České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra počítačů

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Jan Dryk**

Studijní program: Softwarové technologie a management
Obor: Softwarové inženýrství

Název tématu: **Mobilní klient restauračního systému**

Pokyny pro vypracování:

Pomocí nástroje Xamarin vytvořte multiplatformní aplikaci pro systémy Android a iOS, která bude pomocí rozhraní typu REST spolupracovat s existujícím restauračním serverem CashBob. Práci průběžně analyzujte, implementujte, testujte, nasazujte a dokumentujte. Výsledná aplikace bude především umožňovat:
Vytvářet a uzavírat účty.
Přesouvat položky mezi účty.
Vytvářet objednávky a rychloobjednávky.
Placení účtů či jejich částí.
Tisknout účtenky na serveru
Tisknout účtenky na mobilní tiskárně.

Seznam odborné 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] FOWLER, Martin. Destilované UML. Praha: Grada, 2009, 173 s. ISBN 978-80-247-2062-3.
[3] Xamarin - http://xamarin.com/
[4] Jenkins - http://jenkins-ci.org/

Vedoucí: Ing. Martin Komárek

Platnost zadání: do konce letního semestru 2015/2016

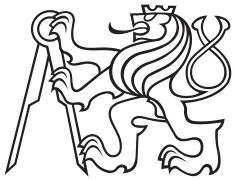
L.S.

doc. Ing. Filip Železný, Ph.D.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 25. 3. 2015

Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Mobilní klient restauračního systému

Jan Dryk
Softwarové technologie a management

20. května 2015
Vedoucí práce: Ing. Martin Komárek

Poděkování / Prohlášení

V první řadě bych chtěl poděkovat svému vedoucímu, Ing. Martinu Komárkovi, za poskytnutí konzultací a cenných připomínek. Dále bych chtěl poděkovat své rodině za podporu při studiu i během psaní této práce. V poslední řade děkuji za spolupráci a rady všem členům týmu projektu CashBob.

Prohlašuji, že jsem práci vypracoval samostatně a použil jsem pouze podklady uvedené v přiloženém seznamu.
Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne 20. května 2015

.....

Abstrakt / Abstract

Cílem práce je vytvořit klientskou aplikaci pro restaurační systém CashBob. Tato aplikace umožní číšníkům vytvářet objednávky a tisknout účtenky přímo u zákazníka, šetříc tak jejich čas i energii. Aplikace byla navržena pro mobilní zařízení s důrazem na tablety a byla vyvinuta pro platformy Android a iOS pomocí nástrojů pro tvorbu multiplatformních aplikací od firmy Xamarin.

The aim of this thesis is to create a client application for a restaurant management system CashBob. This application will help the waiters create orders and print receipts directly at the customer, saving their energy and time. The app was designed for mobile devices with emphasis on tablets. It was build for devices running Android and iOS using multiplatform development tools from Xamarin.

Kapitola 1
Úvod

V dnešní době tablety a telefony velmi rychle nahrazují klasické počítače, díky jejich ceně, mobilitě a jednoduchosti ovládání. Dostatečně výkonný tablet se dá pořídit za zlomek ceny osobního počítače a nabízí operační systém a hardware uzpůsobený k ovládání prstem. Mezi mobilními zařízeními je pro vývojáře největším rozdílem operační systém. Zařízení nabízené firmou Apple mají jejich vlastní operační systém iOS, zatímco většina ostatních výrobců volí volně dostupný Android. Oba tyto operační systémy jsou založeny na velmi odlišných technologiích a programovacích jazycích.

Vyvíjet dvě různé aplikace pro každou platformu je ale časově náročné a vznikají pak rozdíly mezi implementacemi, které by mohly vést k zmatení číšníků. Také funkce v aplikaci musí být snadno a rychle dostupné, aby aplikace neomezovala zaměstnance v práci.

1.1 CashBob

Systém CashBob je komplexní řešení pro restaurace, bary či kavárny. Umožňuje podnikům spravovat objednávky a tisknout účtenky, spravovat zaměstnance a obsahuje i modul pro správu skladu.

Dlouhodobě jej vyvíjí katedra počítacu v rámci různých týmových projektů či bakalárských prací. Systém je napsán v jazyce Java a je rozdělen na klientskou a serverovou část. Ty spolu komunikují pomocí RESTového rozhraní na serverové části. Uživatelské rozhraní klientské části je optimalizováno pro dotykové pokladní terminály.

V minulých letech vznikla podobná aplikace pro Android v rámci několika bakalářských prací. Poslední verze aplikace byla aktualizována v roce 2013 a byla napsána pro Android 2.1 Aplikace má sice mnoho funkcí, ale bohužel je zastaralá a neodpovídá množině funkcí, které nabízí nové verze desktopové aplikace CashBob.

Cílem této práce je navrhnout aplikaci, která bude fungovat na více platformách a bude snadněji udržovatelná. Dále by měla být schopna sloužit jako vzor pro tvorbu nových multiplatformních aplikací pomocí nástrojů od

společnosti Xamarin. Aplikace by měla sloužit jako doplněk systému CashBob a usnadnit tak práci jejím uživatelům. Stávající aplikace pro mobilní zařízení nejsou dostupné pro platformu iOS, takže rozšiřuje i portfolio aplikací, které obsahuje komplexní řešení CashBob.

1.2 Obsah práce

V kapitole 2 jsou popsány jednotlivé cíle této práce.

V kapitole 3 jsou analyzovány podklady pro návrh aplikace.

V kapitole 4 je popsáno jak zprovoznit CashBob Server v cloudu pro snadné testování aplikace.

V kapitole 5 je popsáno jak fungují nástroje Xamarin a framework Xamarin.Forms, na kterých je tato práce založená.

V kapitole 6 je popsána struktura projektů, architektura aplikace a její hlavní části.

V kapitole 7 je popsána nejsložitější architektonická část aplikace, a to tištění účtenek různými tiskárnami.

V kapitole 8 je popsáno jak jsem testoval aplikaci a jaké byly výsledky testování.

Kapitola 9 popisuje výsledek práce, zkušenosti s nástroji Xamarin a doporučení do budoucnosti.

1.3 Doporučené znalosti

Bude předpokládáno, že čtenář této práce má znalosti vývoje aplikací v jazyce C# a má povědomí o architekturách platforem iOS a Android. Dále je vhodná znalost modelovacího jazyku UML.

Kapitola 2
Specifikace cíle

V této kapitole budou podrobně shrnuty cíle bakalářské práce. Jedná se především o tvorbu aplikace a její funkce.

2.1 Hlavní cíle

- **Vytvoření aplikace pro OS Android**
  Bude vytvořena nová aplikace pro OS Android v jazyce C#, která využije sdíleného jádra. Aplikace bude obsahovat pouze kód nutný k využití jádra a kód, který je specifický pro platformu Android.

- **Vytvoření aplikace pro OS iOS**
  Bude vytvořena nová aplikace pro OS iOS v jazyce C#, která využije sdíleného jádra. Aplikace bude obsahovat pouze kód nutný k využití jádra a kód, který je specifický pro platformu iOS.

- **Vytvoření sdíleného jádra**
  Sdílené jádro bude obsahovat logiku aplikace, uživatelské rozhraní a komunikaci se serverem. Aplikace bude vytvořena pomocí nástrojů Xamarin. Nelze použít zdrojový kód stávajících aplikací, protože je napsaný v jiném programovacím jazyce.

- **Rozšíření serverové aplikace**
  Aplikace neobsahuje v současnosti všechny funkce potřebné k zprovoznění některé funkcionality mobilní aplikace. Serverová aplikace bude o tyto funkce doplněna. Jedná se zejména o podporu tisku se serverem.

- **Otestování výstupu práce**
  Jednotlivé části aplikace, především komunikační služby, budou otestovány pomocí unit testů. Celková funkcionalita aplikace bude otestována pomocí UI testů. Kód bude podroben navíc statické analýze.

2.2 Katalog požadavků

1. **Aplikace umožní přihlášení pomocí uživatelského jména a hesla**
   Authentikaci provádí serverová aplikace. Přihlášení je povinné, protože server umožňuje přístup k RESTovému rozhraní pouze authorizovaným uživatelům.

2. **Aplikace umožní zobrazení objednávek na účtu**
   Uživatel si bude moci prohlížet účty a jejich nezpracované objednávky.

3. **Aplikace umožní vytváření nových účtů**
   Uživatel bude moci vytvořit rychloobjednávku, která vytvoří účet s vygenerovaným jménem, nebo vyplnit formulář s údaji pro vytvoření nového účtu.

4. **Aplikace bude umožňovat přesun položkek mezi účty**
   Aplikace bude umožňovat přesun konkrétních objednávek mezi volitelnými účty.

5. **Aplikace bude umožňovat placení účtů či jejich částí**
   Aplikace bude umožňovat zaplacení konkrétních objednávek nebo zkratku pro zaplacení všech objednávek v daném účtu. Dále bude obsahovat zkratku pro přesunutí všech položek.

6. **Aplikace bude umožnovat vytváření objednávek**
   Aplikace bude obsahovat panelové menu, stejně jako v desktopové aplikaci CashBob pro optimální ovládání aplikace. Objednávané položky se budou řadit do seznamu, který může být upraven, před odesláním.

7. **Aplikace bude umožňovat tisk účtenek pomocí CashBob Serveru**
   Aplikace zašle na server požadavek o vytištění účtenky. Ta bude vytištěna na výchozí tiskárnu.

8. **Aplikace bude umožňovat tisk účtenek pomocí tiskárny Star SM-T300**
   Jako mobilní tiskárna byla vybrána tiskárna Star SM-T300, která bude během implementace k dispozici. Systém se bude k tiskárně připojovat pomocí Bluetooth.

9. **Aplikace bude mít přehledné uživatelské rozhraní**
   Aplikace nesmí číšníky zpomalovat, musí být snadno a rychle navigovatelná s minimální chybovostí. Uživatelské prostředí bude optimalizováno pro tablety s velikostí displaye 7 palců, ale bude podporovat zařízení velikosti od mobilních telefonů po 10 palcové tablety.

10. **Aplikace bude komunikovat se serverem pomocí webové služby REST**
    Klient současné verze desktopové aplikace CashBob komunikuje se serverem pomocí RESTového rozhraní. Mobilní aplikace bude komunikovat pomocí stejného rozhraní.

11. **Aplikace bude dostupná pro OS Android úrovně API 20 (verze 4.4)**
    Protože aplikace bude podporovat tisk účtenek pomocí nativních funkcí OS Android, které jsou dostupné od Android verze 4.4, bude aplikace podporovat zařízeni s OS Android 4.4 a vyšší.

12. **Aplikace bude dostupná pro OS iOS verze 7 a vyšší**
    Vzhledem k tomu že, více jak 98% uživatelů má nainstalovanou verzi iOS 7 nebo iOS 8, bude tak bude dostupná pro naprostou většinu uživatelů.

Kapitola 3
Analýza

3.1 Verze operačních systémů

Tato kapitola popisuje mojí logiku při rozhodování, které verze operačních systému budou podporovány touto aplikací. Při rozhodování, které verze operačních systémů budou podporovány, byla samozřejmě brána v úvahu jejich adopce uživateli včetně nabízených funkcí.

3.1.1 Android

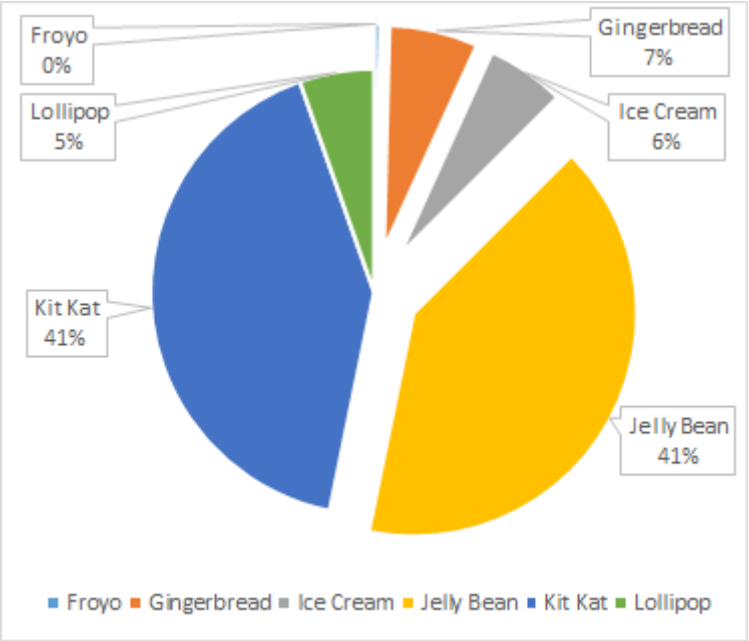
Podle průzkumu internetových stránek 9to5google [1] má 46.8% zařízení verzi Android 4.4 a vyšší. V případě Androidu jsem přistoupil k agresivnější strategii, protože se domnívám, že přístroje se starými verzemi nejsou v cílové skupině zákazníků aplikace CashBob. Vzhledek k tomu, že je aplikace optimalizovaná pro tablety, a Android podporuje tablety teprve od verze 3.0, ale pořádně až 4.0. Vzhledem k tomu, že Android podporuje tablety až od verze 3.0, Dále stará zařízení se staršími verzemi nejsou většinou dostatečně svižná, aby je mohl používat číšník bez toho aby ho zařízení zpomalovalo v práci. Podniky adoptující moderní pokladní systém CashBob budou pořizovat pro zaměstnance tablety nové a myslím si, že podporovat verze starší než verzi 4.4, která vyšla roku 2013 není nutné. Nicméně hlavní důvod, proč jsem ale zvolil Android 4.4 Kit Kat jako minimální verzi je, že od této verze Android obsahuje API pro tisk pomocí operačního systému.

3.1.2 iOS

Podle průzkumu MacRumors [2] má 98% zařízení iOS verze 7 a vyšší. U iOS je adopce nových verzí tak vysoká, že s vybíráním podporované verze není žádný problém. Navíc iOS 8 nepředstavil žádné funkce, které bych potřeboval nutně využít. Aplikace bude dostupná od iOS verze 7 a výše.

Verze	Název	Úroveň API	Adopce
2.2	Froyo	8	0.4%
2.3.3	Gingerbread	10	6.4%
4.0	Ice Cream	15	5.7%
4.1	Jelly Bean	16	16.5%
4.2	Jelly Bean	17	18.6%
4.3	Jelly Bean	18	5.6 %
4.4	Kit Kat	19	41.4 %
5.0	Lollipop	21	5%
5.1	Lollipop	22	0.4%

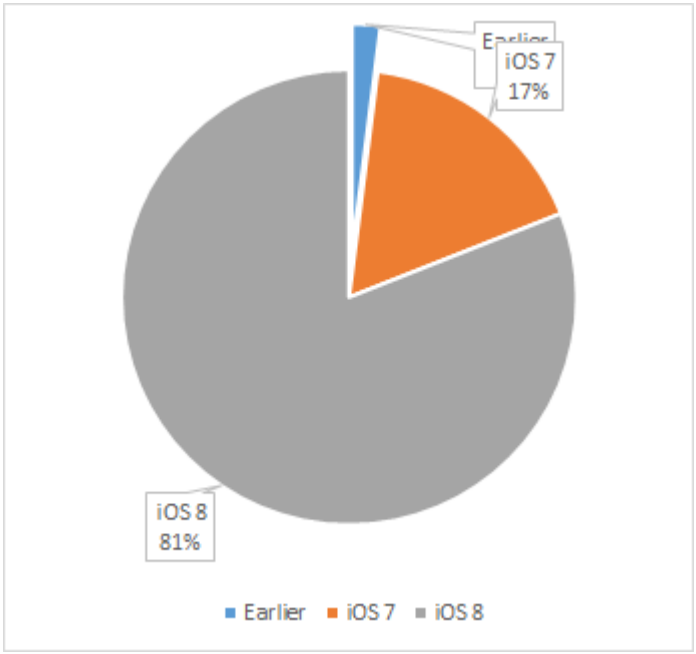
Tabulka 3.1: Rozdělení verzí operačního systému Android podle aktivních instalací



Obrázek 3.1: Rozdělení verzí operačního systému Android podle aktivních instalací

Verze	Adopce
Starší	2%
iOS 7	17%
iOS 8	81%

Tabulka 3.2: Rozdělení verzí operačního systému iOS podle aktivních instalací



Obrázek 3.2: Rozdělení verzí operačního systému iOS podle aktivních instalací

3.2 Velikost menu

V této kapitole se zabývám velikostí a počtem tlačítek v nabídce pro výběr položek k vytvoření objednávky. Součástí optimalizace uživatelského prostředí aplikace byla optimalizace menu pro mobilní zařízení. Původní menu obsahovalo příliš mnoho položek na to, aby byly zobrazeny čitelně na 7 palcovém displayi všechny najednou. Proto bylo rozhodnuto, že bude počet položek redukován.

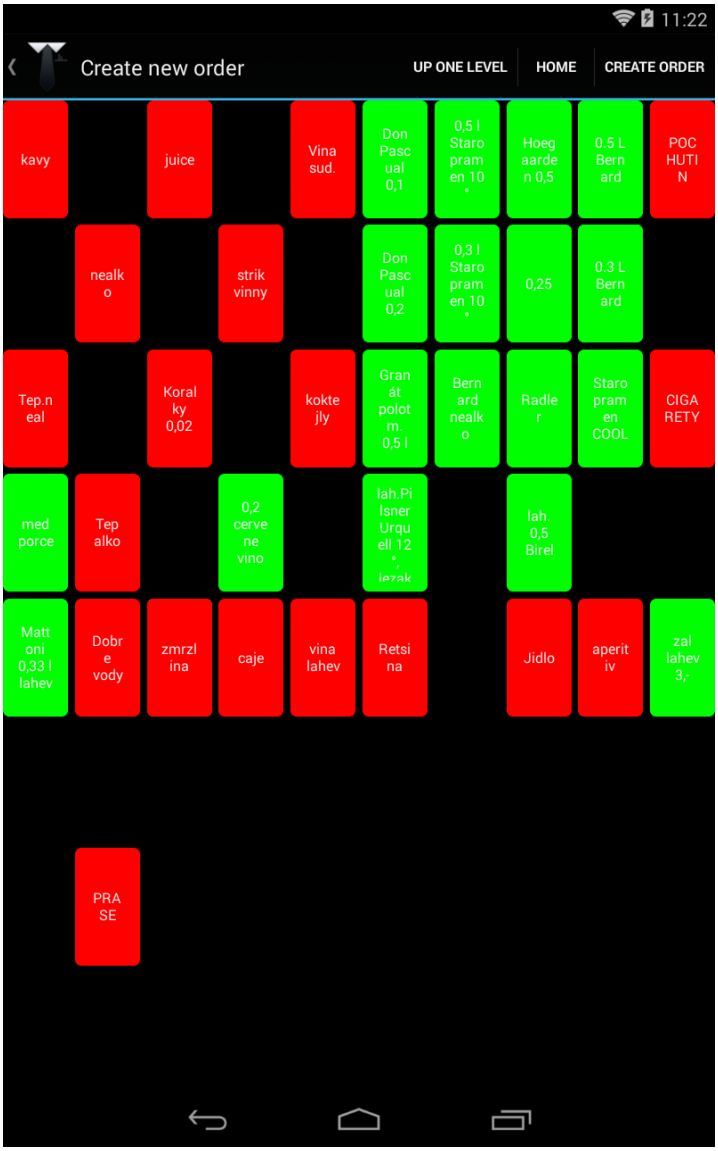
Rozměry obrazovky standardního 7 palcového tabletu jsou 151mm * 94mm. Při určování šířky tlačítek musíme brát v úvahu také čitelnost textu. Dále je cílem, aby tlačítka byla dostatečně veliká a číšnící tak mohli operovat s tabletem rychle a beze strachu, že se překliknou a udělají chybu. V aplikaci totiž mezi tlačítky nejsou mezery. Experimentálně jsem zjistil, že vhodná velikost je $2 * 2cm$.

Chceme aby tlačítka byla minimálně $2 * 2cm$ veliká, takže:

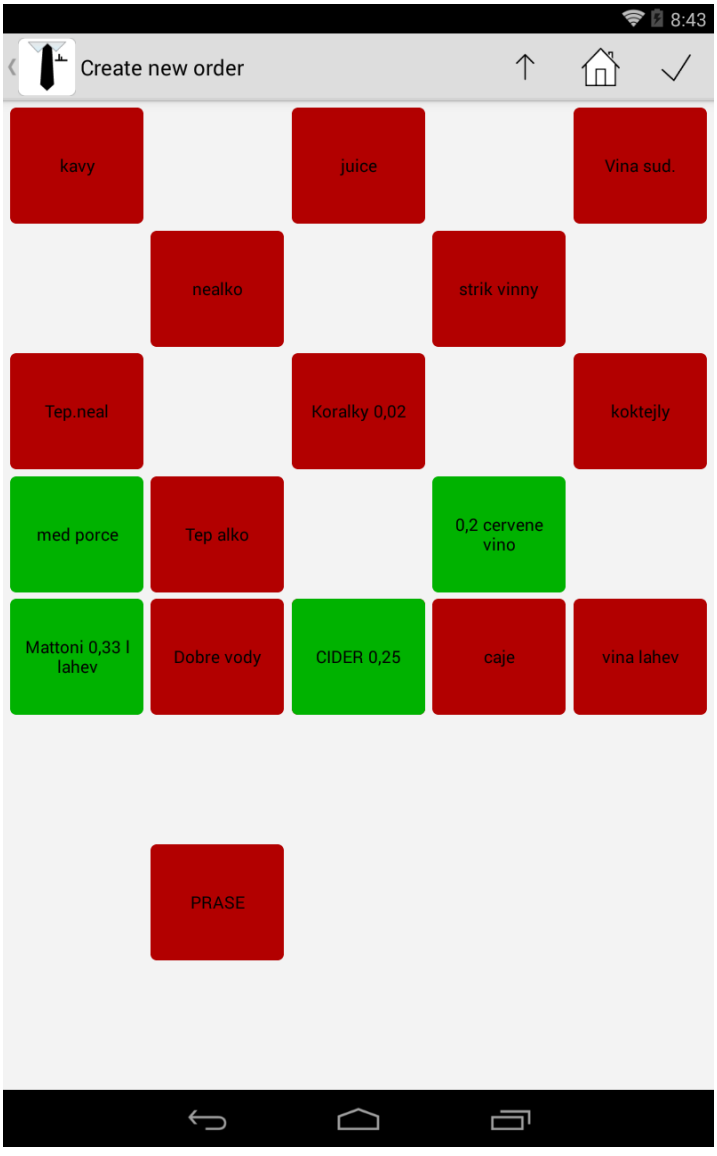
nvertikálně = $\frac{151mm}{20mm} = 7.55$ - zaokrouhlíme na 8

nhorizontálně = $\frac{94mm}{20mm} = 4.7$ - zaokrouhlíme na 5

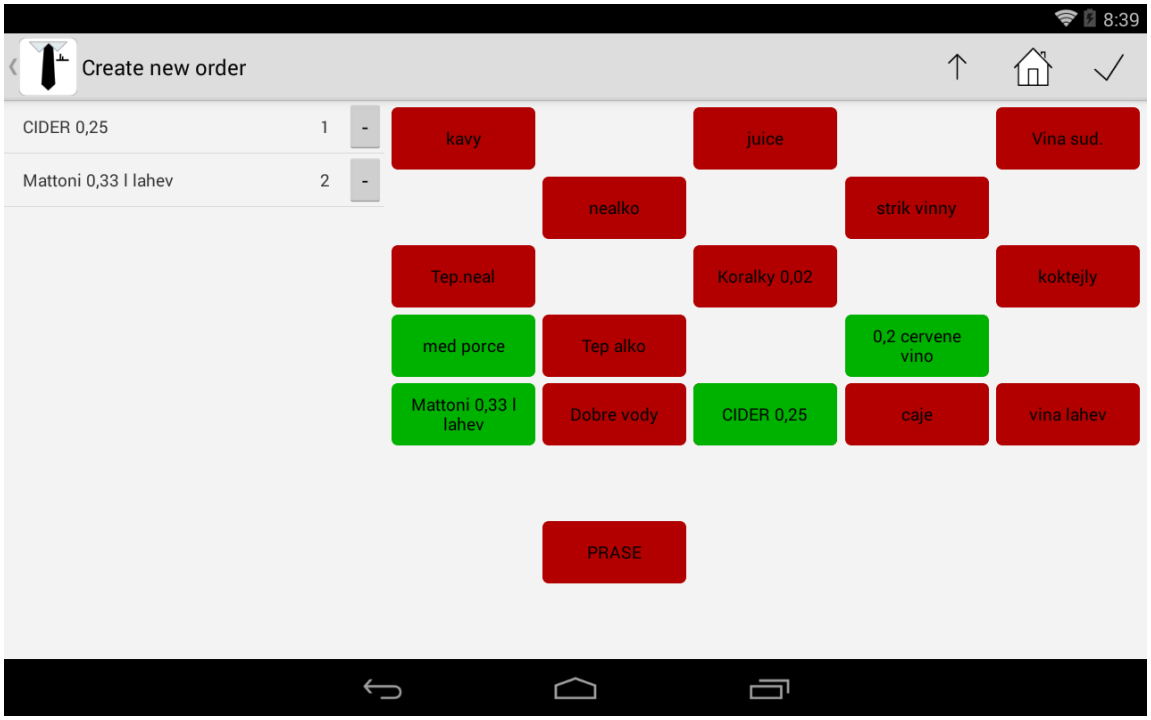
S touto velikostí je možné do dvou úrovní položek dát až $(5 * 8)^2 = 1600$ položek menu, což by mělo být více než dostačující pro každou restauraci.



Obrázek 3.3: Porovnání před úpravou velikosti tlačítek v reálné velikosti



Obrázek 3.4: Porovnání po úpravě velikosti tlačítek v reálné velikosti



Obrázek 3.5: Finální test použitelnosti nových tlačítek na šířku v reálné velikosti

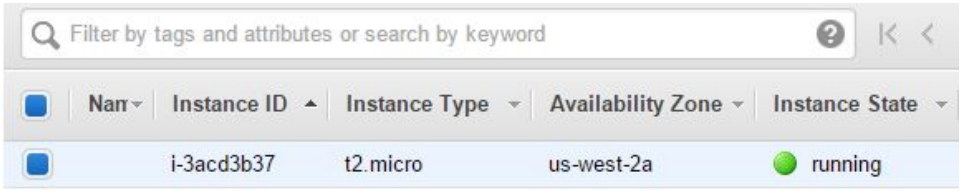
Kapitola 4
Amazon Cloud

Jedním z úkolů bylo spustit CashBob server v cloudu, aby mohl být využíván pro testování klienta na mobilních zařizeních bez počítače. Vzhledem k tomu ze CashBob je napsán jako aplikace, se zabudovaným Tomcat serverem, nelze jej nasadit jako webovou aplikaci, což by bylo pro hostování v cloudu mnohem výhodnější. Z tohoto důvodu je pro hostování CashBob Serveru potřeba celý stroj s operačním systémem. Tuto službu nabízí mnoho společností pod názvem VPS. Pro naše hostování jsem vybral nabídku EC2 firmy Amazon kvůli bezplatnému ročnímu zkušebnímu období.

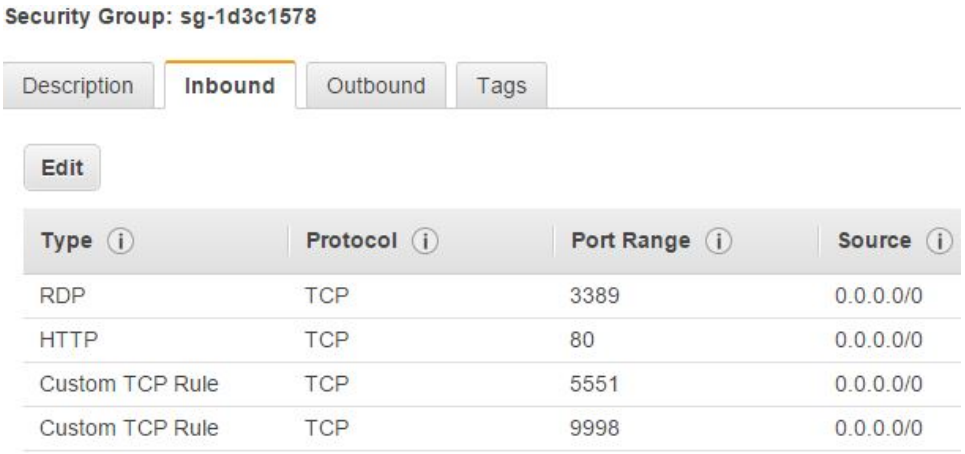
Nejdříve je nutné se zaregistrovat u firmy Amazon a vytvořit si instanci VPS. Postup pro vytvoření instance je popsaný na stránkách Amazon [3]. Po registraci je podle návodu nutno vybrat typ virtuálního stroje, jeho parametry a vytvořit si šifrovací klíč pro authentikaci. Já jsem zvolil virtuální stroj s operačním systémem Microsoft Windows Server 2012 Standard, který bylo možno vyzkoušet v rámci bezplatného zkušebního období. Po vytvoření požadavku na nový virtuální stroj bude několik minut čekat než se nainstaluje a připraví operační systém. Zda-li je stroj připraven se nám indukuje políčko Instance State.

Z důvodů bezpečnosti musíme indikovat v AWS portálu jaké porty chceme otevřít pro náš server. Tohoto docílíme vytvořením nové bezpečnostní skupiny viz obrázek 4.2.

Pokud vidíme v Instance State running můžeme se přes program Remote



Obrázek 4.1: Virtuální stroje

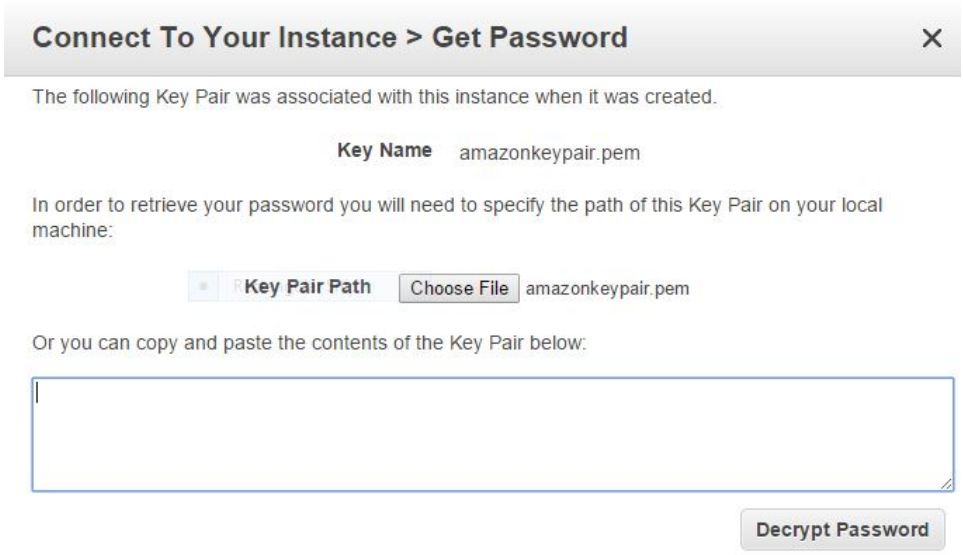


Obrázek 4.2: Security Group

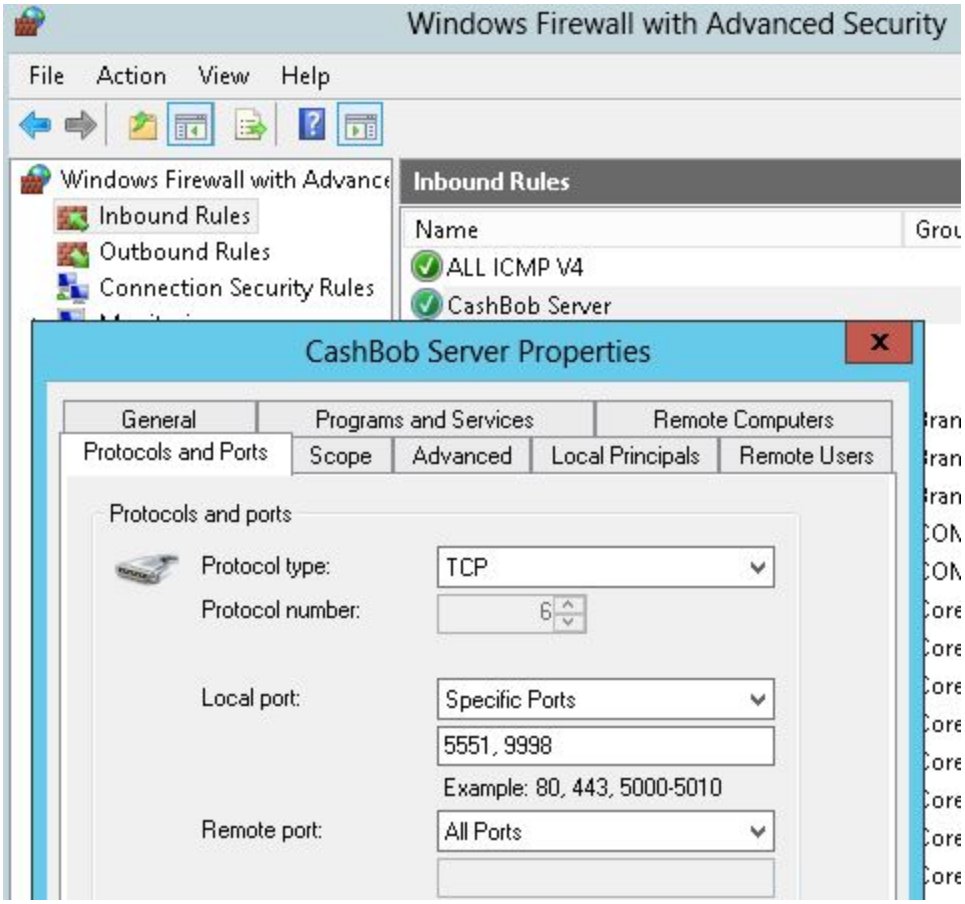
Desktop Connection, který je součástí Windows připojit k virtuálnímu stroji. Webový portál AWS nám ulehčí připojování nabídkou předpřipraveného .rdp souboru ip adresou serveru. Nyní potřebujeme šifrovací klíč, který jsme si vytvořili k odhalení hesla k administrátorskému účtu a můžeme se pustit připravování serveru pro CashBob Server.

Jako první po instalaci aktualizací musíme otevřít žádané porty ve Windows Firewall, abychom mohli přistupovat k restovému rozhraní CashBobu z internetu. Abychom porty povolili musíme vytvořit nové pravidlo, jako na obrázku 4.4. Další věc, kterou musíme udělat je nainstalovat Javu. Důležité je nainstalovat Javu 7, protože CashBob Server má problémy s restovým rozhraním, když běží pod Javou 8. Připravíme si spustitelnou verzi CashBob Serveru, umístíme jí na server a spustíme jí.

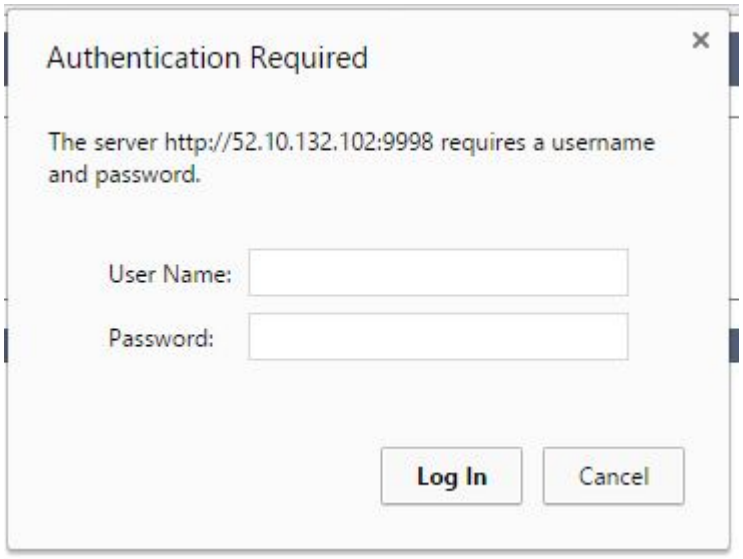
Správnost našeho nastavení můžeme ověřit zadáním na jakémkoliv počítači připojeném k internetu do internetového prohlížeče ip adresy našeho virtuálního stroje, portu 9998 a cesty /rest. Pro muj virtuální stroj je tedy adresa `http://52.10.132.102:9998/rest`. Pokud CashBob Server funguje a je dostupný z internetu nabídne nám prohlížeč dialog k přihlášení viz obrázek 4.5.



Obrázek 4.3: Odhalení hesla



Obrázek 4.4: Pravidlo pro CashBob ve Windows Firewall



Obrázek 4.5: Dialog přihlášení signalizující úspěšné připojení k RESTu CashBob Serveru

Kapitola 5
Xamarin

Xamarin je poměrně mladá firma vedená dvojicí technologických vizionářů Nata Friedmana a Miguela de Icazy. Firma Xamarin je založena na open source software projektu Mono [4]. Mono je open source implementace .NET Frameworku od firmy Microsoft pro Linux a Mac, které Microsoft nepodporuje. Xamarinu se podařilo Mono naportovat jak na operační systém Android tak i na iOS od společnosti Apple. Takto umožňují vývojářům programovat na obě platformy pomocí stejného jazyka (C#) a sdílet tak kód mezi platformami.

5.1 Kompilace

Na Android ani na iOS nemáme administrátorská práva, takže nemůžeme nainstalovat Mono do operačního systému. Xamarin toto obchází speciálními postupy:

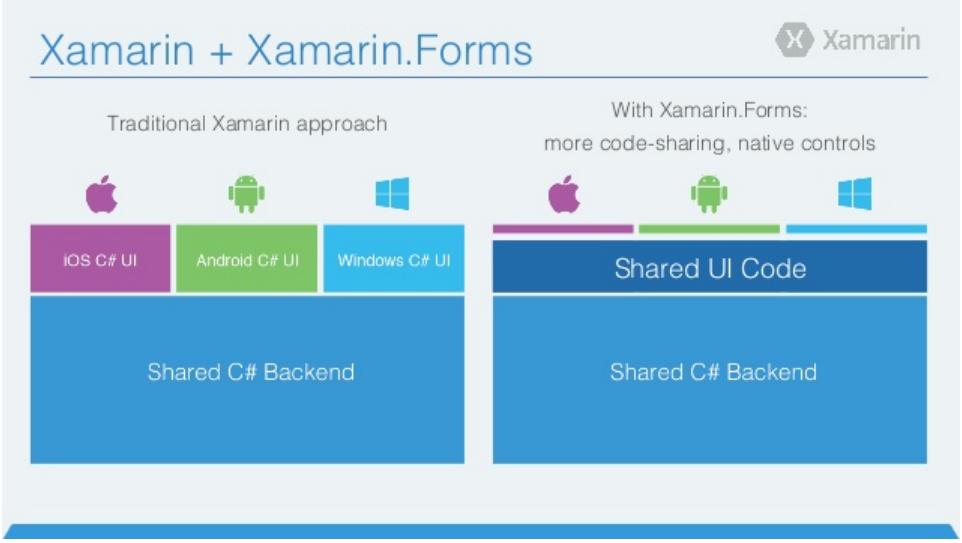
- iOS – C# je kompilován předem do ARM assembleru. Nepoužívané části .NET Frameworku jsou odstraněny při linkování pro zmenšení velikosti aplikace. Apple nepovoluje generování kódu za běhu programu, takže některé vlastnosti jazyka jako například *dynamic* nejsou dostupné.

- Android – C# je kompilován do jazyka IL (Intermediary Language) a sbalen s virtuálním strojem MonoVM, který za chodu programu překompiluje IL do strojového kódu přístroje. Tomuto procesu se říká JIT, neboli Just In Time kompilace. Nevyužité třídy frameworku jsou zase odstraněny při linkování, aplikace ve výsledku běží vedle virtuálního stroje Dalvik/Android Runtime a komunikuje s ním pomocí komunikační služby JNI kvůli využívání nativních objektů/funkcí.

5.2 Konkurence

Xamarin je jediné řešení pro tvorbu multiplatformních aplikací, které nevyužívá webových technologií (HTML, CSS, Javascript). Většina těchto řešení funguje jako nadstavba nad open-source řešením Apache Cordova [5]. Nejznámějším z nich je Adobe PhoneGap [6]. Takovéto aplikace mají oproti nativním mnoho nevýhod, hlavně co se týče uživatelského rozhraní. Webové aplikace jsou mnohdy pomalé a necitlivé. Uživatelé jsou mnohem radši, když vidí nativní komponenty, na které jsou zvyklí. Studie [7] [8] ukazují, že nativní aplikace jsou mnohem příjemnější na používání. Například Facebook nabízel pro mobilní zařízení hybridní aplikaci. Tím, že místo hybridní aplikace nabídnul uživatelům iOS nativní aplikaci se zvýšilo jejich hodnocení na App Store z 1.5 hvězd na 4 hvězdy za 3 týdny [9].

Nabízet mobilní aplikace je vhodné pouze pokud jsou uživatelé nuceni aplikaci používat, jako například Wikipedia, ale ne pokud se jí snažíme prodat. Hlavní výhodou multiplatformních aplikací je jejich cena. Xamarin umožňuje vytvořit nativní aplikace se sdíleným kódem dvěma různými způsoby, viz obrázek 5.1. Jedním přistupem je sdílet jen logiku aplikace a vytvořit uživatelské prostředí pomocí nativních nástrojů každé platformy, nebo využít framework Xamarin.Forms.



Obrázek 5.1: Sdílení kódu

5.3 Xamarin.Forms

Xamarin.Forms je UI framework, který funguje jako abstrakce jednotlivých prvků uživatelského prostředí a při běhu jsou tyto prvky nahrazeny na-

```
using Xamarin.Forms;

var profilePage = new ContentPage {
    Title = "Profile",
    Icon = "Profile.png",
    Content = new StackLayout {
        Spacing = 20, Padding = 50,
        VerticalOptions = LayoutOptions.Center,
        Children = {
            new Entry { Placeholder = "Username" },
            new Entry { Placeholder = "Password",
                        IsPassword = true },
            new Button {
                Text = "Login",
                TextColor = Color.White,
                BackgroundColor = Color.FromHex("77D065") }}}
};

var settingsPage = new ContentPage {
    Title = "Settings",
    Icon = "Settings.png",
    //(...)
};

var mainPage = new TabbedPage { Children = { profilePage,
                                             settingsPage } };
```

Obrázek 5.2: Ukázka Xamarin.Forms stránky zapsané pomocí kódu

tivními variantami pomocí dependency injection. Myšlenka tohoto frameworkem byla poskytnout .NET programátorům způsob jak popsat UI způsobem, na který už jsou zvyklí z technologie WPF [10] a zároveň jim umožnit sdílet i uživatelské prostředí. Jako ve většině UI frameworků programátor vytváří stromovou strukturu jednotlivých prvků. Tuto stromovou strukturu můžeme zapsat dvouma hlavními způsoby: pomocí normálního C# kódu vytvoříme instance objektů a přiřadíme jim pak reference na jejich větve. Nebo pomocí jazyka XAML (Extensible Application Markup Language), založeném na XML, který jej obohacuje o další pravidla jak přesně popsat objekty. Výhodou jazyka XAML je stromová struktura (viz 5.3), která je jasně viditelná, u větších stránek stručnější než C# a hlavně rozděluje kód stránky a strom do dvou oddělených souborů. Proto jsem zvolil ve své práci popis stránek pomocí jazyka XAML.

```
<?xml version="1.0" encoding="UTF-8"?>
<TabbedPage xmlns="http://xamarin.com/schemas/2014/forms"
            xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
            x:Class="MyApp.MainPage">
    <TabbedPage.Children>
        <ContentPage Title="Profile" Icon="Profile.png">
            <StackLayout Spacing="20" Padding="20"
                         VerticalOptions="Center">
                <Entry Placeholder="Username"
                    Text="{Binding Username}"/>
                <Entry Placeholder="Password"
                      Text="{Binding Password}"
                      IsPassword="true"/>
                <Button Text="Login" TextColor="White"
                        BackgroundColor="#77D065"
                        Command="{Binding LoginCommand}"/>
            </StackLayout>
        </ContentPage>
        <ContentPage Title="Settings" Icon="Settings.png">
            <!-- Settings -->
        </ContentPage>
    </TabbedPage.Children>
</TabbedPage>
```

Obrázek 5.3: Ukázka Xamarin.Forms stránky zapsané pomocí XAML

```
public class CalendarView : View
{
    //Vlastnosti
}

[assembly: ExportRenderer(typeof(CalendarView),
    typeof(CalendarViewRenderer))]


public class CalendarViewRenderer : NativeRenderer
{
        protected override void OnModelSet(VisualElement model)
        {
                CalendarView cw = (CalendarView)model;
                //Vytvoreni nativnich prvku na
                // zaklade vlastnosti CalendarView

                base.SetNativeControl(nativeControl);
        }
}
```

Obrázek 5.4: Ukázka anatomie Xamarin.Forms prvku

Každý prvek tohoto frameworku je sestaven z několika tříd, viz obrázek 5.4. První třída představuje model prvku, kterou sdílí všechny platformy a která definuje všechny vlastnosti a funkce tohoto prvku. Potom existuje třída dědící od třídy NativeRenderer pro každou platformu, která na základě vlastností modelu vytvoří stromovou strukturu nativních prvků uživatelského prostředí té konkrétní platformy. Tato třída se stará o to, jak bude ve výsledku prvek vypadat na dané platformě. Aby mohl framework tento renderer využít, tak je potřeba, aby byl zaregistrovaný s dependency injection systémem frameworku pomocí assembly: ExportRenderer atributu.

Takto lze samozřejmě framework i rozšířit o vlastní prvky.

Kapitola 6
Implementace

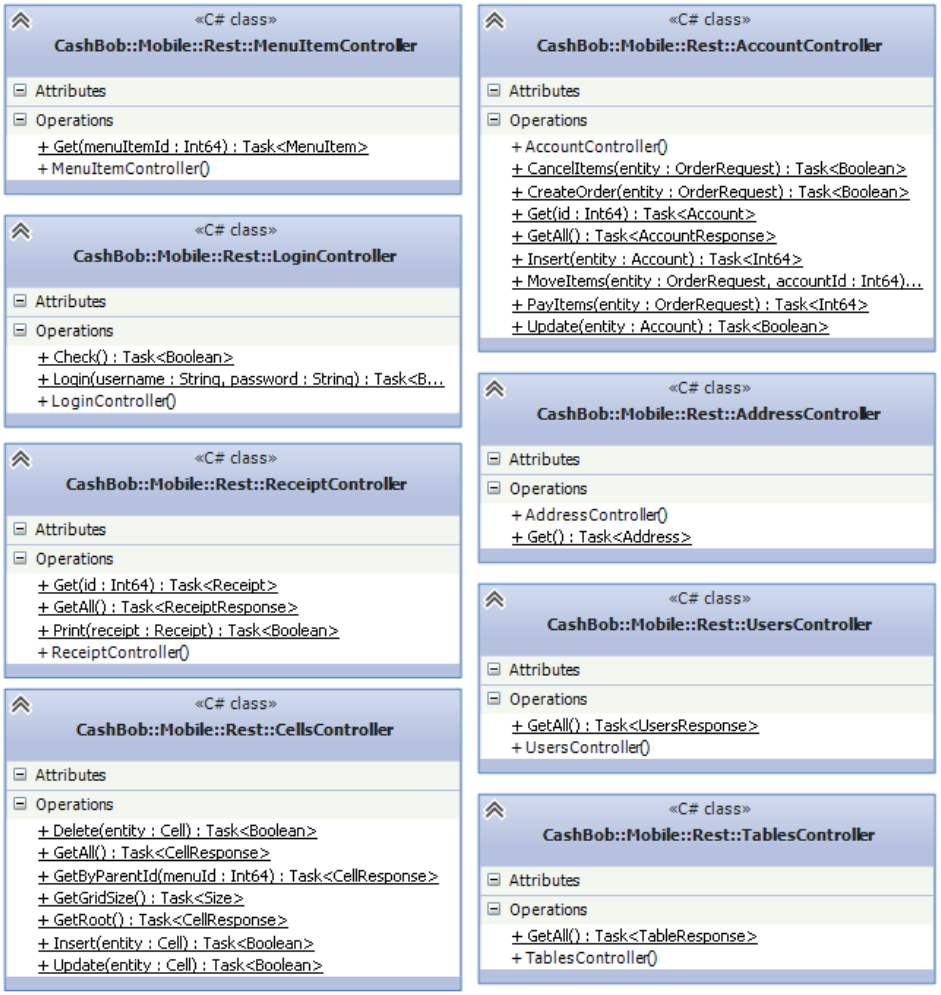
Řešení je rozděleno do tří hlavních projektů a několika pomocných projektů, jako testování a diagramy. Hlavními projekty jsou Android, iOS a sdílená knihovna, kde je většina kódu, hlavně logika aplikace a uživatelské prostředí. Projekty pro specifické platformy obsahují pouze kód specifický pro tuto platformu, který nemůže být sdílen.

Implementaci aplikace jsem původně zamýšlel realizovat podle návrhového vzoru MVVM (Model - View - ViewModel), který se k implementaci takovýchto aplikací často používá. Umožňuje totiž oddělení kódu uživatelského rozhraní od logiky aplikace, což je velmi žádané kvůli implementaci UI vrstvy pro každou platformu zvlášť pomocí nativních prvků. Potom jsem zjistil, že některé prvky Xamarin.Forms (například Picker) nepodporují binding, který je k správné implementaci vzoru MVVM nutný a infrastrukturu pro sdílení UI mezi platformami již zajišťují Xamarin.Forms. Navíc při psaní stránek pomocí jazyka XAML již kód od UI oddělený je, takže na přehlednosti kódu jsem také moc nezískal. Stránky implementované pomocí MVVM jsou: Nastavení, Přihlášení, a částečně Přehled účtů. Ostatní stránky jsem implementoval s logikou v "code-behind", stejně jako u desktopové aplikace.

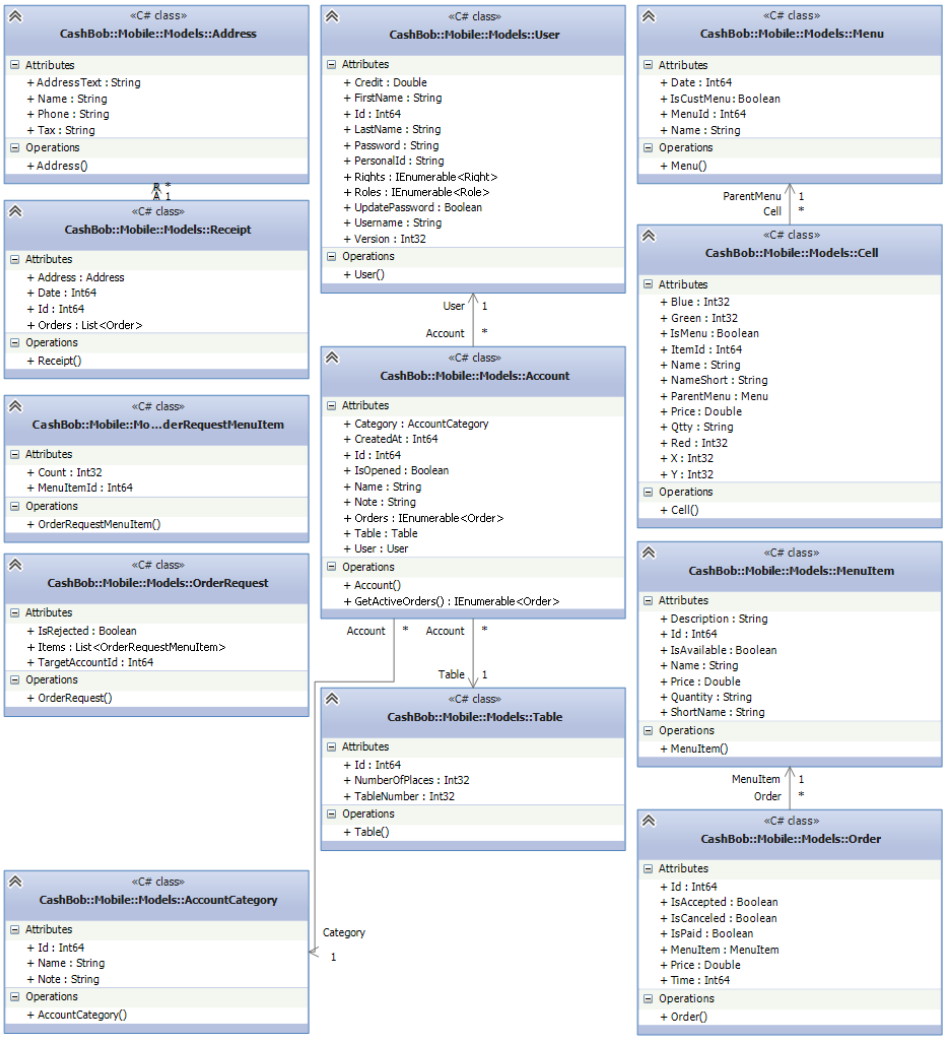
6.1 Dependency Injection

Výběr dependency injection knihovny na mobilních telefonech je omezený, protože kvůli iOS, který zakazuje modifikaci kódu aplikace za běhu, není podporována reflexe, kterou většina dependency injection frameworků využívá. Toto jim umožňuje injectovat pomocí atributů a dalších usnadnění pro programátory. Jako dependency injection kontejner jsem zvolil Ninject [11], protože je z konkurence pro telefony nejvyspělejší a poskytuje nejvíce funkcionality.

Ninject používá k získávání částí k injectování tzv. moduly, což jsou třídy derivující od NinjectModule, které definují seznam částí a podrobnosti



Obrázek 6.1: Kontrollery



Obrázek 6.2: Modely

o jejich cyklu života. Pro tuto aplikaci jsem vytvořil tři moduly. Jeden pro iOS, druhý pro Android a třetí pro sdílený kód. Části získatelné z kontejneru jsou například lokalizační systém, připojení k SQLite [12] databázi nebo http klient.

6.2 Lokalizace

Xamarin neposkytuje sdílený lokalizační systém pro všechny operační systémy. Namísto toho doporučuje využívat lokalizační funkce každého operačního systému zvlášť. Překlady jsou pak na několika místech a složitě se udržují. Navíc jsme chtěli využít už existující překlady z desktopové aplikace CashBob, protože většina textů bude indentických a i údržba překladových párů bude jednodušší. Napsal jsem proto třídu *LocalizationCatalog*, která na základě aktuálního jazyku systému, který dostane jako parametr konstruktoru sestaví slovník překladových párů. Překladové páry jsou získány rozborem překladových souborů přibalených v aplikaci. Tyto soubory jsou ve formátu ukázaném na obrázku 6.3 a jsou pojmenované ve tvaru *restaurace_fel_(jméno)_bundle_(jazyk).properties*.

```
LoadingLabelText=Loading...
confirm_logout_text=Do you really want to log out?
confirm_logout_title=Logging out
```

Obrázek 6.3: Ukázka formátu lokalizace aplikace CashBob

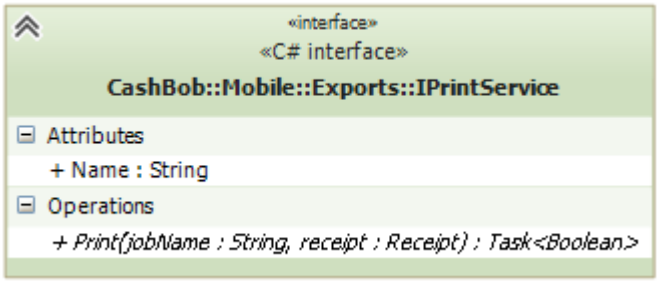
Tuto třídu jde využít přímo požádáním dependency injection kontejneru o instanci třídy *LocalizationCatalog* nebo přes XAML rozšíření *TranslateExtension* jak ukazuje obrázek 6.4.

```
<ContentPage
 xmlns="http://xamarin.com/schemas/2014/forms"
 xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
 xmlns:i18N="clr-namespace:CashBob.Mobile.Common.Localization;
             assembly=CashBob.Mobile"
 x:Class="CashBob.Mobile.Views.ConfirmPaymentPage"
 Title="{i18N:Translate items}">
```

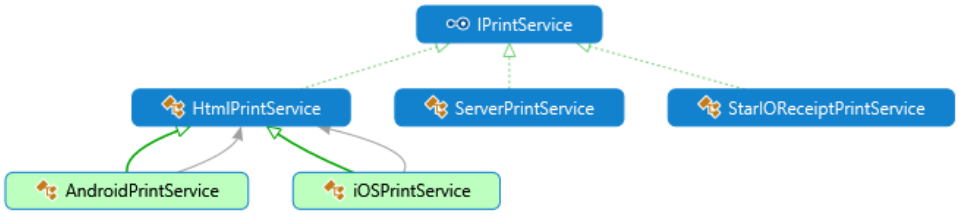
Obrázek 6.4: Ukázka použití třídy TranslateExtension

Kapitola 7
Tisk

Jednou z požadovaných funkcí byl tisk účtenek přímo z aplikace. Účtenky mohou být tištěny několika různými způsoby takže bylo nutno navrhnout modulární architekturu. Mimojiné moduly mohou tisku využívat kód specifický pro určitou platformu, takže je nutné oddělit moduly od uživatelského prostředí. Každý modul sdílí rozhraní *IPrintService* a stará se o něj dependency injection kontejner.

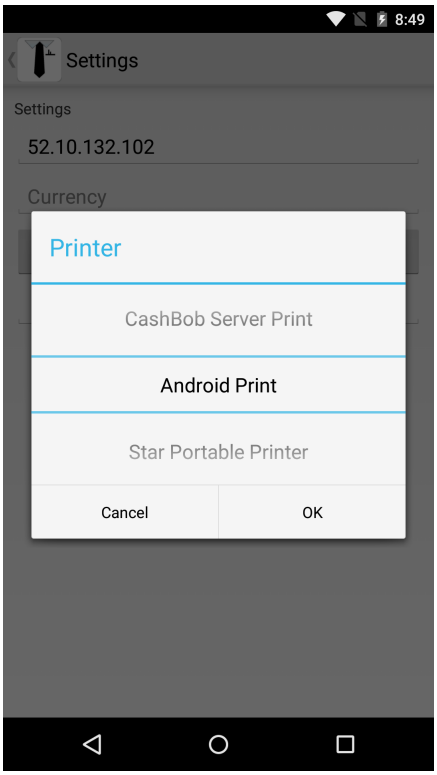


Obrázek 7.1: Rozhraní IPrintService

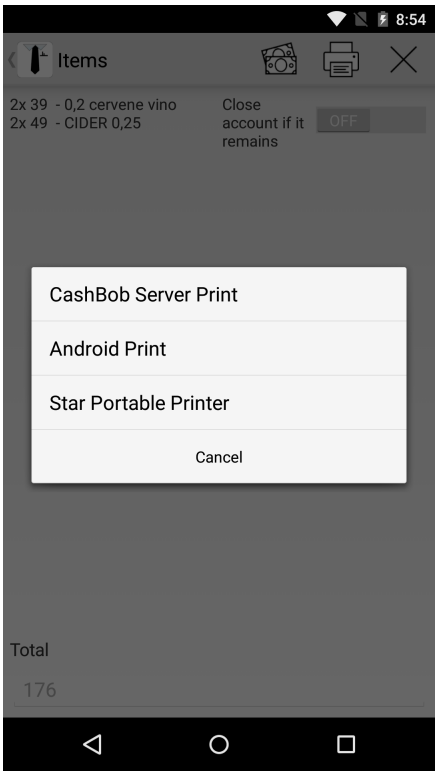


Obrázek 7.2: Graf dědičnosti modulů tisku

V kódu pro každou platformu zvlášť se s dependency injection kontejnerem zaregistrují jenom moduly tisku, které mohou na této platformě tisknout. Jakmile uživatel požádá o tisk, stránka požádá dependency injection kontejner o všechny zaregistrované moduly, vybere z nich ten, který byl zvolen v nastavení (obrázek 7.3) jako výchozí a požádá ho o vykonání tisku.



Obrázek 7.3: Nastavení



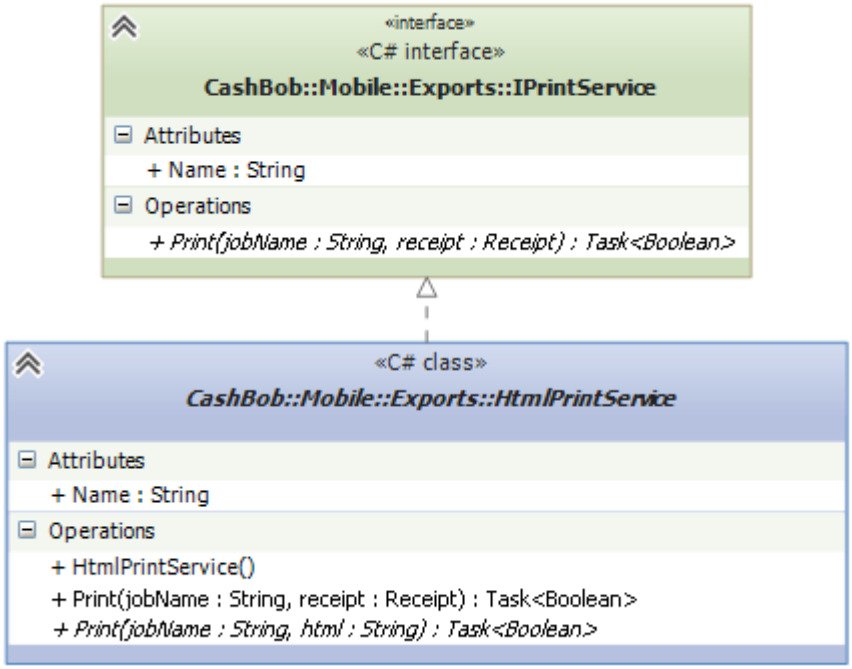
Obrázek 7.4: Dialog výběru modulu

Pokud nebyl v nastavení výchozí modul zvolen zobrazí se dialog pro výběr modulu viz obrázek 7.4.

7.1 Tisk pomocí operačního systému

iOS i Android mají zabudovaný systém pro tisk dokumentů. Android od verze v4.4 obsahuje API pro bezdrátové tisk na tiskárny s WiFi. Navíc pomocí technologie Google Cloud Print [13] mohou majitelé zařízení s operačním systémem Android tisknout na jakoukoliv tiskárnu připojenou k počítači s internetovým připojením. iOS má od verze 4.2.1 pro tisk technologii AirPrint, která funguje pouze se specifickými tiskárnami s podporou technologie AirPrint a WiFi. API pro tisk Androidu i iOS podporují velmi jednoduše tištění webových stránek. Proto jsem se rozhodl generovat vizuální podoby účtenek ve formátu HTML. Tento formát se pro tento úkol hodí kvůli své nezávislosti na platformách, snadnému vytvoření a jednoduchému tisknutí díky své podpoře přes všechny platformy. Aby byla šablona co nejbližší šabloně používané na CashBobu na desktopu, vyexportoval jsem stávající šablonu vytvořenou pomocí Jasper Reports [14] do HTML. K tomu bylo nutno aktualizovat v CashBobu balíček JasperReports na nejnovější verzi

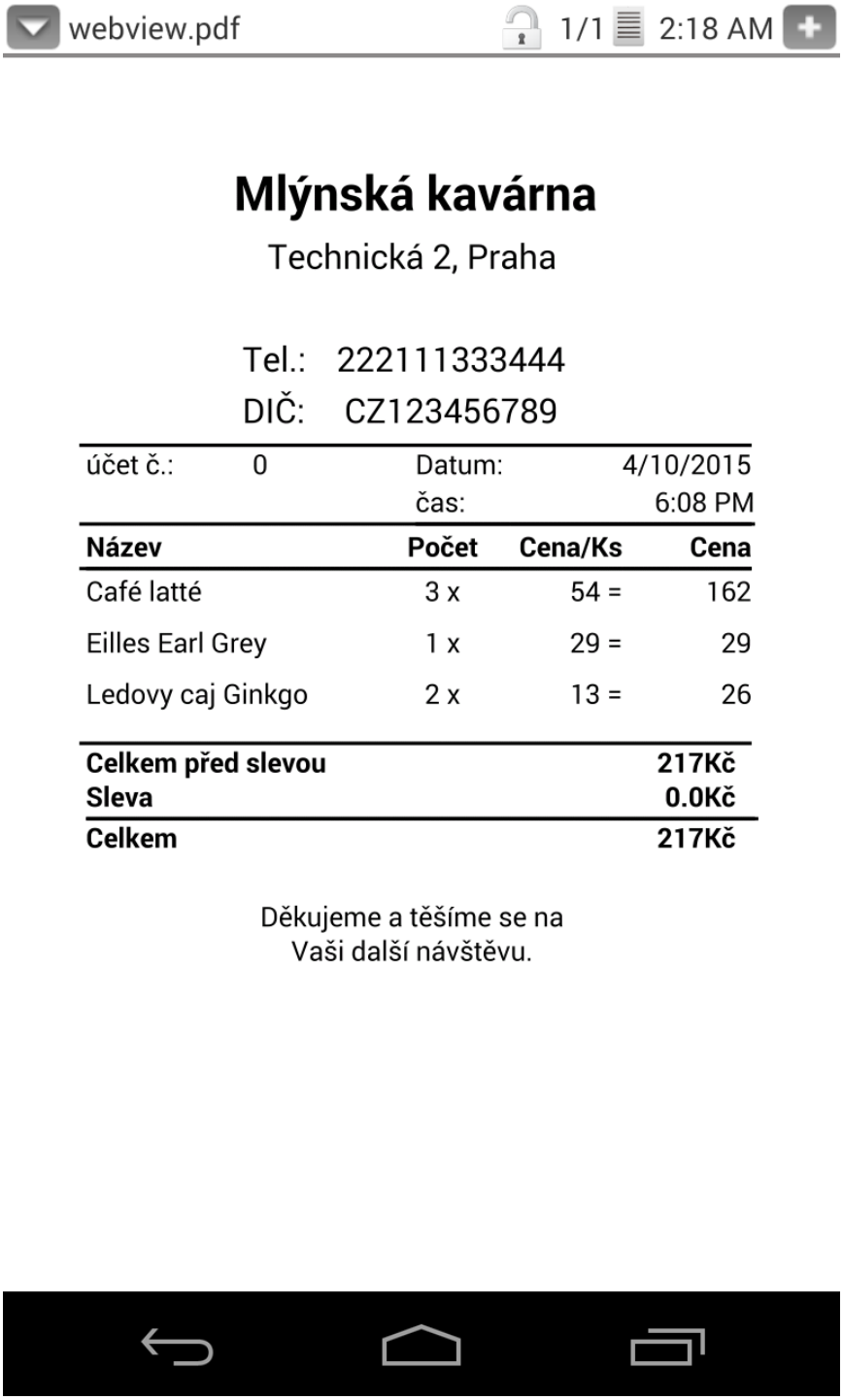
a dočasně do kódu přidat příkaz pro export šablony do formátu HTML. Xamarin pro tvorbu hybridních aplikací implementoval tzv. Razor engine, jazyk vytvořený společností Microsoft jakou součást jejich webového frameworku ASP.NET pro generování dynamického HTML. Výslednou šablonu jsem tedy vytvořil tak, že jsem doplnil HTML vygenerované JasperReports o značky jazyka Razor a C# kód, který doplní do předepsaného HTML informace o objednávkách, času, ceně, atd. O využití šablony pro generaci HTML se stará abstraktní třída *HtmlPrintService* (viz obrázek 7.5), kterou rozšiřují třídy *AndroidPrintService* a *iOSPrintService* (viz obrázek 7.2). Ty jenom předají HTML kód nativnímu API pro tištění konkrétního operačního systému.



Obrázek 7.5: Třída HtmlPrintService

7.2 Tisk pomocí CashBob Serveru

Tisk pomocí CashBob Serveru zahrnoval hlavně práci na serverové aplikaci. Přidal jsem proto RESTovou API pro tisk účtenek (`/rest/receipt/print`), která po přijetí POST requestu s daty tuto účtenku vytiskne na výchozí tiskárnu. Data účtenky jsou objekt typu ReceiptDTO z CashBobServeru serializované do JSONu, jako například na obrázku 7.7. CashBob Server nikdy pro tisk připraven nebyl, takže bylo nutno s ním přibalit i adresář se šablonami pro JasperReports.



Obrázek 7.6: Účtenka podle HTML šablony

```
{   "id" : 90,
    "date" : 1365764037390,
    "orders": [
        {   "id" : 942006,
            "ispaid" : true,
            "time" : 1365764031134,
            "isaccepted" : true,
            "iscanceled" : false,
            "price" : 39.0,
            "menuItem":
            {   "menuitemid" : 452,
                "name" : "0,2 cervene vino",
                "price" : 39.0,
                "quantity" : "1",
                "isavailable" : true
            }
        }]
}
```

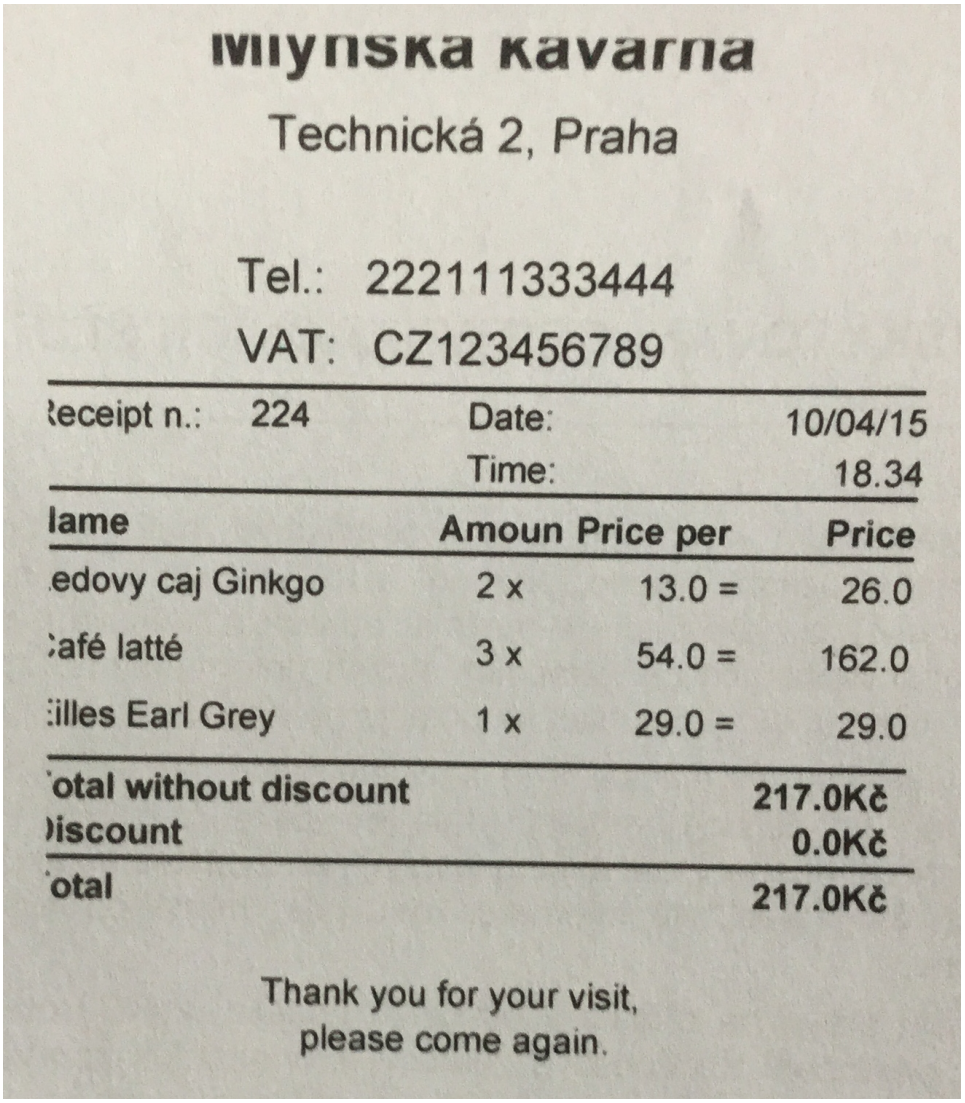
Obrázek 7.7: Ukázka ReceiptDTO ve formátu JSON

Na klientské části stačilo vytvořit nový modul, který pošle HTTP POST request na server s daty právě vytvořené účtenky. Tento modul představuje třída *ServerPrintService*.

7.3 Tisk s přenosnou tiskárnou

U implementace tisku s mobilní tiskárnou firmy Star Micronics modelu SM-T300 [15] jsem narazil na zdaleka nejvíce problémů. Tato přenosná tiskárna komunikuje s ostatními zařízeními pomocí technologie Bluetooth, nebo sériového portu. Největším problémem se ukázalo, že žádné moje iOS zařízení (iPhone 5S, iPad Air 2) neobjevilo přes BlueTooth tiskárnu, přestože Star Micronics propagují v oficiálních materiálech kompatibilitu s iOS. U zařízení s Android vše proběhlo podle plánu a tiskárnu se podařilo úspěšně spárovat. Z tohoto důvodu podporuje přenosnou tiskárnu pouze verze aplikace pro Android.

Dalším krokem bylo StarIO SDK, které zprostředkuje připojení a komunikaci s tiskárnou. Star Micronics neposkytují toto SDK pro Xamarin Android, pouze Java verzi. Xamarin sice umožňuje vytvoření vazebných knihoven pro využívání knihoven psaných v Java s C#, ale vygenerovaná vazebná knihovna nebyla korektní. Při snaze vytvořit objekt z této knihovny konstruktor vyhodil výjimku. Nakonec jsem problém se SDK vyřešil tak, že jsem knihovnu pro javu dekompiloval pomocí IntelliJ IDEA IDE [16] a



Obrázek 7.8: Účtenka vytištěná vzdáleně přes CashBob Server

Obrázek 7.9: Párování zařízení s přenosnou tiskárnou

přepsal do C#.

Obrázek 7.10: Účtenka z přenosné tiskárny

```
private void GetPairedWoosimPrinter() throws StarIOPortException {
        BluetoothAdapter var1 = BluetoothAdapter.getDefaultAdapter();
        Set var2 = var1.getBondedDevices();
        BluetoothDevice[] var3 = (BluetoothDevice[])var2
            .toArray(new BluetoothDevice[var2.toArray().length]);

        for(int var4 = 0; var4 < var3.length; ++var4) {
            String var5 = var3[var4].getAddress();
            if(ValidWoosimMacAddress(var5)) {
                this.portName = "BT:" + var5;
                return;
            }
        }

        throw new StarIOPortException("Cannot find bluetooth printer");
}
```

Obrázek 7.11: Ukázka java kódu dekompilovaného s oficiálního StarIO SDK pro Android

```
private void GetPairedWoosimPrinter()
{
    foreach (BluetoothDevice t in BluetoothAdapter
                .DefaultAdapter
                .BondedDevices) {
        if (ValidWoosimMacAddress(t.Address))
        {
            portName = "BT:" + t.Address;
            return;
        }
    }
    throw new StarIOPortException("Cannot find bluetooth printer");
}
```

Obrázek 7.12: Ukázka dekompilovaného kódu ručně přepsaného do C#

Kapitola 8
Testování

Tato kapitola se zabývá testováním obou mobilních aplikací pomocí unit testů, automatizovaných UI testů a manuálního testování uživateli. Testy byly prováděny na zařízeních iPhone 5S, iPad Air 2, Galaxy S5 a emulátorech obou platforem.

8.1 Unit Testing

Unit testování mobilních aplikací je složitější, než testování aplikací pro osobní počítače, protože kód který testujeme musí běžet na mobilním zařízení, což je jiný stroj, než na kterém aplikaci vyvíjíme. Proto nelze použít testovací nástroje zabudované do IDE. K unit testování jsem vybral framework xUnit [17], který obsahuje jako jeden z mála podporu pro Xamarin iOS i Xamarin Android. Všechny unit testy jsou v projektu *CashBob.Mobile.UnitTests*. Testy, které vyžadují spolupráci se serverem, kterých je většina, dědí třídu *ApiTestBase*. Tato třída zaregistruje s dependency injection systémem nastavení připojení k serveru v cloudu, který byl připraven pro tyto účely v počáteční části této práce. Pro spuštění testů je nutno spustit projekt *CashBob.Mobile.UnitTests*, který je aplikace pro Android a v té zvolit volbu "Run Everything". Výsledky testů se zobrazí v této aplikaci.

Unit testy jsem využil hlavně při vývoji složitěji dostupných funkcí aplikace, abych nemusel pokaždé změně funkci testovat manuálně.

8.2 UI Testing

K UI testování aplikace jsem framework Xamarin Test Cloud [18]. Xamarin Test Cloud je služba, kterou nabízí firma Xamarin k testování mobilních aplikací na stovkách reálných zařízení v pobočce Xamarinu a odeslání zpět výsledků testů. Tato služba je však navržena pro velké korporace a základní varianta stojí $1,000/měsíc. Tyto nástroje naštěstí ale umožňují testování

jednotlivě na svém zařízení aby mohli vývojáři své testy před odesláním do test cloudu ověřit na svém zařízení.

Programátor ve svých testech nakonfiguruje proměnnou typu *IApp* s apk nebo bundlem aplikace, kterou chce testovat a tato proměnná nabízí fluentní api k ovládání aplikace. Obsahuje funkce jako například najít UI prvek podle parametrů, stisknutí na koordinátech nebo určitého prvku, pořizování screenshotů, provádění gest, ovládání hlasitosti, psaní textu nebo tahy prstem. Zajímavé je, že tyto nástroje nepracují velmi dobře s Xamarin.Forms což je produkt stejné firmy. Problém je v tom, že ovládání prvků se většinou odehrává kolem hledání UI prvku pomocí jeho ID a vykonání na něm nějaké akce. Ale Xamarin.Forms ID přiřazené Xamarin.Forms prvku nepropisuje do nativní varianty takže ve výsledku ID prvku chybí. Xamarin sice nabízí workaround, který jsem použil, ale mohu jen konstatovat, že nefunguje ve všech případech. Hlavně v případech, kdy je Xamarin.Forms prvek přeložen do několika nativních prvků.

Výhodou Xamarin Test Cloud testů je, že mohou být integrovány do normálních unit testovacích frameworků, jako například MSTest nebo NUnit, a také Continuous Integration serverů jako například TFS, TeamCity nebo Jenkins.

Testy jsou k nalezení v projektu *CashBob.Mobile.Tests*.

8.3 Manuální testování

Aplikace byla testována extenzivně manuálně hlavně formou explorativního testování [19], kdy byl princip aplikace uživateli vysvětlen, a ten pak procházel aplikací a zkoušel všechny funkce aplikace. Narazil jsem při tomto testování s lidmi na několik zajímavých chyb, které by nemohly být odhaleny UI testováním. Například jsem narazil na problém kdy netrpělivý tester klikal na tlačítko, dokud aplikace nespadla, protože nahrávala data z internetu před zobrazením nové stránky. Protože je celá aplikace async a UI není blokováno, tlačítko mohl zmáčknout znova a funkce nebyla připravena na opakované volání v jednom momentu. Další zajímavý problém, který byl odhalen manuálním testováním byl s dlouhými popisky tlačítek v panelech nástrojů. Na menších obrazovkách se nevešly všechny položky menu a uživatel musel otočit obrazovku aby mohl například zaplatit účet. Tento problém jsem vyřešil nahrazením dlouhých textových popisků ikonkami. Ikonky pro Android jsem některé vzal z desktopové verze, ale ostatní ikonky jsem musel vytvořit sám. Ikonky pro iOS jsem musel všechny vytvořit zvlášť protože iOS zobrazuje ikonky jednobarevně a dodaný obrázek používá jenom jako masku.

Všechny nalezené závady i z ostatních formem testování byly odstraněny.

	Sdílený	Android	iOS	StarIO SDK
Index udržovatelnosti	86	77	85	81
Cyklomatická složitost	559	203	26	807
Hloubka dědičnosti	7	7	4	3
Vazby tříd	215	81	45	66
Řádků kódu	875	356	36	2001

Tabulka 8.1: Statistiky kódu

8.4 Statická analýza kódu

Dále jsem provedl statickou analýzu kódu zabudovanými analytickými nástroji Visual Studia. Statická analýza pomáhá nalézt slabá místa v kódu a eliminovat potenciální chyby. Chyby odhalené v hlavním projektu jsem odstranil, chyby v dekompilovaném kódu (StarIO.Android) jsem nechal z časových důvodů. Výsledné statistiky jsou vidět na tabulce 8.1

Kapitola 9
Závěr

9.1 Zhodnocení

Tato práce měla za cíl položit základ nové moderní generaci komplexního řešení pro restaurace a kavárny CashBob. V rámci této práce měla aplikace obsahovat jenom základní funkcionalitu pokladního modulu, ale byla od začátku zamýšlená tak, aby do budoucna mohla nahradit desktopového klienta. Udržovat a rozšiřovat tuto aplikaci bude podstatně jednodušší než každou aplikaci zvlášť.

Výsledkem bakalářské práce jsou funkční a otestované mobilní aplikace s uživatelsky přívětivým designem. Sdílejí jádro, které obsahuje uživatelské prostředí, logiku aplikace a komunikuje se serverem prostřednictvím protokolu REST. Projekt jsem udržoval na svém TFS serveru a vyplynulo z něj 350+ commitů a 3500+ řádek kódu.

Všechny požadavky byly splněny, včetně zprovoznění CashBob Serveru v cloudu pro snažší uživatelské testování. Jediné, co se nepodařilo zcela realizovat, bylo tištění s přenosnou tiskárnou Star Micronics na platformě iOS, kvůli nekompabilitě tiskárny s žádným z mých iOS zařízení (iPhone 5S a iPad Air 2). Pokud vyjde nový model tiskárny, který bude podporovat nová iOS zařízení, neměl by být problém tento modul doplnit. Funkcionalita mobilní aplikace byla otestována pomocí unit testů, UI testů a hlavně uživatelským testováním.

9.2 Zkušenosti s Xamarinem

Moje zkušenosti s nástroji Xamarin jsou poněkud smíšené. Na začátek musím říct, že považuji nástroje Xamarin za budoucnost vývoje nejen mobilních aplikací, které mají nesmírný potenciál. Byť jsem v tomto projektu nedosáhl neuvěřitelně vysokých procent sdíleného kódu, myslím si, že toto je ojedinělý případ kvůli tiskárně Star Micronics. V iOS aplikaci, která podporu pro tuto tiskárnu nemá sdílím 96% kódu, zato v Android aplikaci sdílím 70% když nepočítám SDK pro spolupráci s tiskárnou Star Micronics. Co je ale hlavní,

tak sdílím všechnu logiku aplikace a uživatelské prostředí, což jsou nejčastěji upravované části. Navíc jsem mohl použít mnoho pokročilých funkcí jazyka C#, které bych u Objective-C/Swift nebo Javy použít nemohl (například async). V tomto ohledu jsem s Xamarinem velmi spokojen a plánuji využívat Xamarin v dalších projektech i v budoucnu.

Projekt jsem vyvíjel v Xamarin for Visual Studio v 3.9 a Visual Studio 2013. Jako build server pro iOS aplikace jsem použil virtuální PC s OS X Maverics. Nakonec jsem si kvůli Xamarin.Forms byl nucen ještě nainstalovat doplňek do Visual Studia ReSharper [20] od firmy JetBrains. Bez něj totiž nástroje Xamarin neobsahují doplňování a kontrolu syntaxe XAML stránek pro Xamarin.Forms a psát stránky v poznámkovém bloku by bylo velmi časově náročné. A to je jenom začátek velmi dlouhé kapitoly. Nástroje Xamarin pro člověka zvyklého na neprůstřelnou kvalitu .NET frameworku a Visual Studia byly návratem do reality. Chyby v nástrojích jsou na denním pořádku. Například debugger nefunguje pořádně do dnes a na úroveň .NET frameworku se asi nikdy nedostane. Uvědomuji si, že Xamarin má mnohem těžší úkol, ale jejich nástroje také nejsou levné. Často se stávalo, že mi týdny nástroje nefungovali vůbec a nemohl jsem tak pracovat, nebo jsem narazil na chybu a čekal pár dní než jí opraví. Co jsem koukal po internetu na názory ostatních vývojářů na toto téma, většinou byli ještě méně schovívaví.

Během roku, který jsem pracoval na této práci jsem byl svědkem výrazného zlepšení kvality jejich nástrojů. Například teď nabízí emulátor pro Android Xamarin Android Player [21], který musí každý Android vývojář nutně mít vzhledem k tomu jak bídné emulátory od Google jsou. Také je volitelnou součástí nástrojů od firmy Xamarin Profiler a Xamarin Insights, nástroje pro monitorování rychlosti a odchytávání chyb v provozu. Vytvořit v Xamarinu složitou a stabilní aplikaci se rozhodně dá a dokonce bych řekl, že to tam teprve začíná Xamarin ukazovat ze sebe to nejlepší. Mojí největší chybou, byla vysoká očekávání produktivity a jednoduchosti zacházení s nástroji, po mé dlouholeté zkušenosti s .NET frameworkem.

9.3 Návrh další práce

9.3.1 Slevy

V době tvorby této práce nejsou v aplikaci funkční slevy. Tato funkcionalita byla již v minulosti předmětem minimálně jedné práce, ale nebyla dotažena do konce. Díky funkci vlastní položky menu není tato vlastnost úplně nutná, ale myslím si, že tato funkce rozhodně patří do všestranného softwarového řešení pro restaurace.

9.3.2 Kompatibilita desktopových aplikací s Javou 8

Z nějakého záhadného důvodu při spuštění CashBob Serveru pod Javou 8 RESTová API neodpovídá na dotazy. Bylo by vhodné tuto chybu opravit, pravěpodobně použitím novější verze komponent, aby mohly být využívány výhody nové verze jazyka.

9.3.3 CashBob Server jako webová aplikace

Vzhledem k tomu, že na serveru není nutno provádět moc uživatelských interakcí, mohla by serverová aplikace implementovaná čistě jako webová aplikace. Na druhou stranu by se lehce zkomplikovala instalace samostatných aplikací o instalaci a jedná se tedy o kontroverzní rozhodnutí, ale myslím si, že za uvážení rozhodně stojí. Toto by značně pomohlo k hostování aplikace v cloudu, hlavně v automatizaci, škálování, a ceně hostování. Serverová aplikace by pak mohla být nabízena jako BaaS a snadno tak i zpoplatněna.

9.3.4 Rozšířit tuto aplikaci o klienta pro Windows

Na konferenci dotnetConf 2015 představila firma Xamarin beta verzi frameworku Xamarin.Forms s podporou pro Windows "Modern UI". Díky tomuto rozšíření a architektuře aplikace by mělo být celkem snadné rozšířit aplikaci i o verzi pro Windows Phone a Windows. Nejvíce práce by bylo implementovat moduly pro tisk na těchto platformách.

9.3.5 Rozšířit multiplatformního klienta o pokročilou funkcionalitu desktopového klienta

V tuto chvíli může tato aplikace fungovat pouze jako doplňek desktopového klienta, jelikož obsahuje pouze základní funkcionalitu. Aby mohla tato aplikace nahradit desktopového klienta, měla by obsahovat veškerou jeho funkcionalitu. I jako doplňková aplikace je vždy lepší mít více možností.

9.3.6 Export a import dat

Pro začátek by asi stačilo aby aplikace měla uživatelsky dostupnou volbu pro kopírování databáze, protože podniky si nemohou dovolit ztratit takováto data. Jinak si myslím, že bylo vhodné umožnit exportovat a importovat jenom určité části databáze (například pouze menu). Platí pořekladlo: „data se dělí na zálohovaná a ještě neztracená“. Tato funkcionalita je užitečná i pro vývojáře aplikace.

9.3.7 Usability testing

Aplikace je uzpůsobena pro snadné používání, ale nikdy nebyla testována s reálnými uživateli. Usability testing může odhalit nedostatky, které by uživatele na aplikaci zvyklé nikdy nenapadly.

Bibliografie

[1] Stephen Hall for 9to5google. *Android Lollipop still straggling, adoption barely passes 5% in March*. URL: http://9to5google.com/2015/04/08/android-lollipop-adoption-5-march/ (cit. 03. 05. 2015).

[2] Joe Rossignol for MacRumors. *iOS 8 Adoption Reaches 81% Following Apple Watch Launch*. URL: http://www.macrumors.com/2015/04/29/ios-8-adoption-81-apple-watch/ (cit. 04. 05. 2015).

[3] Amazon. *Setting Up with Amazon EC2*. URL: http://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/get-set-up-for-amazon-ec2.html (cit. 23. 04. 2015).

[4] Mono Project. *Mono Project*. URL: http://www.mono-project.com/ (cit. 23. 04. 2015).

[5] Apache. *Apache Cordova*. URL: https://cordova.apache.org/ (cit. 23. 04. 2015).

[6] Adobe. *PhoneGap*. URL: http://phonegap.com/ (cit. 23. 04. 2015).

[7] Kinvey. *Native vs Web vs Hybrid: How to Select the Right Platform for Your Enterprise's Apps*. URL: http://www.kinvey.com/native-web-hybrid-strategy-eBook (cit. 23. 04. 2015).

[8] Inc. Forrester Research. *Improving Enterprise Mobility Meeting Next-Generation Demands For Development, Delivery, And Engagement*. URL: https://s3.amazonaws.com/corpassets.moovweb.com/wp/Forrester-Improving-Enterprise-Mobility-2014.pdf (cit. 23. 04. 2015).

[9] Josh Constine. *Facebook Speeds Up Android App By Ditching HTML5 And Rebuilding It Natively Just Like The iOS Version*. URL: http://techcrunch.com/2012/12/13/facebook-android-faster/ (cit. 23. 04. 2015).

[10] Microsoft. *WPF (Windows Presentation Foundation)*. URL: https://msdn.microsoft.com/en-us/library/aa970268(v=vs.110).aspx (cit. 23. 04. 2015).

[11] Ninject. *Ninject - Open source dependency injector for .NET*. URL: http://www.ninject.org/ (cit. 23. 04. 2015).

[12] Ninject. *Ninject - Open source dependency injector for .NET*. URL: http://www.sqlite.org/ (cit. 23. 04. 2015).

[13] Google. *Google Cloud Print*. URL: http://www.google.com/cloudprint/learn/ (cit. 23. 04. 2015).

[14] Jaspersoft. *JasperReports Library*. URL: https://community.jaspersoft.com/project/jasperreports-library (cit. 23. 04. 2015).

[15] Inc Star Micronics. *Star SM-T300 Mobile Printer*. URL: http://www.starmicronics.com/printer/portable_printers/sm-t300 (cit. 23. 04. 2015).

[16] JetBrains s.r.o. *IntelliJ Idea*. URL: https://www.jetbrains.com/idea/ (cit. 23. 04. 2015).

[17] Brad Wilson. *xUnit.net, a free, open source, community-focused unit testing tool for the .NET Framework*. URL: https://github.com/xunit/xunit (cit. 23. 04. 2015).

[18] Xamarin inc. *Mobile App Testing On Hundreds Of Devices - Xamarin Test Cloud*. URL: http://xamarin.com/test-cloud (cit. 26. 04. 2015).

[19] James Bach. *What is Exploratory Testing?* URL: http://www.satisfice.com/articles/what_is_et.shtml (cit. 20. 05. 2015).

[20] JetBrains s.r.o. *ReSharper Productivity Tool for Visual Studio*. URL: https://www.jetbrains.com/resharper/ (cit. 28. 04. 2015).

[21] Xamarin inc. *Simulate Android apps with the Xamarin Android Player*. URL: https://xamarin.com/android-player (cit. 30. 04. 2015).

Příloha A
Seznam použitých zkratek

.NET Framework vyvíjený firmou Microsoft

HTTP HyperText Transfer Protocol

GET Dotazovací metoda protokolu HTTP

POST Metoda protokolu HTTP

HTML HyperText Markup Language jazyk webových stránek

MVVM Softwarová architektura Model-View-View Model

REST Representational State Transfer, architektura API založené na protokolu HTTP

XF Xamarin.Forms, UI framework vyvíjený firmou Xamarin

UI User Interface, uživatelské rozhraní

UX User Experience, dojem uživatele ze zacházení s aplikací

API Application Programming Interface, sbírka funkcí a tříd, kterou mohou programátoři využívat

JSON JavaScript Object Notation, způsob zápisu objektů využívaný jazykem Javascript

SDK Software Developer Kit, soubor nástrojů pro vývoj software

BaaS Backend-as-a-Service, nabízení webových služeb pro aplikaci jako produkt

XML eXtensible Markup Language obecný značkovací jazyk, který byl vyvinut a standardizován konsorciem W3C

XAML eXtensible Application Markup Language, jazyk vyvinutý firmou Microsoft pro popis a initializaci struktury objektů

VPS Virtual Private Server, je server běžící na virtualizovaném hardware provozovaný hostingovou službou

IL intermediate language, je mezijazyk využívaný kompilátorem před generováním cílového strojového kódu

JNI Java Native Interface je rozhraní umožňující propojit kód běžící na virtuálním stroji Javy s nativními programy a knihovnami napsanými v jiných jazycích

JIT speciální metoda překladu do strojového kódu až za běhu programu

CSS Cascading Style Sheets jsou jazyk pro popis způsobu zobrazení elementů na stránkách napsaných v jazycích HTML, XHTML nebo XML

WPF Windows Presentation Foundation je UI framework součástí .NET

SQL Structured Query Language, standardizovaný strukturovaný dotazovací jazyk, který je používán pro práci s daty v relačních databázích

IDE Integrated Development Environment, software pro vývoj software.

ID Identifikátor

IP Internet Protocol, základním protokol pracující na síťové vrstvě používaný v počítačových sítích a Internetu

TFS Team Foundation Server, centralizovaný systém správy verzí

Wi-Fi Wireless Fidelity - označení pro několik standardů IEEE 802.11 popisujících bezdrátovou komunikaci v počítačových sítích

BlueTooth je proprietární otevřený standard pro bezdrátovou komunikaci propojující dvě a více elektronických zařízení

Příloha B
Manuál

B.1 Instalační příručka pro vývojáře

Aby bylo možno přiložený kód znovu zkompilovat a zpustit, je nutné mít nainstalované a licencované nástroje Xamarin. Ty lze stáhnout ze stránek firmy Xamarin (http://xamarin.com/download). Pro kompilaci iOS aplikace je nutné iOS SDK, které je dostupné pouze pro OS X. Lze využít Xamarin Studio pro Mac nebo Visual Studio s Xamarin Build Serverem na zařízení s OS X a iOS SDK. Aplikace byla vyvinuta ve Visual Studiu a projekt nebyl v Xamarin Studiu testován. Pro spuštění aplikace na přístroji je nutné mít zařízeni registrované s vývojářským účtem pro iOS a s ním i správně nastavený XCode. Z tohoto důvodu není dodaný instalační balíček pro iOS.

Návod pro nastavení přístroje lze nalézt na adrese http://developer.xamarin.com/guides/ios/getting_started/installation/device_provisioning/.

Návod pro nastavení vývojového prostředí s Visual Studiem lze nalézt na adrese http://developer.xamarin.com/guides/ios/getting_started/installation/windows/.

Pro kompilaci aplikace pro Android je nutné Android SDK. Instalaci správného Android SDK zařídí instalační program platformy Xamarin.

Spouštění projektu ve Visual Studiu nebo v Xamarin Studiu je velmi podobné. Nejdříve vyberte kýžený projekt jako výchozí (Solution Explorer > pravé tlačítko myší na projekt > Set as Startup Project) a dejte Debug v panelu nástrojů. Pokud je vše korektně nastaveno zobrazí se zvolený emulátor a aplikace se na něm nainstaluje a spustí. V případě, že chcete aplikaci spustit na zařízení, připojte zařízení k počítači (K build serveru u iOS + Visual Studio) a vyberte jej na panelu nástrojů jako cílové.

B.1.1 Testy

Unit Testy jsou v projektu CashBob.Mobile.UnitTests, který je v podstatě samostatnou aplikací. Pro spuštění testů nejdříve vyberte tento projekt jako

výchozí (Solution Explorer > pravé tlačítko myší na projekt > Set as Startup Project) a dejte Debug v panelu nástrojů.

UI Testy jsou jako normální Unit testy spustitelné na počítači pomocí zabudovaného test runneru Visual Studia. Momentálně jsou nakonfigurované jen pro platformu Android. Před spouštěním testů je nutné zapnout vyžadovaný emulátor na kterém mají testy běžet.

B.2 Uživatelský manuál aplikace

K instalaci aplikace na zařízení s operačním systémem Android stačí stáhnout .apk soubor na Váš Android přístroj a spustit ho. Operační systém Vás potom provede instalací. Aplikace vyžaduje minimální verzi Android v4.4 (KitKat).

K instalaci aplikace na zařízení s operačním systémem iOS je potřeba mít zařízení zaregistrováno u vývojářského účtu pro iOS. Instalace je pak nejsnažší s kompilací pomocí IDE.

B.2.1 Server

Tento krok je volitelný, jelikož lze využít testovací server v cloudu, který byl připraven jako součást této práce.

Lokální server spustíte pomocí programu "CashBobServer.exe"z přibaleného archivu (CashBobServerShade-distribution.zip).

Funčnost serveru lze ověřit zadáním adresy `http://localhost:9998/rest` do Vašeho prohlížeče, pokud se Vás prohlížeč dotáže na jmého a heslo, vše je v pořádku.

B.2.2 Nastavení a přihlášení

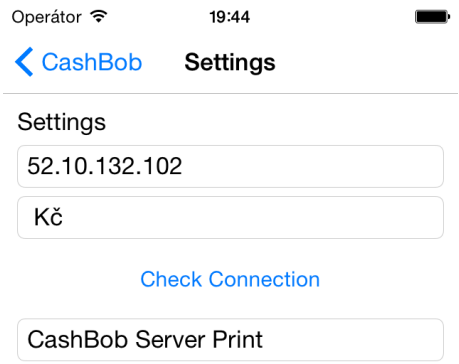
Nejdříve je nutné nastavit IP adresu serveru. Tu lze nastavit ze stránky nastavení na kterou se dostanete stisknutím tlačítka Settings (viz obr. B.2).

Tablet a počítač s CashBob serverem musí být na stejné síti. Do políčka IP Adresa vyplňte IP adresu počítače, na kterém běží server. V případě cloudu je adresa o 52.10.102.132. Adresu lze zjistit pomocí cmd a příkazu ipconfig. Měnu není nutné vyplňovat, ale pro české koruny se vyplňuje " Kč".

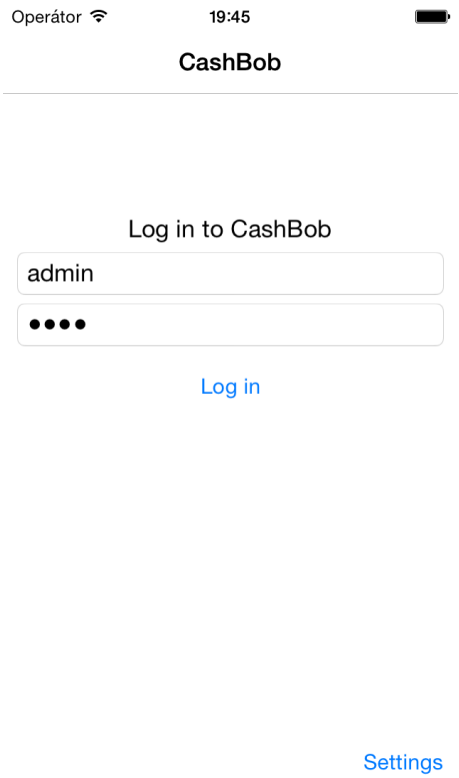
Správnost IP adresy lze otestovat stisknutím tlačítka "Check Connection"viz obr. B.1. Bez funkčního spojení nebude možno se přihlásit.

Z nastavení se dostanete tlačítkem zpět nebo stiskem názvu stránky (Settings na obrázku B.1).

K Přístupu do aplikace je nutné se přihlásit pod svým přihlašovacím jménem. Výchozí jméno a heslo pro přihlášení je "admin" a "pass" u přibaleného i cloudového serveru.



Obrázek B.1: Nastavení



Obrázek B.2: Přihlášení

B.2.3 Připojení tiskárny Star Micronics SM-T300

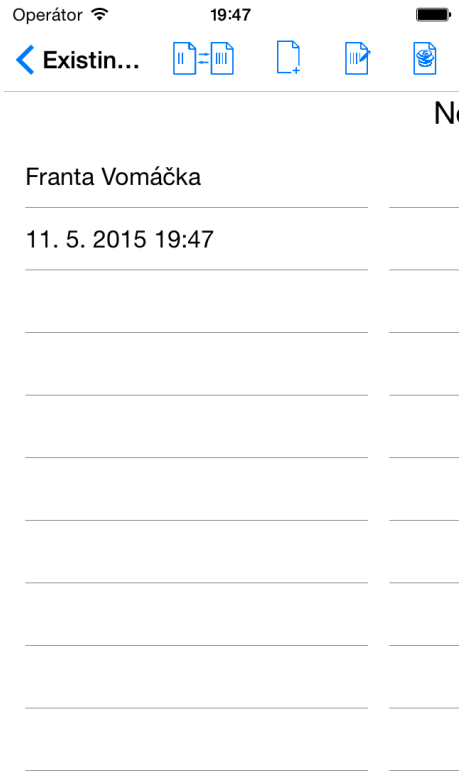
Jak zprovoznit a připojit tiskárnu k zařízení je popsáno v manuálu tiskárny, který je dostupný na http://www.starmicronics.com/support/manual.aspx?printerCode=SM-T300. Po nabití baterie stiskněte tlačítko POWER po dobu 5 vteřin pro zapnutí tiskárny. Potom spárujte tiskárnu se svým zařízením pomocí technologie Bluetooth. Tiskárna se objeví pod názvem Star Micronics a PIN její 1234. Po úspěšném spárování můžete s tiskárnou tisknout.

B.2.4 Funkce aplikace

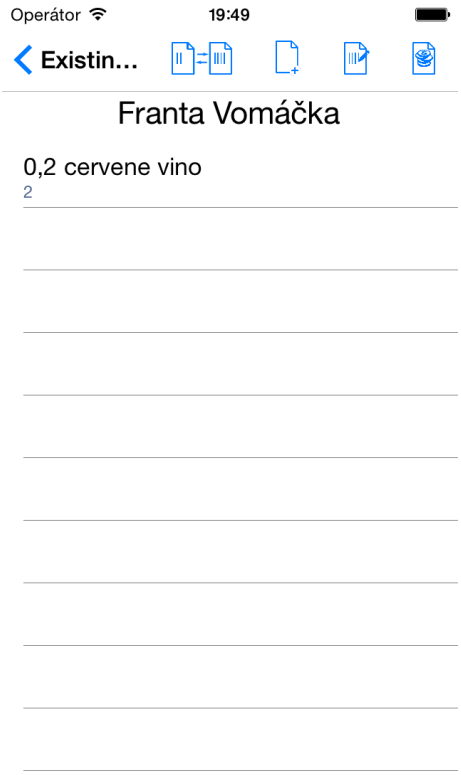
Po přihlášení do aplikace se dostanete na stránku s existujícími účty viz obrázek B.3. Jako většina stránek v aplikaci má i tato stránka vysouvací

postranní panel. Tento panel bude na telefonech schovaný a musí být vysunut tahem z levého okraje obrazovky. Na tabletech bude vyditelný vždy pokud bude na displayi dostatek místa. Z této stránky lze procházet účty, kterých seznam naleznete v postranním panelu viz obrázek B.3. V hlavní části obrazovky je pak přehled aktivních objednávek právě zvoleného účtu jak je vidět na obrázku B.4.

Z této stránky jsou na horní liště nástrojů dostupné další funkce ve formě ikonek. Ikonky nemají pod nimy popis z důvodu nedostatku místa, ale popis lze ukázat u jakékoliv ikonky v aplikaci podržením prstu na ikonce. Jedná se o přesouvání položek mezi účty, vytvoření nového účtu, vytvoření nové objednávky na právě zvolený účet a placení položek zleva do prava.



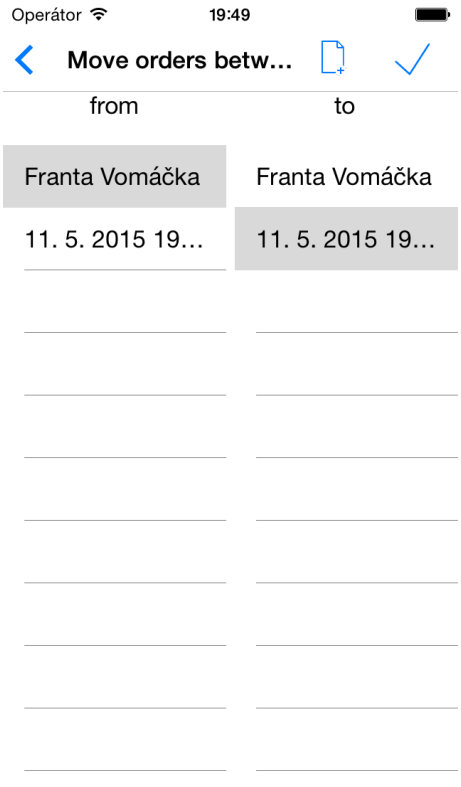
Obrázek B.3: Existující účty



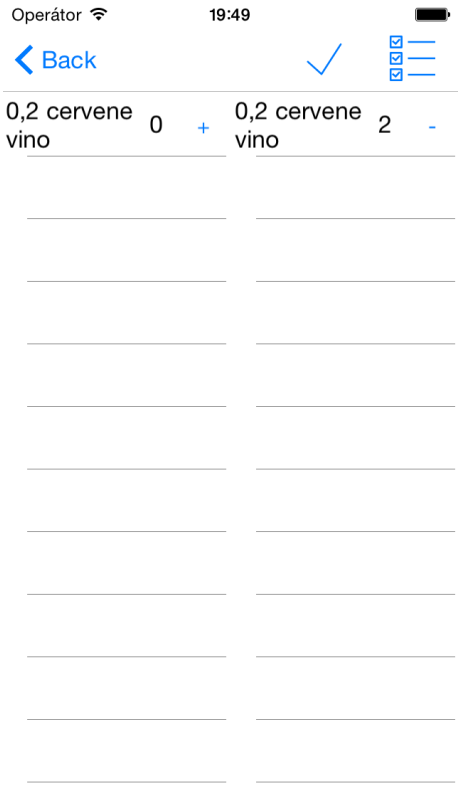
Obrázek B.4: Obsah účtu

První z funkcí je přesun položek mezi účty. Tato funkce se stává z dvou kroků a umožňuje přesunout aktivní objednávky z jednoho účtu na druhý. V prvním kroku jak ukazuje obrázek B.5 vybereme zdrojový a cílový účet. Pokud cílový účet ještě neexistuje, můžeme ho vytvořit zvolením na liště nástrojů ikonky vytvořit nový účet (první ikona zleva). Až vybereme cílový a zdrojový účet potvrdíme volbu pomocí ikonky potvrdit (druhá zleva).

V druhém kroku vybíráme které položky přesunout. Položky volíme k přesunu pomocí tlačítka plus u dané položky. Počet zvolených položek jednoho typu lze upravit pomocí příslušného tlačítka mínus. V liště nástrojů jsou dostupné dvě funkce. První z nich (zleva) potvrdí výběr, druhá automaticky přesune všechny dostupné položky.



Obrázek B.5: Přesunout položky mezi účty - výběr účtů

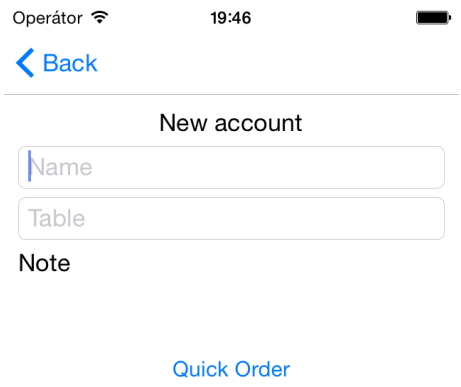


Obrázek B.6: Přesunout položky mezi účty - výběr položek

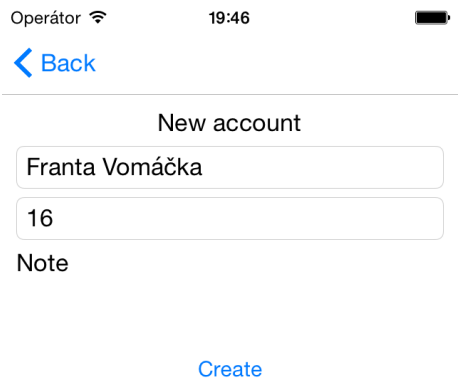
Druhou funkcí dostupnou z hlavní stránky je vytvoření nového účtu. Tento dialog je identický pro vytváření účtu z hlavní stránky nebo ze stránky přesouvání položek. Pokud je dialog prázdný, vypadá jako na obrázku B.7. V tomto případě nelze vytvořit účet, protože nemá název, ale je možné vytvořit rychloobjednávku. Tato volba vytvoří účet s časovým údajem jako názvem. Takovýto účet je určen pro rychlé naplnění a zaplacení.

Pokud specifikujeme název, změní se tlačítko rychloobjednávka na tlačítko vytvořit. Dále můžeme přidružit k účtu specifický stůl a napsat poznámku.

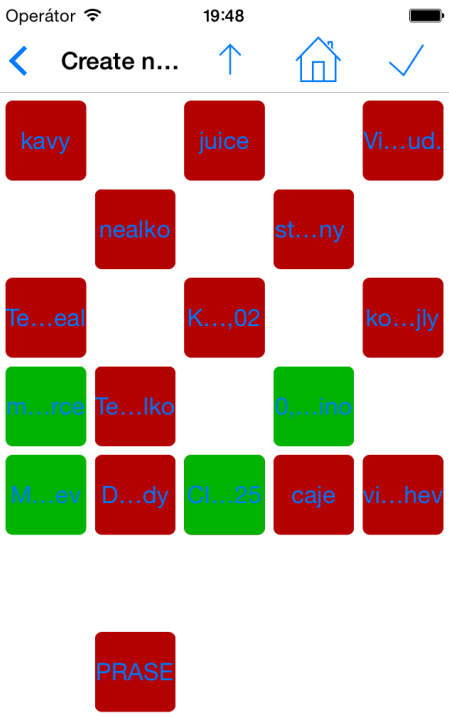
Třetí funkcí dostupnou z hlavní stránky je vytvoření objednávky. Jednotlivé položky dostupné k objednání jsou rozděleny do mřížkového menu



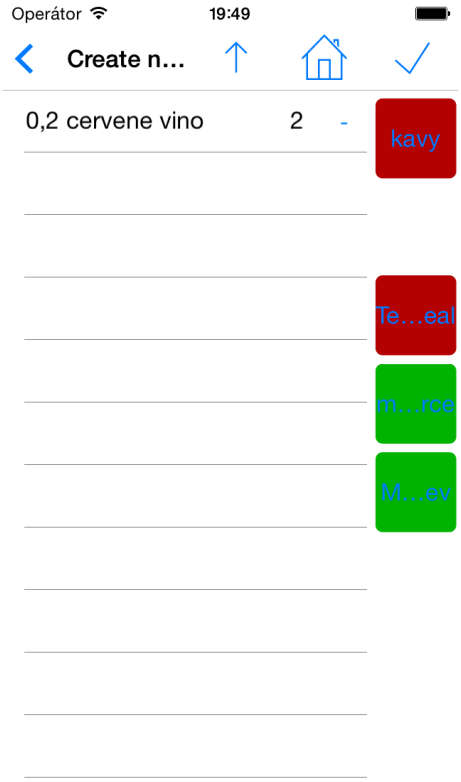
Obrázek B.7: Rychloobjednávka



Obrázek B.8: Vytvořit účet



Obrázek B.9: Vytvořit objednávku



Obrázek B.10: Vytvořit objednávku - postranní panel

(obrázek B.9). Některé položky reprezentují objednatelnou položku, jiné zas podmenu. V tomto případě se jsou rozděleny do dvou barev: zelené a červené v tomto pořadí. Zvolené položky se pak plní do postranního panelu viz obrázek B.10. Počet položek jednoho typu lze upravit pomocí příslušného tlačítka mínus.

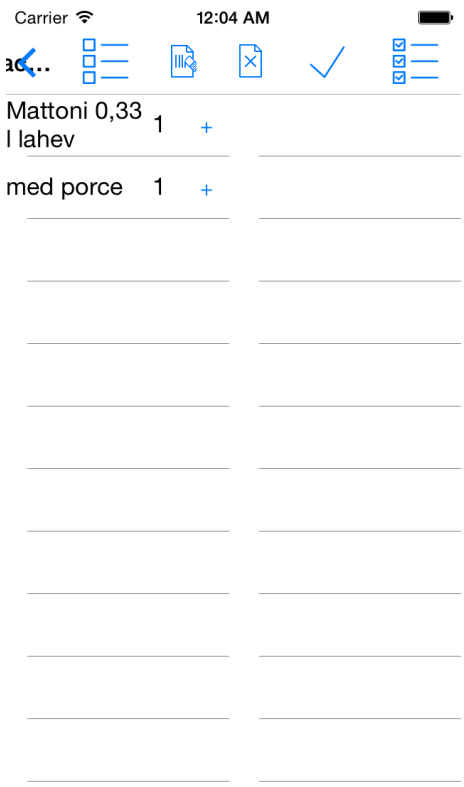
V liště nástrojů jsou dostupné funkce pro navigaci stromovou strukturou menu. Ikonky reprezentují zleva do prava přejít o úroveň nahoru, přejít do počáteční úrovně a vytvořit objednávku.

Poslední funkcí je platba objednávek (obrázek B.11). Z tohoto dialogu jsou dostupné funkce pro práci s objednávkami. Jsou to vyčistit výběr, stornovat položky, zaplatit vybrané a zaplatit všechny. Po vybrání položek se zobrazí druhý dialog, viz obrázek B.12, odkud lze zkontrolovat platbu, zaplatit účet, nebo pouze vytisknout účtenku bez placení. Poslední tlačítko placení zruší a vrátí uživatele na hlavní obrazovku.

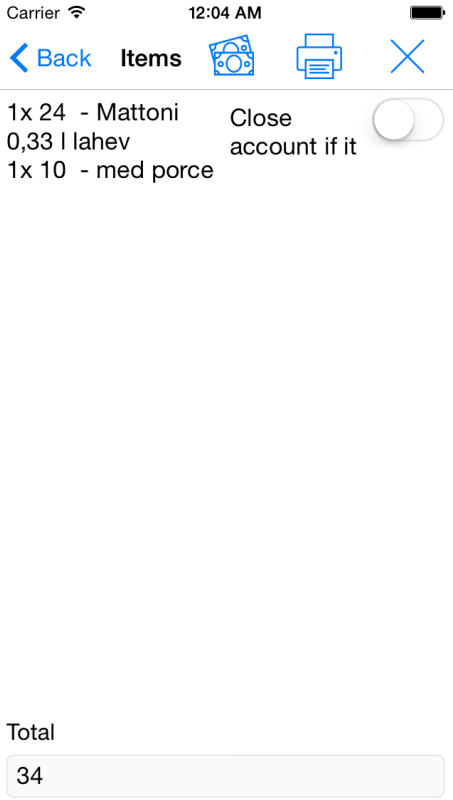
Po zaplacení aplikace automaticky vytisk účtenku. Pokud v nastavení není nastavena výchozí tiskárna, zobrazí se dialog s výběrem až tří možností.

Aplikace může nabídnout některé z následujících voleb, závisle na tom, které daná platforma podporuje.

- **Apple AirPrint**
  Tato volba zobrazí systémový dialog pro tisk. Účtenku pak lze tisknout s AirPrint kompatibilními WiFi tiskárnami.
- **Android Print**
  Tato volba zobrazí systémový dialog pro tisk. Účtenku pak lze tisknout s podporovanými WiFi tiskárnami nebo uložit jako PDF.
- **CashBob Server Print**
  Tato volba pošle žádost aplikaci CashBob Server na počítač, která vytiskne účtenku na výchozí tiskárnu počítače. Tuto funkci nelze využít s cloudovým serverem, protože k němu není připojená žádná tiskárna.
- **Star Portable Printer**
  Tato volba umožňuje tisk s mobilní tiskárnou Star Micronics SM-T300. Tiskárna musí být připojena k zařízení přes Bluetooth, aby mohla být účtenka vytisknuta.



Obrázek B.11: Zaplatit - výběr položek



Obrázek B.12: Zaplatit - potvrzení

Příloha C
Obsah přiloženého CD

- **/Bin**
  - **CashBob.Mobile.apk** - instalační balíček mobilní aplikace CashBob pro operační systém Android
  - **CashBobServer.zip** - archiv s aplikací CashBob Server
  - **CashBobClient.zip** - archiv s aplikací CashBob Client
- **/Source**
  - **/CashBob.Mobile** - zdrojové kódy mobilních aplikací
  - **/CashBob** - zdrojové kódy systému CashBob
- **/Doc**
  - **/Design** - grafické produkty této práce
  - **/Thesis** - zdrojové soubory textové části práce (LaTeX)
  - **/Documentation** - vygenerovaná HTML dokumentace k mobilní aplikaci
  - **manual.pdf** - samostatná instalační a uživatelská příručka
  - **drykjan_2015bach.pdf** - text bakalářské práce ve formátu PDF
  - **vykaz.xlsx** - Výkaz práce ve formátu aplikace Microsoft Excel
- **/Video**
  - **uitest.mp4** - Video-ukázka probíhajícího UI testu
- **readme.txt** - textový soubor s popisem obsahu CD