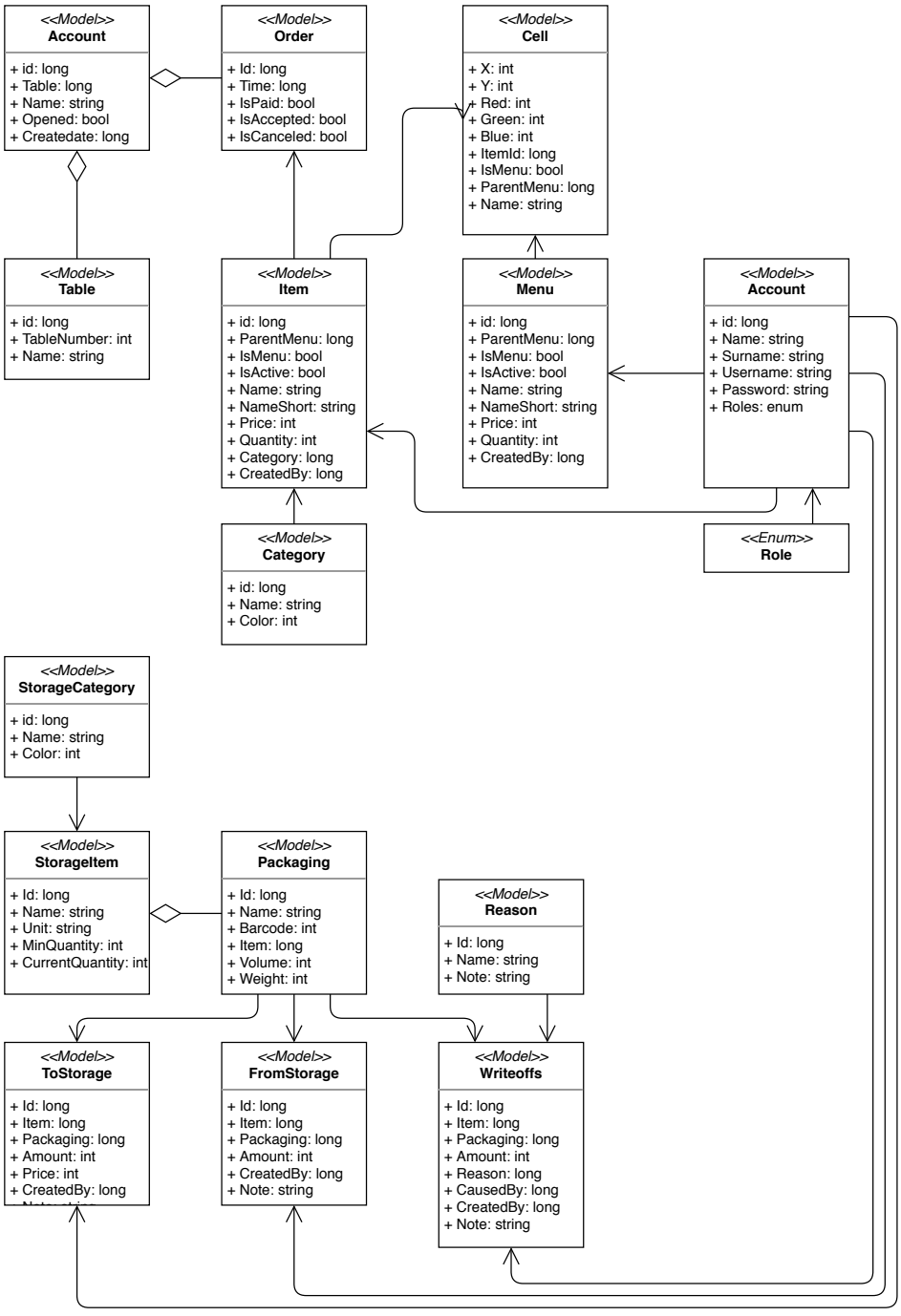
Při implementaci rozhraní REST ve frameworku Core jsem využil automatického mapování JSON objektů na modely tak, jak jsem již zmínil v kapitole 3 – Jednotlivé modely jsem dále vytvářel tak, aby co nejpřesněji kopírovaly původní rozhraní, které mělo následující strukturu:



Obrázek 4.1: Struktura REST API původní aplikace

V první iteraci vývoje tedy byly vytvořeny základní modely entit tak, aby kopírovaly původní aplikaci a bylo tak možné zprovoznit komunikaci s

Aplikací pro číšníky. Samotné entity aplikace byly prozatím také navrženy tak, aby kopírovaly původní stav:



Obrázek 4.2: Návrh entit aplikace

4.1.2 Výsledný stav

Výsledkem iterace je tedy stav, ve kterém jsou funkční následující REST rozhraní:

Manipulace účtů

Adresa: Rest/Account

Metoda: GET

Vrátí seznam všech otevených účtů, neobsahuje nadbytečná data jako seznamy objednávek každého účtu.

Adresa: Rest/Account/id

Metoda: GET

Vrátí informace o účtu specifikovaného id. JSON objekt již obsahuje i seznam otevřených objednávek.

Adresa: Rest/Account/id

Metoda: POST

Vytvoří nový účet

Adresa: Rest/Account/id/payItems

Metoda: POST, PUT

Přebírá seznam objednávek k zaplacení a tyto objednávky označí a archivuje na straně serveru.

Adresa: Rest/Account/id/order

Metoda: POST, PUT

Přebírá seznam objednávek a přidá jej na účet ve stavu přijato/nezaplaceno.

Adresa: Rest/Account/id

Metoda: PUT

Změní specifikovaná pole účtu, nemění seznam objednávek, ty proto mohou být v JSON objektu prázdné.

Adresa: Rest/Account/id

Metoda: DELETE

Smaže účet. Metoda přidána jen pro úplnost CRUD operací, účty by se mazat nikdy neměly z archivačních důvodů.

Manipulace inventáře

Adresa: Rest/Item/

Metoda: GET

Vrátí seznam všech aktivních položek menu.

Adresa: Rest/Item/id

Metoda: GET

Vrátí informace o položce specifikovaného id.

Adresa: Rest/Item/

Metoda: POST

Vytvoří novou položku.

Adresa: Rest/Item/id

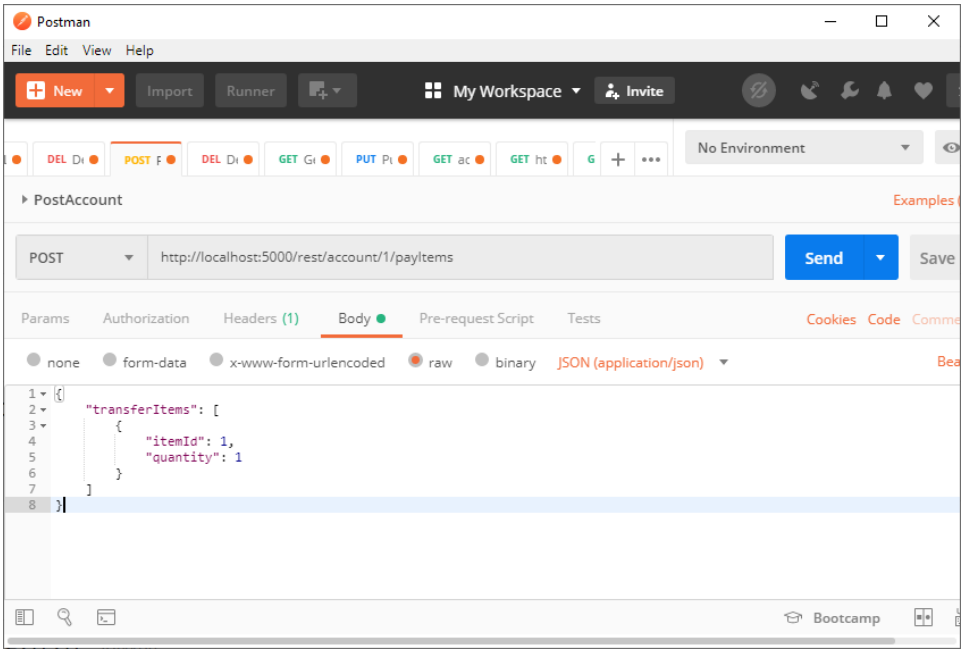
Metoda: PUT

Upraví položku. Jelikož se archív objednávek na tyto položky odkazuje, neměl by nikdy být měněna celá položka, pouze provedeny mírné úpravy. Každá objednávka má proto specifikovanou vlastní cenu, aby nebylo ovlivněno účetnictví podniku.

Adresa: Rest/Item/id

Metoda: DELETE

Opět, pouze pro úplnost.



Obrázek 4.3: Testování platby

4.1.3 Testování API

Testování v této fázi proběhlo za použití nástroje Postman, který umožňuje sestavovat vlastní JSON objekty a zasílat je na jenotlivá REST URI a také získat odpověď.

4.1.4 Závěr iterace

Výsledkem implementace je hotový backend serverové aplikace, který plně komunikuje s původní Android aplikací. V té musely být provedeny některé změny, jako například úpravy v očekávaných kódech, změna struktury objektů pro JSON parser tak, aby aplikace dokázala pracovat s nově vytvořenými modely.

Neimplementované požadavky

Aplikace prozatím neumožňuje přesun objednávek mezi účty, jelikož by to vyžadovalo větší zásah do Android aplikace. Dále pak ještě není implementován backend pro tisk objednávek, který jsem ale zatím nezařadil mezi funkční požadavky.

Kapitola 5

Závěr a zhodnocení prace

Závěrem bych rád zhodnotil výsledek a míru splnění zadání této diplomové práce. Jak je již z implementační části zjevné, nebyl jsem schopen splnit některé části zadání - bohužel, z osobních důvodů a jimi způsobenému nedostatku času jsem nebyl schopen dále pokračovat v implementaci požadavků. Výsledkem práce je tedy funkční a otestovaný backend aplikace se základním uživatelským rozhraním - v průběhu implementace poslední iterace se ukázalo, že automatická generace kódu pro CRUD interface bohužel nebere v potaz ne-primitivní atributy a tedy ani vazby mezi entitami - úplný přístup ke vší funkcionalitě je proto možný pouze skrze REST rozhraní. Dále nebylo plně implementováno odesílání evidence tržeb daňové správě a tedy ani tisk účtenek. Nakonec nebyla provedena úplná analýza konkurenčních řešení.

5.1 Nasazení aplikace

Aplikace je v současném stavu nasazena na následující internetové adrese:

```
http://zarecky.aspfree.net
```

Je samozřejmě možné se k ní připojit z upravené verze Android aplikace, která je také součástí výstupu této diplomové práce.

Literatura

[clo] *Cloud computing*, https://cs.wikipedia.org/wiki/Cloud_computing, [Online; navštíveno 22.5.2019].

[cor] *Introduction to asp.net core*, https://docs.microsoft.com/cs-cz/aspnet/core/?view=aspnetcore-2.2, [Online; navštíveno 22.5.2019].

[Če16] Tomáš Červenka, *Webové rozhraní restauračního systému*, 2016.

[eeta] *Eet popis rozhraní*, https://www.etrzby.cz/assets/cs/prilohy/EET_popis_rozhrani_v3.1.1.pdf, [Online; navštíveno 22.5.2019].

[eetb] *etržby- elektronická evidence tržeb*, https://www.etrzby.cz/, [Online; navštíveno 17.5.2019].

[Hog16] Tomáš Hogenauer, *Bakalářská práce tomáše hogenauera*, 2016.

[int] *Vývoj internetu v Čr za posledních 10 let*, https://www.lupa.cz/pr-clanky/vyvoj-internetu-v-cr-za-poslednich-10-let/, [Online; navštíveno 22.5.2019].

[jav] *Spring framework 5*, https://spring.io/, [Online; navštíveno 22.5.2019].

[lin] *Linq*, https://cs.wikipedia.org/wiki/LINQ, [Online; navštíveno 22.5.2019].

[pok] *Nejlepší eet pokladna 2019*, https://www.5nej.cz/srovnani-eet-pokladen/, [Online; navštíveno 22.5.2019].

[res] *Rest api tutorial*, https://restfulapi.net/rest-architectural-constraints/, [Online; navštíveno 22.5.2019].

[rmi] *Java remote method invocation*, https://cs.wikipedia.org/wiki/Java_remote_method_invocation, [Online; navštíveno 22.5.2019].

[soa] *Soap*, https://en.wikipedia.org/wiki/SOAP, [Online; navštíveno 22.5.2019].

[Zá16] Pavel Zářecký, *Vývoj aplikací na platformě android*, 2016.

Příloha A

Seznam příloh

Zdrojové kódy aplikace. sources.zip

Tato práce ve formátu PDF. thesis-zarecky.pdf