VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

SPOLEČNÝ VÝVOJ APLIKACE PRO VÍCE MOBILNÍCH PLATFOREM

DEVELOPMENT OF APPLICATION FOR MUTLIPLE MOBILE PLATFORMS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **VIKTOR SEIDL**
AUTHOR

VEDOUCÍ PRÁCE **Ing. ZBYNĚK KŘIVKA, Ph.D.**
SUPERVISOR

BRNO 2018

Zadání bakalářské práce/20412/2017/xseidl03

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2017/2018

Zadání bakalářské práce

Řešitel: **Seidl Viktor**
Obor: Informační technologie
Téma: **Společný vývoj aplikace pro více mobilních platforem**
Development of Application for Mutliple Mobile Platforms
Kategorie: Uživatelská rozhraní

Pokyny:

1. Seznamte se s vývojem aplikací s grafickým uživatelským rozhraním pro více cílových platforem (např. Windows (UWP), iOS, Android) s následným zaměřením na technologie Xamarin, Xamarin.Forms a Prism for Xamarin.Forms.
2. Dle pokynů vedoucího navrhněte demonstrační aplikaci pro platformy UWP, Android a iOS.
3. Následně aplikaci implementujte se zaměřením na odladění funkčnosti a vzhledu alespoň pro jednu vybranou platformu.
4. Kromě funkčního testování proveďte i demonstraci funkčnosti aplikace i na ostatních podporovaných platformách. V rámci testování dokumentujte problémové aspekty na různých platformách a pokuste se navrhnout jejich řešení.
5. Konkrétně (s využitím vytvořené aplikace) i obecně porovnejte nativní aplikace a aplikace využívající Xamarin z různých hledisek (velikost, výkonnost, odezva, uživatelské rozhraní apod.) alespoň pro jednu platformu.

Literatura:

- Charles Petzold: Creating Mobile Apps with Xamarin.Forms. Microsoft Press, 2016.
- Developer Center - Xamarin. *Xamarin* [online]. Xamarin, 2017 [cit. 2017-10-22]. Dostupné z: https://developer.xamarin.com/
- dle doporučení vedoucího

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Křivka Zbyněk, Ing., Ph.D.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2017
Datum odevzdání: 16. května 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Vo všeobecnosti existuje niekoľko platforiem pre vývoj mobilných aplikácií. Na túto problematiku nadväzuje aj bakalárska práca, ktorá sa zaoberá vývojom aplikácií pre platformy ako napríklad Android, iOS a Windows Phone. Vhodný nástroj na zjednotenie všetkých troch platforiem ponúka technológia Xamarin.

Xamarin, patriaci pod technológiu .NET framework, je postavený na základe voľno šíriteľného projektu Mono. Touto technológiou bola pre bakalársku prácu vytvorená aplikácia s názvom Mobilní Objednávání Kredit 2. Aplikácia slúži na objednávanie jedál v systéme Kredit od spoločnosti ANETE spol. s r.o. Pre účely tejto práce bola aplikácia vyladená na platformu Android a následne porovnaná s dostupnou verziou aplikácie vyvinutou v programovacom jazyku Java. Aplikáciu sa podarilo úspešne naprogramovať pre platformy Android a iOS a následne nahrať do príslušných obchodov danej platformy, kde sú aplikácie poskytnuté užívateľom.

Abstract

In general, there are several platforms for mobile application development. This bachelor thesis is also related to the development of applications for platforms such as Android, iOS and Windows Phone. Xamarin technology is a suitable tool for unifying all three platforms.

Xamarin belongs to the technology of .NET framework and it is based on the open-source Mono project. With this technology, an application called Mobilní Objednávání Kredit 2 was created for the bachelor thesis. The application serves for ordering meals in the Kredit system from ANETE spol. s r.o. For the purpose of this work, the application was improved for the Android platform and then compared to the available version of the application developed in the Java programming language. The application was successfully programmed for Android and iOS platforms and then uploaded to the appropriate stores on the platforms where the applications are offered to users.

Kľúčové slová

rámec, NuGet, C#, PCL, vkladanie závislostí, Iconize, navigačná služba, MVVM, platforma, Android, iOS, Xamarin, Mono, Prism

Keywords

framework, NuGet, C#, PCL, dependency injection, Iconize, navigation service, MVVM, platform, Android, iOS, Xamarin, Mono, Prism

Citácia

SEIDL, Viktor. *SPOLEČNÝ VÝVOJ APLIKACE PRO VÍCE MOBILNÍCH PLATFOREM*. Brno, 2018. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Zbyněk Křivka, Ph.D.

SPOLEČNÝ VÝVOJ APLIKACE PRO VÍCE MOBILNÍCH PLATFOREM

Prehlásenie

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing.Zbyněka Křivku,Ph.D. Další informace mi poskytli pán Ing. Tomáš Juříček ze spoločnosti ANETE spol. s r.o.. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Viktor Seidl
14. mája 2018

Poďakovanie

Veľmi rád by som sa poďakoval pánovi Ing. Zbynkovi Křivkovi, Ph.D. za čas, ktorý mi venoval a odbornú konzultáciu pri riešení bakalárskej práce. Poďakovanie taktiež patrí pánovi Ing. Tomášovi Juříčkovi ako externému konzultantovi, ktorý ma pri technológii Xamarin usmerňoval a poskytol mi cenné rady. Na záver sa chcem poďakovať aj firme ANETE spol. s r.o., ktorá poskytla hardvérové vybavenie na tvorbu mobilnej aplikácie.

Obsah

1 Úvod **3**

2 Mobilné platformy **4**
2.1 Android 4
2.2 iOS 5
2.3 Windows Phone 5
2.4 Vývoj pre viaceré mobilné platformy 5

3 Xamarin **7**
3.1 Jednotlivé platformy 9
3.2 Xamarin.Forms 10
3.3 Xamarin - inštalácia 11
3.4 Framework Prism 12

4 Návrh a implementácia **14**
4.1 Návrh aplikácie 14
4.2 Získavanie dát zo servera 15
4.3 Implementácia v Xamarin.Forms 16
4.3.1 Prism registrácia 17
4.3.2 Navigačný panel 18
4.3.3 Špecifické vlastnosti pre platformu 19

5 Štatistiky **21**
5.1 MobilKredit Android vs. MobilKredit 2 Xamarin.Forms 21
5.1.1 Prihlasovanie užívateľa 21
5.1.2 Objednávanie jedla 23
5.2 Android vs. Xamarin.Forms 25
5.2.1 Záťaž aplikácií na pamäť 25
5.2.2 Cyklické odosielanie požiadavky na server 26

6 Testovanie - chyby a ich riešenia **31**
6.1 Optimalizácia ListView 32

7 Zverejňovanie aplikácie **33**
7.1 Android 33
7.2 iOS 33

8 Záver **35**

Literatúra **36**

Prílohy **37**

A Tabulky výsledkov pre štatistiky **38**

B Obrázky Mobilní Objednávání Kredit 2 **40**

C Obsah pamäťového média **43**

Kapitola 1

Úvod

Vo všeobecnosti existuje niekoľko platforiem pre vývoj mobilných aplikácií. Do užívateľského popredia sa dostali napríklad Windows Phone, iOS alebo Android. Cieľom tejto bakalárskej práce je zoznámiť sa s vývojom aplikácií, ktoré je možné spustiť pod viacerými mobilnými platformami, taktiež nazývané ako multiplatformové aplikácie. Pre tieto účely sa práca zameriava na technológiu Xamarin, ktorá spojuje platformy Android, Windows Phone a iOS. Touto technológiou bola naprogramovaná aplikácia pre firmu ANETE spol. s r.o. na objednávanie jedál pod názvom Mobilní Objednávání Kredit 2 alebo pod skráteným názvom MobilKredit. Návrh grafického rozhrania vychádza z pôvodnej aplikácie Mobilní Objednávání Kredit, ktorá je naprogramovaná pre platformu Android. Tento názov sa odlišuje od pôvodnej aplikácie číslicou 2 a to z dôvodu, že na Google Play sa nemôžu nachádzať aplikácie s rovnakým názvom. Aplikácia programovaná vybranou technológiou Xamarin by mala bežať aspoň pod dvoma mobilnými platformami. Výber platforiem sa uberal smerom k platforme iOS a Android nakoľko tretia platforma Windows Phone upadla a jej vývoj v dnešnej dobe nie je spoločnosťou Microsoft podporovaný. Aplikácie boli v priebehu vývoja testované vo virtuálnych zariadeniach. Pre vývoj sa použil nástroj Visual Studio pre operačný systém Windows, taktiež bolo potrebné mať k dispozícii počítač od firmy Apple, ktorý zabezpečila firma ANETE spol. s r.o.

V prvej kapitole tejto práce sú popísané rôzne technológie slúžiace na vývoj multiplatformových aplikácii ako napríklad platformy iOS, Adroid, Windows Phone alebo Blackberry. Ďalšia kapitola sa venuje podrobne technológii Xamarin, prostredníctvom ktorej sa aplikácia vyvíjala a za ňou nasleduje popis vyvíjanej aplikácie, postupy a návrhy. V neposlednom rade sa práca venuje aj zverejňovaniu aplikácie na jednotlivých obchodoch slúžiacich na distribúciu aplikácií pre platformu Android a iOS. Najdôležitejšiu časť práce tvoria štatistiky medzi natívnymi aplikáciami, vzniknuté na daných platformách a aplikáciami, ktoré sú vytvorené technológiou Xamarin. Táto práca zahŕňa komplexné informácie o problematike multiplatformových zariadení a zhodnocuje plusy a mínusy pre vytváranie aplikácii technológiou Xamarin.

Kapitola 2

Mobilné platformy

Moderná doba prináša veľké množstvo mobilných telefónov s rôznymi mobilnými platformami. Samozrejme, k tým najčastejšie používaným patria platformy iOS, Android alebo Windows Phone. Mobilná platforma je softvér, ktorý beží na mobilnom zariadení a umožňuje nám jeho pohodlné ovládanie. V dnešnej dobe sa mobilný telefón nevyužíva len na telefonovanie alebo posielanie sms, ale jeho funkcionalita je ďaleko rozsiahlejšia. Na každú mobilnú platformu je možné vyvíjať aplikácie, ktoré na nej bežia a tie nám môžu uľahčiť rôzne činnosti, napríklad aplikácia kalendár. Existujú aplikácie, ktoré nám neslúžia na uľahčenie života, ale na zábavu (rôzne hry). Taktiež sa pomocou nich môžeme vzdelávať. Súčasná doba umožňuje vyvíjať aplikácie kýmkoľvek so znalosťami programovacieho jazyka potrebného pre danú platformu. Pre Android to je napríklad Java alebo Kotlin, iOS má Objective-C a Swift a pre Windows Phone C#, prípadne iné jazyky, ak to daná platforma podporuje. Taktiež je možné použiť nástroje ako je Xamarin, ktorý táto práca bližšie popisuje.

Užívateľ má v dnešnej dobe širokú možnosť výberu operačného systému pre smartfón. Najväčší podiel na trhu má Android a iOS[1].

2.1 Android

Android je rozsiahly operačný systém – jeden z najpoužívanejších na svete, ktorý bol vytvorený firmou Android, Inc., ktorú v roku 2005 získala spoločnosť Google. Jeho podiel na trhu je približne 80 %. Tento podiel je tak vysoký z toho dôvodu, že Android je open source platforma, ktorá sa nepoužíva len pre mobilné platformy, ale taktiež pre navigácie, tablety a iné zariadenia. Najnovšie sa používa aj v automobiloch. Jeho operačný systém je založený na jadre Linuxu a vývojárom poskytuje efektívne nástroje pre vývoj ako napríklad Software Development Kit. Túto platformu je možné programovať v jazyku Java, prípadne ako kombináciu s C++. Najnovšia dostupná verzia počas písania tejto práce je 8.1 pod názvom Oreo. Všetky verzie systému Android sú pomenované po sladkostiach [2].

- Android 4.0 Ice Cream Sandwich
- Android 5.0 Lollipop
- Android 7.0 Nougat

[1] bližšie štatistiky je možné nájsť v nasledujúcom článku
http://gs.statcounter.com/os-market-share/mobile/worldwide

2.2 iOS

Ďalší operačný systém pre mobilné platformy patrí do finančne vyššej kategórie, a to aj napriek tomu, že firma za posledný rok vydala verzie zariadení s nižšou finančnou hodnotou v porovnaní s ich predchodcami. Hovoríme o operačnom systéme, ktorý pochádza od firmy Apple Inc, pomenovaný iOS s pôvodným názvom iPhone OS. Je určený pre zariadenia od rovnakej spoločnosti ako operačný systém Mac OS. iOS je možné použiť pre zariadenia ako je iPhone, iPod Touch, iPad. iOS vznikol z Mac OS X, ktorý je určený pre osobné počítače. Túto platformu je možné programovať v jazyku Swift alebo Objective-C, a to vo vývojárskom nástroji Xcode. Aktuálna najnovšia verzia iOS je iOS 11. Nakoľko operačný systém iOS nie je voľne šíriteľný, je problematické nahrať akúkoľvek aplikáciu do zariadenia bez použitia ich oficiálneho obchodu Apple Store, kde je každá aplikácia pred vydaním otestovaná.

2.3 Windows Phone

Windows Phone je operačný systém vyvinutý firmou Microsoft. Predchodcom Windows Phone bol Windows Mobile, ktorý nebol veľmi známy. Operačný systém Windows Phone je zameraný na užívateľské rozhranie s dizajnom Metro, ktorý je použitý aj v operačných systémoch stolných počítačov, a to od verzie Windows 8. Windows Phone sa ani napriek pokročilej technológii contiunuum neuchytil na trhu. Contiunuum je technológia, ktorá umožňuje zmeniť mobilné zariadenie pripojením externého hardvéru na stolný počítač s bežnými perifémymi zariadeniami (myš, klávesnica, obrazovka). Spoločnosť Microsoft zvažuje ukončenie podpory pre tento mobilný operačný systém na rok 2018. Túto informáciu potvrdzuje aj verzia .NET štandardu 2.0[2], ktorá funguje len na operačných systémoch Windows Fall Creation. Vzhľadom na to, že verzia Fall Creation nie je prispôsobená na mobilné zariadenia, tak po aktualizovaní .NET štandardu 2.0 nie je možné na platformu Windows Phone aplikáciu vyvíjať. V dôsledku toho sa v prílohe bakalárskej práce nevyskytujú ukážky na platformu Windows Phone.

2.4 Vývoj pre viaceré mobilné platformy

Každý človek má odlišné požiadavky, a preto vznikajú rôzne platformy, či už v mobilnej sfére alebo inej. Samozrejme, to prináša rôzne úskalia a otázky. V dnešnej dobe sú dve prioritné platformy - Android a iOS. Android beží na viacerých zariadeniach, takže dominuje kvantitatívne. Okrem toho sú užívateľmi používané aj iné platformy ako je Windows Phone alebo BlackBerry. Každý vývoj, či sa už jedná o rovnakú aplikáciu alebo nie, stojí čas, peniaze, ľudské zdroje a ani po vydaní aplikácie nie je koniec. Každú aplikáciu je potrebné udržiavať, prípadne rozširovať podľa požiadaviek zákazníka. Na základe toho vznikli technológie, ktoré umožňujú jednotný vývoj pre viac mobilných platforiem. Napríklad nástroj Sencha umožňuje písať program pre mobilné platformy v jazyku HTML5, ale umožňuje vývoj len pre platformy iOS a Android. PhoneGap, ktorý vlastní firma Adobe, umožňuje písanie aplikácií v jazyku HTML5, CSS a JavaScript. Okrem toho umožňuje aj generovanie pre iOS a Androi, či BlackBerry a Windows Phone. Existujú aj špecializované programy, ktoré sa zameriavajú výhradne na vývoj hier pre mobilné platformy ako je Unity 3D. Unity

[2].NET štandard 2.0 https://docs.microsoft.com/en-us/dotnet/standard/net-standard

3D zahŕňa platformy Android, iOS, Windows a niektoré herné konzoly PlayStation, Xbox, Wii, alebo iné rozhrania Web, Linux.[3]

Táto práca sa bližšie špecializuje na technológiu Xamarin, ktorá predstavuje veľký potenciál do budúcna pre vývoj viacerých mobilných platforiem. Súčasťou technológie je aj jedna z platforiem Universal Windows Platform, ktorú vytvoril Microsoft. Napriek tomu tieto technológie skrývajú problematiku pri vývoji na viacerých mobilných platformy. Každá platforma má iné zobrazovanie základných panelov, navigácií alebo tlačidiel, nehovoriac o ovládaní danej platformy gestami. Práve tieto rozdiely neuľahčujú vývoj na viaceré mobilné platformy.

[3]Viac informácií o týchto platformách je dostupných na stránke http://thinkapps.com

Kapitola 3

Xamarin

Xamarin je technológia na vyvíjanie multiplatformových aplikácií, postavená na základe voľno šíriteľného projektu Mono. Mono je založené na ECMA štandardoch pre Common Language Infrastructure a C#, a preto Xamarin patrí pod technológie .NET framework, ktoré vlastní spoločnosť Microsoft. .NET framework beží prevažne na Microsoft Windows. Obsahuje prostredie pre spúšťanie aplikácii, tak isto obsahuje potrebné knižnice na vývoj a táto trieda knižníc sa volá Framework Class Library (FCL). .NET framework neviedol iba k vývoji mobilných platforiem, ale aj k vývoju vstavaných systémov alebo webových aplikácií. V .NET-e je možné používať kombináciu viacerých jazykov takzvanú interoperabilitu (každý jazyk môže používať kódy napísané v iných programovacých jazykoch).

Súčasťou Mona je [6]:

- .NET kompilátor Roslyn
- Mono-Runtime, poskytujúci Just in Time a Ahead of time kompiláciu
- .NET systém pre Garbage Collector
- Systém pre chovanie vlákien
- Bassic class libraries, sú kompiláciou .NET, rozšírenie pre špeciálne prípady o Mono class libraries.

Xamarin bol založený v roku 2011 a o 5 rokov neskôr ju odkúpila spoločnosť Microsoft. Táto technológia je ešte len v samotnom vzostupe. Jej hlavnou výhodou je vývoj pre všetky mobilné platformy zahŕňajúci aj technológiu UWP (Universal Windows Platfom), ktorá sa na trh dostala len nedávno.

Na počiatku vývoja tejto technológie bolo náročné určiť, ktorý jazyk bude zastrešovať všetky tri platformy. Bolo potrebné vyriešiť 4 základné problémy:

1. Vzory/formy užívateľských rozhraní
    - Ide o rozdiel v navigáciách pri rôznych aplikáciách a stránkach. Rôzne konvencie pri spracovaní dát alebo odlišné zobrazovanie komponentov boli prvé problémy pri návrhu spoločného jazyka.
2. Rozdielne vývojárske prostredia

- Každý programátor pozná pri danej platforme IDE, s ktorým pracoval. Bolo potrebné rozhodnúť, ktoré IDE by bolo najvhodnejšie pre spoločný jazyk.
  - Pre iOS sa používa Xcode na zariadení MAC.
  - Pre Android sa používa Android Studio pre viaceré platformy.
  - Windows vývojári používajú Visual Studio.

3. Rôzne programovacie rozhrania

   - Na daných rozhraniach sú naimplementované podobné komponenty, ale s rôznymi pomenovaniami, napríklad tlačidlo toggle, ktoré zabezpečuje prepínanie medzi dvoma stavmi. Veľmi ľahko by sa tento komponent dal prirovnať vypínaču.
     - V iPage to je view nazývané UISwitch.
     - Na Android zariadeniach widget, nazývaný Switch.
     - Na Windows Runtime API, je spomínaný ovládací prvok pomenovaný ToggleSwitch.

4. Rozdielne programovacie jazyky

   - Programovacie jazyky, ktoré sa používajú pre jednotlivé platformy, sú príbuzné, nakoľko všetky tri jazyky vychádzajú z jazyka C.
     - Objective-C pre iPhone a iPad
     - Java pre Android zariadenia
     - C# pre Windows

Napokon bolo rozhodnuté využiť C#, ktorý bol relatívne mladý programovací jazyk oproti Objectiv-C alebo Java. Neskôr bol rozšírený o lambda funkcie, LINQ a asynchrónne programovanie, čím sa C# stal multiparadigmatickým programovacím jazykom. C# môže byť tradične imperatívny, ale taktiež môže byť rozšírený o deklaratívne a funkcionálne paradigma. Dôležitým faktorom bolo, že .NET Framework podporoval rôzne knižnice, ktoré sa vyskytovali v bežnej práci a v rôznych typoch, napríklad [3]:

- Math
- Collections
- File I/O
- Networking
- Security
- Threading

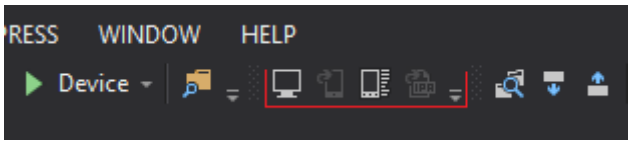
V tomto bode je dôležité prihliadnuť na fakt, že každý kód C# je potrebný vedieť preložiť na danú platformu. Z toho dôvodu sa vývojári Xamarinu prvé tri roky sústredili na vývoj prekladačov. Vyvinuli tri základné sady pre .NET knižnice.

- Xamarin.MAC, ktorá sa vyvinula z MonoMac projektu.
- Xamarin.iOS, ktorá sa vyvinula z MonoTouch.

- Xamarin.Android, ktorá sa vyvinula z Mono for Android alebo MonoDroid.

Pomocou týchto API môžu programátori písať kód pre všetky tri platformy, a pritom môžu používať .NET Framework knižnice [7].

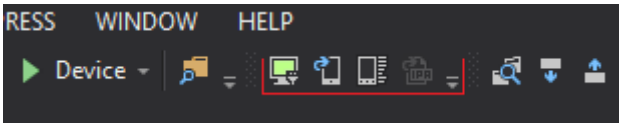
V prípade, že programátor vyvíja aplikáciu na operačnom systéme Windows a aplikácia bude používaná aj na zariadeniach s operačným systémom iOS, je potrebné, aby programátor pre preklad aplikácie mal nastavený vzdialený prístup z Visual Studia na Mac zariadenie, na ktorom sa vzdialený preklad aplikácie vykoná. Knižnice (NuGet) vo Visual Studiu musia byť na oboch zariadeniach rovnakej verzie.



Obr. 3.1: Odpojené Apple zariadenie (Mack mini).

Na obrázku 3.1 môžeme vidieť (zľava od červenej čiary):

- Xamarin Agent: slúži na pripojovanie Visual Studia k zariadeniu Mac.
- Show iOS Simulator: spustenie simulátoru na zariadení Mac. Visual Studio Enterprise má možnosť simulátor v Mac prenášať na Windows. Nie je potrebný vzdialený prístup na Mac. Táto možnosť je od verzie Visual Studio 15.5.1 dostupná aj pre Visual Studio Community a Profesional.
- Device Log: Logovanie zariadenia.
- Show IPA File on Build Server: zobrazenie preloženého súboru. Súbor sa nahráva do iTunes, kde aplikáciu schvália a zverejnia.



Obr. 3.2: Pripojené Apple zariadenie (Mack mini).

3.1 Jednotlivé platformy

Xamarin umožňuje vyvíjať aj pre vybrané platformy, a to tak, že programátor založí projekt pre Xamarin.Android alebo inú platformu (Xamarin.iOS). Výhodou je jednoduchá schopnosť prispôsobenia sa na špecifickú platformu užívateľského rozhrania a využitie jeho maxima. Samozrejme, tento vývoj prináša aj nevýhody. Programátor sa musí naučiť natívne prvky užívateľského rozhrania. Pri samostatnom vývoji na špecifickú platformu nastáva problém, keď zákazník bude mať časom záujem aj o inú platformu, ako len tú, pre ktorú je softvér vyvíjaný v technológii Xamarin. Potom sa musí celá aplikácia prepísať do platformy, o ktorú zákazník žiada. Táto problematika vedie k myšlienke, či nie je vhodnejšie aplikáciu programovať vo framework-u Xamarin.Forms, ktorý je podrobne charakterizovaný v nasledujúcej kapitole.

3.2 Xamarin.Forms

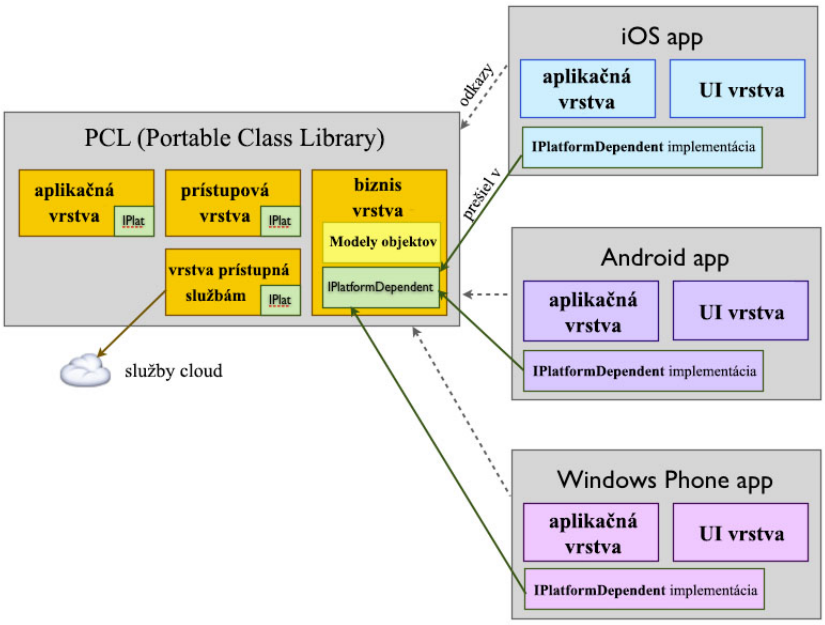
Ako už bolo v prechádzajúcich kapitolách spomenuté, pod technológiou Xamarin je možné vyvíjať programy na technológie Windows, Android a iOS. Xamarin nám umožňuje taktiež založiť projekt, ktorý bude medzi platformami zdielať jeden projekt nazývaný PCL (Portable Class Libraries). Tento projekt nám zdieľa kód naprieč ostatnými platformami, ktoré boli k nemu priradené. Xamarin.Forms má svoje výhody aj nevýhody. Výhody:

1. Centralizované zdieľanie kódu.
2. Refaktorovanie kódu v rámci PCL a špecifického projektu.
3. PCL je ľahké odkazovať na iné projekty v projekte (solution).
4. Tvorba jedného uživateľského rozhrania pre všetky platformy.
5. Nie je potrebné, aby programátor poznal natívne prvky jednotlivých platforiem.

V PCL projekte máme spoločné knižnice, kódy, testy pre všetky platformy, prináša aj svoje nevýhody, a to napríklad:

1. Kedže PCL využíva spoločné informácie medzi viacerými platformami, knižnice špecifické pre platformu v nej nemôžu byť odkazované. Napríklad knižnica Community.CsharpSqlite.WP7.
2. Nesmie obsahovať triedy, ktoré sa vyskytujú v MonoTouch a Mono pre Android.
3. Vzhľadom na to, že sa jedná o nový Framework, tak obsahuje stále chyby, s ktorými sa musí programátor vysporiadať.

Do určitej miery je možné niektoré nevýhody obísť použitím návrhových vzorov Provider alebo Dependency Injection. Nasledujúci diagram 3.3 znázorňuje architektúru aplikácie s viacerými platformami použitím Portable Class Library na zdieľanie kódu, ale taktiež pomocou Dependency Injection na odovzdávanie funkcií závislých na platforme [5]:



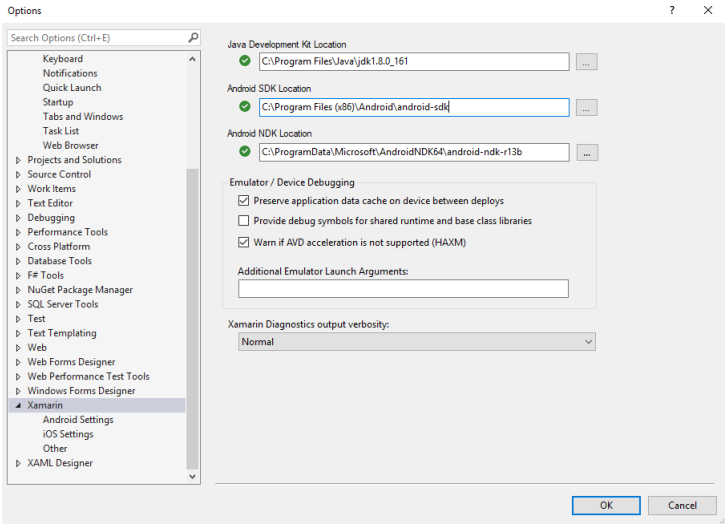
Obr. 3.3: Architektúra aplikácie s viacerými platformami.

Jedna z nevýhod je taktiež čas spúšťania aplikácii pri Portable Class Library projektoch. Tento čas je zásadne rozdielny oproti natívnym aplikáciam alebo aplikáciam v Xamarin-e pre vybranú platformu. V dôsledku toho treba pri vývoji zvážiť, ako užívateľ často pristupuje k aplikácii a či pre neho nebude problém počkať o pár sekúnd naviac pri jej spúšťaní.

3.3 Xamarin - inštalácia

Nevyhnutnou súčasťou inštalácie bolo stiahnutie Visual Studio Community, nakoľko je táto verzia bezplatne dostupná na stránkach Visual Studia[1]. Následne bolo zvolené pri inštalácii tlačidlo „Custom“, kde sa zaškrtla položka C#/.NET Xamarin a taktiež Universal Windows App Development Tools. Po nainštalovaní Visual Studia bolo potrebné skontrolovať Android SDK, či je správne prepojený s Visual Studiom. Skontrolovanie prebiehalo v Tools –> Options –> Xamarin ako je vidieť na obrázku 3.4. V prípade, že sa vyskytolo zelené odškrtnutie v krúžku, zadaný odkaz na cestu k Android SDK je správny.

[1]Visual Studio https://www.visualstudio.com



Obr. 3.4: Nastavenie adresy pre Android SKD, JDK, NDK.

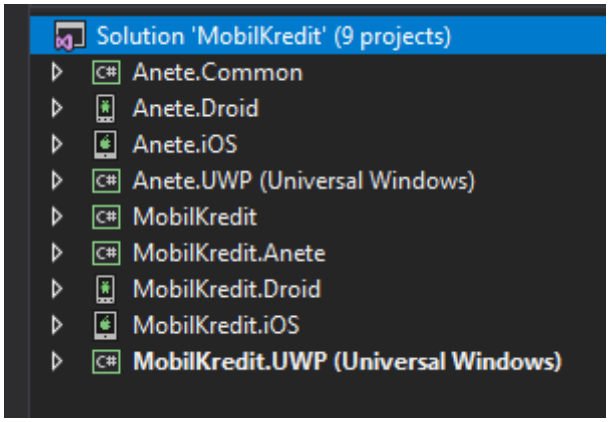
Pre Universal Windows Platform sme museli stiahnuť balíček Microsoft .NETCore .UniversalWindowsPlatform. Tento balíček môžeme stiahnuť pomocou package manager console a príkazu `install-package Microsoft.NETCore.UniversalWindowsPlatform`[2]. Je pravdepodobné, že v novšej verzií Visual Studia už nebude nutné sťahovať tento balíček.

3.4 Framework Prism

Prism je voľne šíriteľný framework pre vytváranie ľahko udržiavateľných a testovateľných XAML aplikácií vo WPF, UWP a v Xamarin technológii. Správy v Prism sú k dispozícii pre každú platformu [1], v našom prípade sa jedná o projekt vo Visual Studiu s názvom MobilKredit a tie sú nezávisle vyvíjané na svojich časových osách. Každý projekt má časovú os zvlášť pre jedno zariadenie, napríklad projekt MobilKredit.Droid, zobrazený na obrázku 3.5, je určený pre zariadenia s operačným systémom Android.

Daný framework poskytuje podporu návrhových vzorov, ktoré sú užitočné pri písaní a udržiavaní aplikácií. Patrí tam napríklad MVVM, vkladanie závislostí, príkazy. Prism má spoločnú základňu, kde sa vyvíja aplikácia pre všetky platformy. Veci, ktoré sú špecifické pre jednotlivé platformy, musia byť implementované v príslušných projektoch pre platformu.

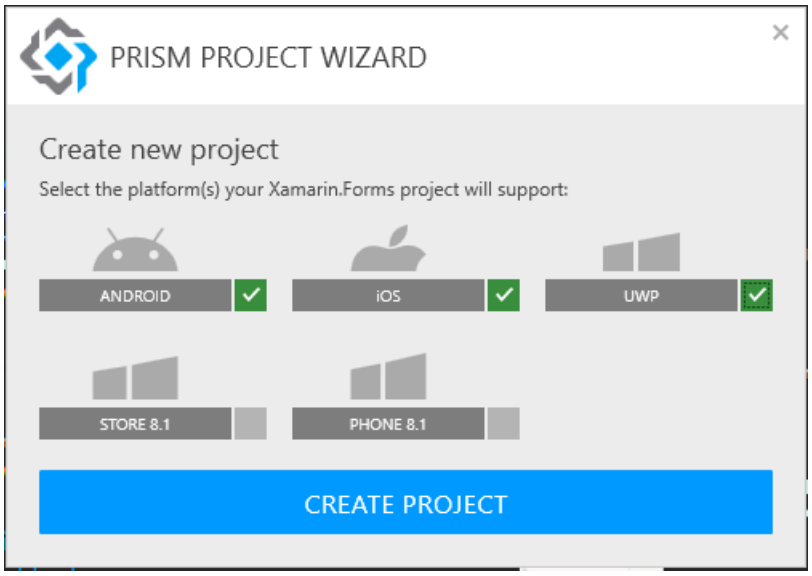
[2] Microsoft.NETCore.UniversalWindowsPlatform
https://www.nuget.org/packages/Microsoft.NETCore.UniversalWindowsPlatform



Obr. 3.5: Visual Studio projekt.

Tento framework je možné stiahnuť priamo v programe Visual Studio (Tools -> Extensions and Updates), kde sa nachádza pod názvom Prism Template Pack[3].

Výhodou framework-u Prism je tiež založenie projektu pre jednotlivé platformy. V technológii Xamarin.Forms je potrebné pridať požadované alebo odstrániť nepotrebné platformy. V prípade Xamarin Prism sa po založení projektu zobrazí okno, v ktorom je možné vybrať platformy, s ktorými chceme pracovať (Android, iOS, atď.).



Obr. 3.6: Výber platforiem vo Visual Studiu.

[3]Prism Template Pack
https://marketplace.visualstudio.com/items?itemName=BrianLagunas.PrismTemplatePack

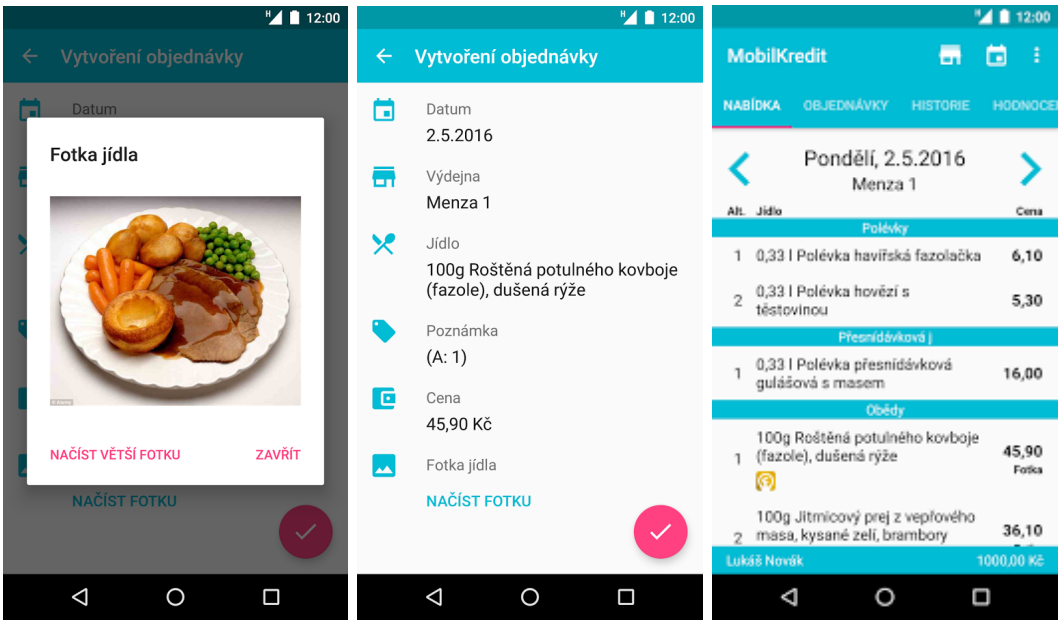
Kapitola 4

Návrh a implementácia

4.1 Návrh aplikácie

Návrh aplikácie nebolo potrebné riešiť vzhľadom na to, že aplikácia bola vyvinutá a zavedená do bežného používania na platforme Android. Úlohou bolo naprogramovať podobnú aplikáciu ako je Mobilní Objednávání Kredit pomocou technológie Xamarin.

Aplikácia slúži na objednávanie jedál rôznych výdajných miest. Pracuje s požiadavkami na vzdialený server, ktorý sťahuje z databázy informácie o jedlách, výdajných miestach, prípadne iné informácie potrebné na prácu s aplikáciou.



Obr. 4.1: Pôvodná aplikácia MobilKreditu v Androide.

Na obrázku 4.1 môžeme vidieť návrh troch rôznych okien v aplikácii pre Android. Na to, aby bol dosiahnutý podobný efekt grafického rozhrania, bolo použitých niekoľko nástrojov. Pre každú platformu bolo potrebné vytvoriť obrázky, či už pre tlačidlá, alebo informatívne obrázky pre užívateľa, ktoré sa budú používať na danej platforme. Problém bol vyriešený

pomocou balíčku Iconize Plugin for Xamarin[1]. Tento balíček zabezpečuje jednotné obrázky pre vyvíjané mobilné platformy Xamarin.Android, Xamarin.iOS alebo Xamarin.UWP. Pred použitím je potrebné balíček inicializovať pre každú platformu samostatne pomocou príkazu:

```
Plugin.Iconize.Iconize.With(new Plugin.Iconize.Fonts.FontAwesomeModule());
```

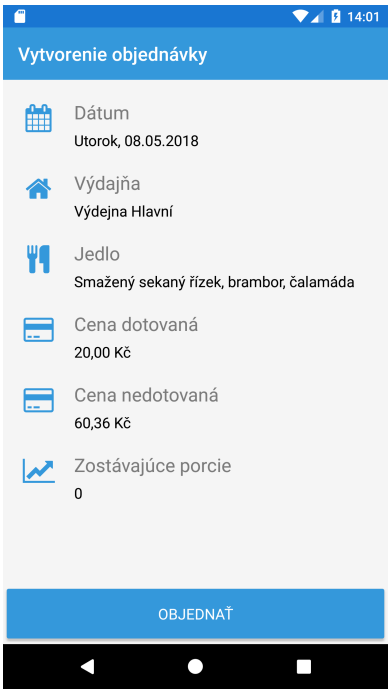
Výpis 4.1: Inicializácia balíčka Iconize pre špecifické platformy.

Pre použitie komponentu s Iconize v XAML súbore, je nutné najskôr zavolať balíček ako atribút v koreni.

```
xmlns:iconize="clr-namespace:FormsPlugin.Iconize;assembly=FormsPlugin.Iconize"
```

Výpis 4.2: Inicializácia balíčka Iconize pre XAML súbor.

Na obrázku 4.2 sa nachádzajú v ľavej časti okna modré piktogramy. Tieto piktogramy boli vytvorené pomocou spomínaného balíčka Iconize[2]. Viac fotografií z aplikácie je možné nájsť v prílohe B tejto práce.



Obr. 4.2: Vytvorenie objednávky v aplikácii MobilKredit 2.

4.2 Získavanie dát zo servera

Získavanie dát zo servera zabezpečila firma ANETE spol. s r.o., ktorá má svoje vlastné API z predchádzajúcej verzie pre Android aplikáciu. API je pripojené ako samostatný projekt MobilKredit.Anete, ako je zobrazené na obrázku 3.5.

[1] Dostupné na https://github.com/jsmarcus/Xamarin.Plugins/tree/master/Iconize
[2] Iconize piktogramy https://fontawesome.com/icons?from=io

API nedokáže stiahnuť informácie o danom výdajnom mieste skôr ako sa užívateľ prihlási. Preto je potrebné vytvoriť takzvanú session, ktorá bude udržiavať spojenie. Session je možné vytvoriť prihlásením pomocou nasledujúceho príkazu.

```
string result = await _orderingProxy.LoginByCardAsync(string);
```

Výpis 4.3: Prihlasovanie užívateľa voči serveru.

`_orderingProxy` je vytvorené pomocou rozhrania `IOrderingService`. Po vytvorení session (prihlásenie užívateľa) je môžeme získavať dáta napríklad o výdajných miestach príkazom v ukážke 4.4, alebo príkazom z ukážky 4.5 získame dáta o aktuálnom jedálnom lístku vo výdajnom miest.

```
var canteen = await _orderingProxy.GetCanteensAsync(DateTime.Now);
```

Výpis 4.4: Požiadavka na zoznam výdajných miest.

```
Collection<MenuItem> menu = await _orderingProxy.GetMenuAsync(date,
    Canteen.IdCanteen);
```

Výpis 4.5: Požiadavka na zoznam jedál.

API komunikuje so vzdialeným serverom, kde sú uložené dáta vo formáte JSON[3]. Dáta sú sťahované asynchrónne, čo umožňuje plynulejší chod grafickému rozhraniu a operáciám na ňom (otváranie panelov, prezeranie menu). Dáta, ktoré sa sťahujú zo servera, nie sú ukladané lokálne na zariadenie, čo má za následok fakt, že aplikácia musí byť neustále pripojená k internetu.

4.3 Implementácia v Xamarin.Forms

Aplikácia Mobilní Objednávání Kredit 2 je naprogramovaná pod technológiou Xamarin a jeho rozšírením Xamarin.Forms. V rámci tejto práce je používaný framework Prism 3.4.

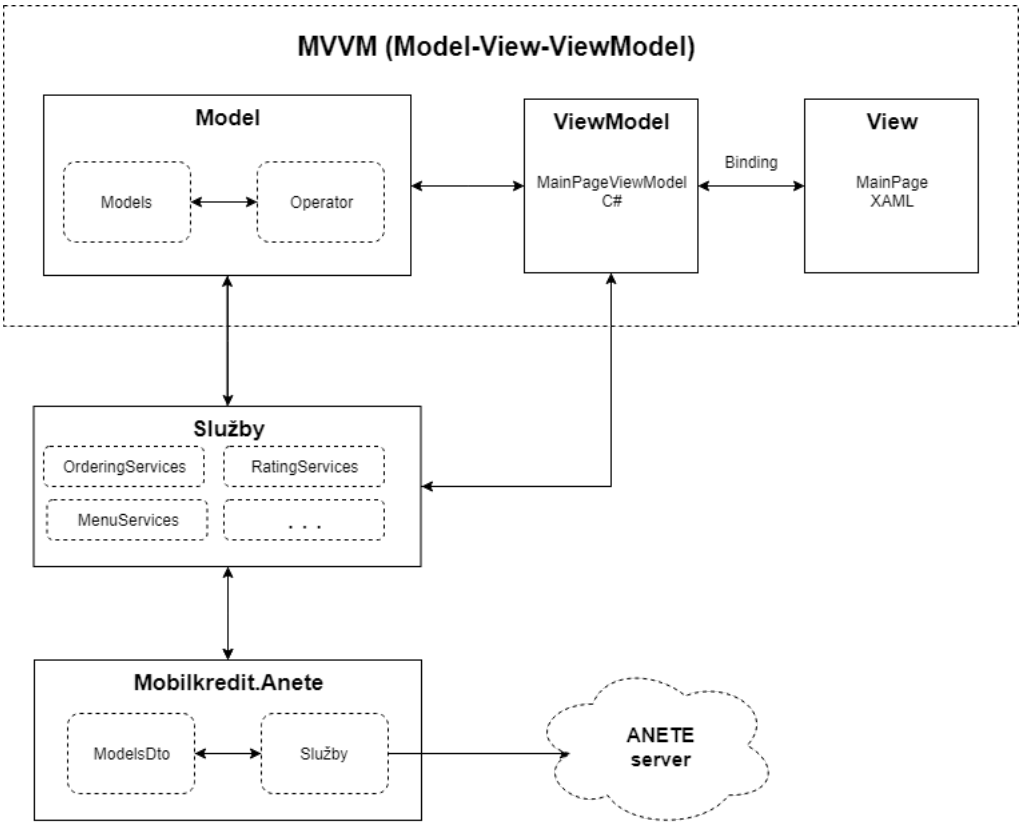
Spomenieme si niečo o PCL projektu, nakoľko je spoločný pre všetky zvolené platformy v projekte. V PCL projekte je používaný MVVM (model-view-viewmodel) návrhový vzor, založený na oddelení grafického rozhrania a dátovej zložky [8]. Popis jednotlivých zložiek:

- Model - popisuje dáta v aplikácii, s ktorými pracuje.
- View - reprezentuje užívateľské rozhranie. V tomto prípade ide o jazyk XAML, pod ktorým sa užívateľské rozhranie programuje.
- ViewModel - drží stav aplikácie a spojuje View a Model vrstvu. S View vrstvou komunikuje pomocou bindingu.

V projekte MobilKredit 2 sa vytvorila ešte jedna zložka s názvom Služby, ktorú znázorňuje obrázok 4.3. Tá slúži na komunikáciu s projektom MobilKredit.Anete od firmy ANETE spol. s r.o.. Tento projekt obsahuje modely s príznakom Dto, ktoré vracia priamo server, prípadne ich posiela aplikácia na server. Dto je v tomto prípade skratka data object. Tieto objekty treba previesť do modelov, ktoré sa používajú v projekte, vo väčšine prípadov sú identické, ale môžu obsahovať niektoré vlastnosti navyše oproti dto modelom. Tieto vlastnosti môžu slúžiť napríklad pri rozhodovaní, prípadne dočasnom uložení údajov pre

[3] JSON RFC 7159 https://tools.ietf.org/html/rfc7159

model. Prevod medzi modelom a modelomDto nám zabezpečuje spomínaná služba v obrázku 4.3 označená ako Služby. Napríklad služba `MenuServices` slúži na konvertovanie modelu `MenuItemDto` na model `MenuItem`.



Obr. 4.3: Implementácia Mobilní Objednávání Kredit 2.

4.3.1 Prism registrácia

Nakoľko bol používaný framework Prism, bolo taktiež potrebné pred inicializáciou registrovať jednotlivé služby pomocou návrhového vzor unikát (známejší pod anglickým pojmom singleton). Táto registrácia sa vykonávala v preťaženej metóde `RegisterTypes`, ktorá pochádza z framework-u Prism.

```
protected override void RegisterTypes(IContainerRegistry containerRegistry)
{
    containerRegistry.RegisterSingleton<IOrderingProxy,
    containerRegistry.RegisterSingleton<ICatchException, CatchException>();
    containerRegistry.RegisterSingleton<MenuService>();
    containerRegistry.RegisterSingleton<OrderingService>();

    containerRegistry.RegisterForNavigation<MainPage>();
    containerRegistry.RegisterForNavigation<MenuInfoPage>();
    containerRegistry.RegisterForNavigation<MenuPage,>();
}
```

Výpis 4.6: Registrácia služby vo frameworku Prism.

Ukážka 4.6 obsahuje registráciu stránky, ktorá sa bude užívateľovi zobrazovať. V prípade MVVM modelu je možné k jednotlivým stránkam (View) pridať ViewModel znázornený v kóde 4.7. Ukážka 4.6 nepriradzuje ViewModel k View z dôvodu, že v jednotlivých XAML súboroch na zobrazovanie užívateľského rozhrania ju priradí automaticky, nakoľko sa v koreni nastavil parameter zobrazený v kóde 4.8. Tento parameter zabezpečuje, že program sám pri preklade priradí k view patričný ViewModel. ViewModel musí obsahovať rovnaký názov ako View a za názvom musí byť priradený reťazec. ViewModel takto rozpozná program ktorý, ViewModel patrí k danému View alebo môže programátor ViewModel pomenovať rozličným spôsobom ale pri registrácií musí ViewModel zadefinovať k patričnému View, ako to je v ukážke 4.7

```
containerRegistry.RegisterForNavigation<MenuPage, MenuPageViewModel>();
```

Výpis 4.7: Manuálne priradenie ViewModelu pre View počas registrácie.

```
prism:ViewModelLocator.AutowireViewModel="True"
```

Výpis 4.8: Automatické priradenie ViewModelu pre View.

4.3.2 Navigačný panel

Slúži na prepínanie medzi jednotlivými view panelmi. Pre zobrazenie navigačných panelov je používaná trieda `NavigationService`, ktorú je možné použiť napríklad na zobrazenie panelu jedálneho lístka.

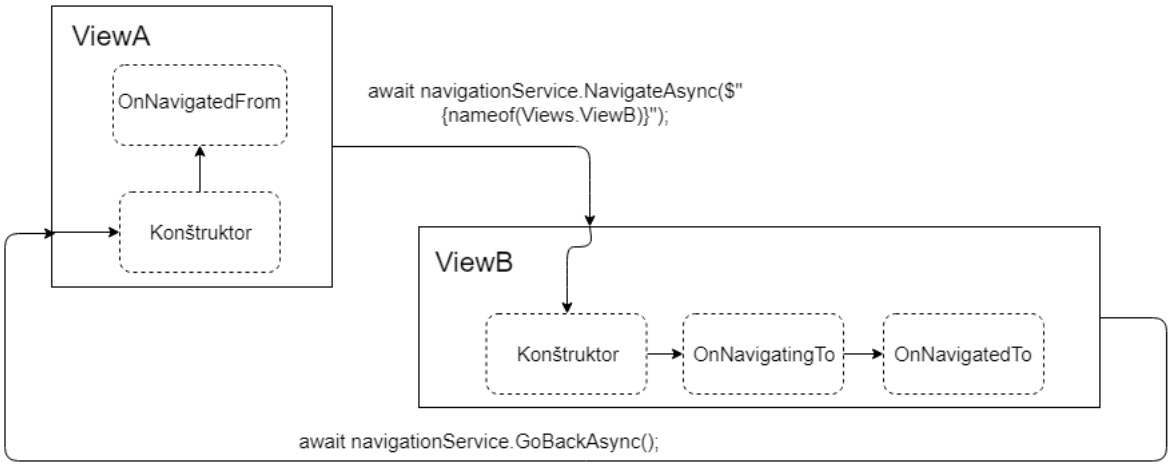
```
NavigationService.NavigateAsync(\$"{nameof(Views.MenuPage)}", param);
```

Výpis 4.9: Navigácie vo framework-u Prism.

Prvý parameter je možné zadávať ako reťazec k požadovanému iew panelu, čo umožňuje ľahšie uloženie a v prípade ukončenia aplikácie opätovné načítanie posledného zobrazeného panelu. `NavigationService` umožňuje predávať jednotlivým view panelom parametre, a to pomocou `NavigationParameters`.

```
NavigationParameters parameters = new NavigationParameters();
parameters.Add("MenuItem", selectedItemMenu);
```

Výpis 4.10: Pridanie parametru do navigácie v Prism.



Obr. 4.4: Diagram navigácie medzi dvoma stránkami vo framework-u Prism.

Rozdiel medzi `OnNavigatingTo` a `OnNavigatedTo` je ten, že metóda `OnNavigatingTo` sa volá pred vytvorením grafického rozhrania, zatiaľ čo `OnNavigatedTo` sa volá až keď sa grafické rozhranie vytvorí.

4.3.3 Špecifické vlastnosti pre platformu

Niektoré časti programu nebolo možné naprogramovať v zdielanom projekte PCL, nakoľko boli špecifické pre každú platformu samostatne. Špecifické funkcionality sa programujú pomocou vytvoreného rozhrania v PCL projekte, kde sa definujú metódy, ktoré je potrebné v platforme naprogramovať. Príkladom môže byť ukážka s pridaním záznamu do kalendára.

```
public interface ISpecApplication
{
    void AddToCalendar(string title, string location, DateTime datum);
}
```

Výpis 4.11: Rozhranie pre špecifickú funkcionalitu platforiem.

Keď je rozhranie definované, stačí už len vytvoriť triedu pomocou vytvoreného rozhrania v PCL ku každej platforme, v ktorej chceme danú špecifikáciu implementovať. Pre iOS platformu je trieda zobrazená nasledovne a pre Android sa bude líšiť odlišnou implementáciou funkcie `AddToCalendar`. Volanie takejto špecifickej funkcie, zobrazenej v časti kódu 4.13, sa vykonáva pomocou `DependencyService`.

```
public class SpecApplication : ISpecApplication
{
    public void AddToCalendar(string title, string location, DateTime
        datum)
    {
        EKEventStore eventStore = (EKEventStore)SpecContext;
        EKEvent newEvent = EKEvent.FromStore(eventStore);
        newEvent.AddAlarm(EKAlarm.FromDate(DateTimeToNSDate(datum.AddMinutes(10))));
        newEvent.StartDate = DateTimeToNSDate(datum);
        newEvent.EndDate = DateTimeToNSDate(datum);
```

```
        newEvent.Title = title;
        newEvent.Notes = location;
        newEvent.Calendar = eventStore.DefaultCalendarForNewEvents;

        NSError e;
        eventStore.SaveEvent(newEvent, EKSpan.ThisEvent, out e);
        if(e == null)return;
        UIAlertView alert = new UIAlertView()
        {
            Title = title,
            Message = e.LocalizedDescription
        };
        alert.AddButton("OK");
        alert.Show();
    }

    private NSDate DateTimeToNSDate(DateTime date)
    {
        if (date.Kind == DateTimeKind.Unspecified)
            date = DateTime.SpecifyKind(date, DateTimeKind.Local);
        return (NSDate)date;
    }
}
```

Výpis 4.12: Implamentácia ISpecApplication pre platformu Android.

```
Xamarin.Forms.DependencyService.Get<ISpecApplication>()
.AddToCalendar(OrderItem.MealAltDescription, OrderItem.CanteenDescription,
    OrderItem.Date.Date);
```

Výpis 4.13: Volanie funkcie špecifickej na platforme.

Kapitola 5

Štatistiky

Táto kapitola sa venuje dvom druhom testov. Prvý druh testu bude zahŕňať celkový výkon pôvodnej aplikácie a aplikácie napísanej v Xamarin.Forms. Druhý test bude zameraný na vybrané úlohy. V tomto prípade bola zvolená sieťová komunikácia a zaťaženie vykresľovaním komponentov.

Všetky tieto testy sú vykonávané na platforme Android. Výber platformy bol podmienený tým, že Android Studio poskytuje plnohodnotný profiler, ktorým bolo možné zmerať požadovaný výkon jednotlivých aplikácii. Tento nástroj je možné stiahnuť zadarmo[1].

Pre testovanie bol zvolený emulátor, ktorý je súčasťou Android studia. Emulátor simuluje mobilný telefón Pixel XL a beží na API verzii 25 (Android 7.1.1). Android 7.1.1 sa v dnešnej dobe dáva do väčšiny mobilných telefónov, ktoré obsahujú spomínaný operačný systém Android.

5.1 MobilKredit Android vs. MobilKredit 2 Xamarin.Forms

Testy pre MobilKredit a MobilKredit 2 boli zvolené tak, aby sa dali rovnocenne otestovať. Vyberali sa podľa nasledujúcich pravidiel:

- Nulová interakcia užívateľa.
- Procesy medzi dvoma akciami (udalosťami) musia byť presne a jasne merateľné.

Všetky hodnoty v tejto sekcii sú uvedené v časovom formáte (mm:ss,000).

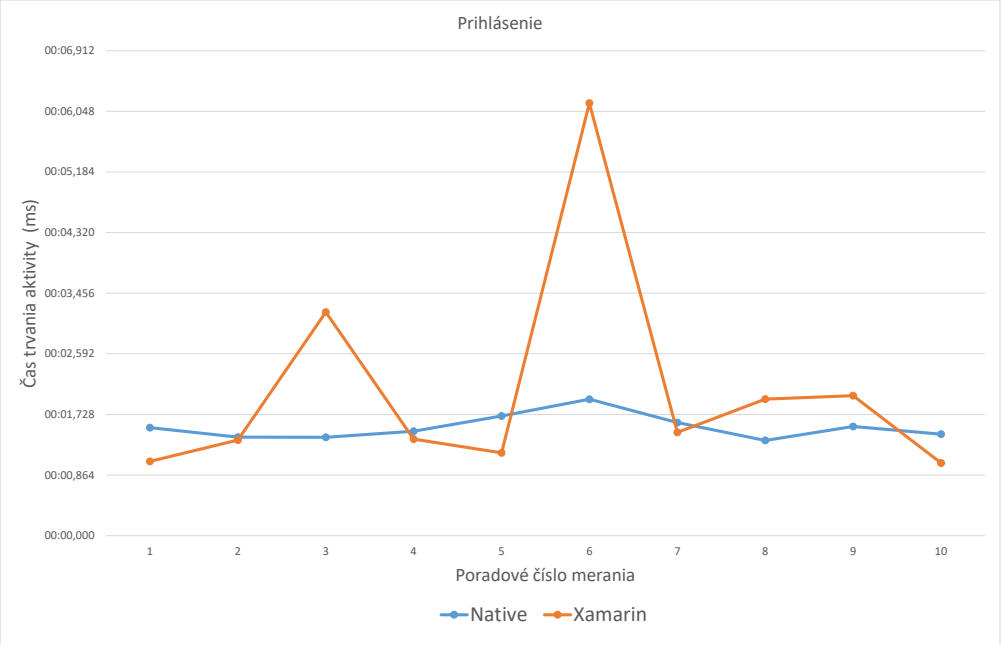
5.1.1 Prihlasovanie užívateľa

Prihlasovanie užívateľa prebiehalo podľa vyššie spomínaných pravidiel. Do políčok s menom a heslom sa zadali požadované potrebné údaje, spustil sa profiler na meranie záťaže CPU pre daný proces, následne sa kliklo na tlačidlo prihlásiť. Keď sa užívateľ prihlásil, profiler sa zastavil. Ako prvé bolo potrebné detekovať stlačenie tlačidla v profileri, ktorý interakciu zaznamenával. Kliknutie bolo v profileri nájdené ako udalosť s názvom onClick. Z udalosti onClick bol zapísaný začiatok vykonávania tejto udalosti. V ďalšom kroku bola nájdená udalosť pre vytvorenie okna po prihlásení, ktorá sa nachádzala pod udalosťou pomenovanou onFrame. Z udalosti onFrame bol zapísaný konečný čas udalosti a tieto časy

[1] Android Studio https://developer.android.com/studio/index.html

boli následne odčítané a zapísané do tabuľky. Takto bolo vykonaných 10 opakovaní pre každú aplikáciu.

Časy prihlasovania v Xamarin.Forms sú skoro identické ako pre pôvodnú aplikáciu napísanú v Jave znázornené na grafe 5.1. Na krivke reprezentujúcej Xamarin.Forms je možné vidieť dve anomálie počas prihlasovania. V týchto dvoch bodoch bolo pozorované, že aplikácia vykreslila užívateľské grafické rozhranie dvakrát za sebou. To je spôsobené tým, že vývojári Xamarinu nemajú optimalizované volania pre komponent TabbedPanel v rámci vybraných platforiem táto chyba sa prejavila prvý krát na verzii 2.3.0 a stále sa nachádza na verzii 2.5.0 v ktorej je naša aplikácia naprogramovaná. Na githube Xamarin.Froms môžete o tejto chybe nájsť viacej informácií [2]. Vo verzii Xamarin.Forms 3.0 by mala byť táto verzia opravená.



Obr. 5.1: Prihlasovanie aplikácií.

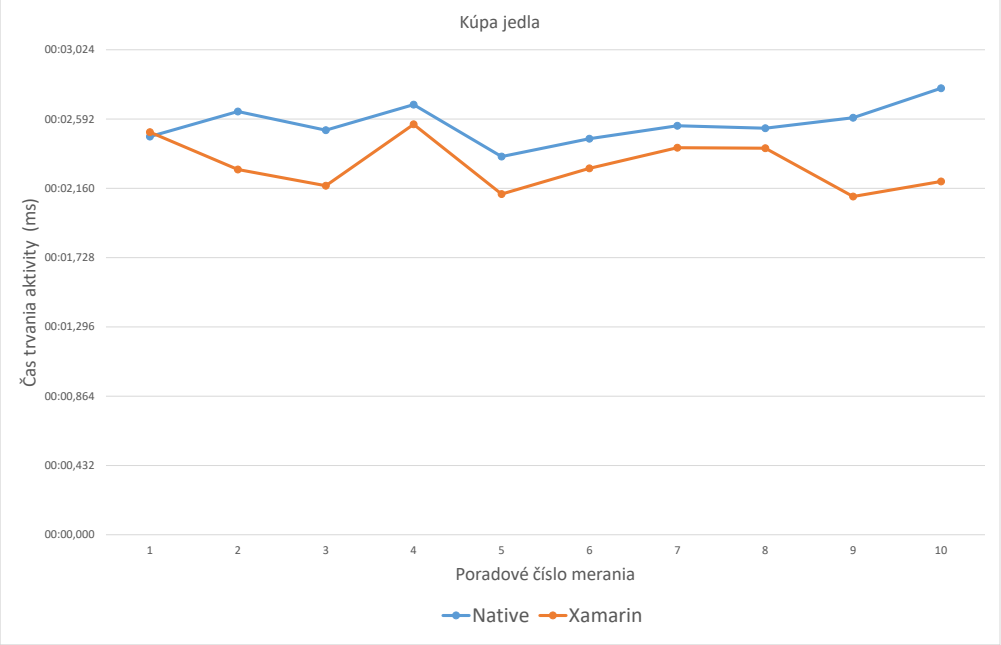
[2] Xamarin.Forms github https://github.com/xamarin/Xamarin.Forms/issues/1438

Android			Xamarin.Forms		
Začiatok onClick	Koniec doFrame	Rozdiel doFrame - onClick	Začiatok onClick	Koniec doFrame	Rozdiel doFrame - onClick
08:52,709	08:54,248	00:01,539	00:24,886	00:25,945	00:01,059
13:15,596	13:17,000	00:01,404	04:25,749	04:27,112	00:01,363
12:55,445	12:56,848	00:01,403	05:22,762	05:25,947	00:03,185
14:03,239	14:04,728	00:01,489	06:49,501	06:50,878	00:01,377
14:47,630	14:49,336	00:01,706	08:29,984	08:31,164	00:01,180
00:04,572	00:06,517	00:01,945	13:37,968	13:44,132	00:06,164
01:22,550	01:24,162	00:01,612	14:48,266	14:49,739	00:01,473
02:08,932	02:10,289	00:01,357	15:45,718	15:47,664	00:01,946
03:04,355	03:05,911	00:01,556	16:38,386	16:40,381	00:01,995
03:42,868	03:44,314	00:01,446	17:26,352	17:27,389	00:01,037
Priemer			Priemer		
0,017659602	0,017677492	00:01,546	0,007210153	0,007234203	00:02,078

Tabuľka 5.1: Tabuľka meraní pre prihlásenie.

5.1.2 Objednávanie jedla

Objednávanie jedla prebiehalo obdobným spôsobom ako pri prihlasovaní. Odchytil sa začiatok udalosti onClick pomocou profileru a následne sa odchytil koniec udalosti onFrame. Na grafe 5.2 je možné vidieť, že aplikácia MobilKredit 2 je nepatrne rýchlejšia ako natívna aplikácia. Pri pôvodnej aplikácii napísanej v Jave trvalo aplikácii priemerne spracovať objednávku 2,562 sekundy, zatiaľ čo v aplikácii napísanej Xamarin.Forms 2,307 sekundy. Z hľadiska užívateľa je tento rozdiel nepozorovateľný, nakoľko ide o rozdiel 256 milisekúnd.



Obr. 5.2: Objednávanie jedla v aplikácii.

Android			Xamarin.Forms		
Začiatok onClick	Koniec doFrame	Rozdiel doFrame - onClick	Začiatok onClick	Koniec doFrame	Rozdiel doFrame - onClick
15:17,637	15:20,120	00:02,483	01:04,352	01:06,862	00:02,510
18:06,385	18:09,024	00:02,639	14:08,901	14:11,178	00:02,277
22:56,273	22:58,796	00:02,523	15:22,004	15:24,180	00:02,176
24:38,198	24:40,880	00:02,682	17:44,901	17:47,460	00:02,559
25:19,497	25:21,855	00:02,358	18:48,132	18:50,256	00:02,124
29:45,911	29:48,380	00:02,469	19:38,121	19:40,406	00:02,285
31:23,036	31:25,586	00:02,550	20:29,587	20:32,000	00:02,413
36:36,249	36:38,784	00:02,535	25:01,200	25:03,610	00:02,410
37:16,408	37:19,008	00:02,600	25:48,031	25:50,140	00:02,109
39:54,279	39:57,063	00:02,784	26:44,399	26:46,602	00:02,203
Priemer			Priemer		
0,019529946	0,019559602	00:02,562	0,012835218	0,012861914	00:02,307

Tabuľka 5.2: Tabuľka meraní pre prihlásenie.

Z grafu 5.2 môžeme vidieť, že Xamarin.Forms je pri objednávaní jedla rýchlejší ako natívna aplikácia je to z dôvodu, že programovací jazyk C# dokáže rýchlejšie pracovať dáta zo servera ako programovací jazyk Java. Bližšie si o tomto spracovaní povieme v kapitole 5.2.2, kde sa venujeme spracovaniu požiadavku na server.

5.2 Android vs. Xamarin.Forms

Táto kapitola je zameraná na dva testy a ich vplyv na mobilné zariadenie. Pre tieto testy boli vytvorené dve aplikácie pod názvom TestPerformance. Jedna aplikácie je naprogramovaná pod technológiou Java pre Android a druhá technológiou Xamarin.Forms. Testy sú uvedené nasledovne:

1. Vykresľovanie komponentov v komponente ListView - záťaž na pamäť.
2. Cyklické posielanie požiadaviek na server, pričom server vracia dáta vo formáte JSON - sieťová komunikácia.

5.2.1 Záťaž aplikácií na pamäť

Na odmeranie záťaže bol použitý test na generovanie komponentu do ListView. ListView je komponent, v ktorom je možné zobrazovať údaje cyklicky, pod zadaným rozvrhnutím komponentov, alebo taktiež nazývaný template. Zaťaženie pamäte bolo merané pomocou profileru v časti memory, a to nasledovným spôsobom:

1. spustenie aplikácie Test
2. ustálenie aplikácie a zapísanie hodnôt do pamäte, ktorú zobrazoval profiler
3. otvorenie testu na generovanie komponentov v ListView a opakovanie bodu č. 2
4. vygenerovanie určitého počtu komponentov a zapísanie hodnoty do pamäte
5. zmena v generovanom počte komponentov a návrat k bodu č. 4

Po vyššie vykonanom postupe bola získaná nasledujúca tabuľka 5.3. Každý vygenerovaný komponent v ListView obsahoval údaje o poradí, v ktorom bol vytvorený a náhodne vygenerované číslo nesúce dátovú hodnotu.

Počet komponentov	Android	Xamarin.Forms
0	11,0 MB	35,4 MB
5	11,3 MB	38,8 MB
10	11,3 MB	37,1 MB
20	12,2 MB	38,0 MB
50	12,6 MB	38,1 MB
100	14,1 MB	38,3 MB
200	17,9 MB	38,2 MB
500	18,9 MB	38,4 MB
1000	17,3 MB	38,4 MB
15000	18,2 MB	39,2 MB
45000	20,7 MB	40,1 MB

Tabuľka 5.3: Meranie záťaže pamäti na mobilnom zariadení.

V tabuľke 5.2 môžeme vidieť, že technológia Xamarin.Forms je v priemere 3-násobne náročnejšia na pamäť ako pôvodná aplikácia v Androide. Tento fakt môže spôsobovať pri zariadeniach s nižšou pamäťou zhoršenie vykonávania paralelných operácií (taktiež známe pod názvom multitasking).

5.2.2 Cyklické odosielanie požiadavky na server

Ďalší test bol zameraný na získavanie dát zo servera pomocou požiadavky (requestu) na získanie dát v JSON formáte. Každý cyklus mal nasledujúce parametre:

1. počet požiadavok na server
2. počet opakovaní celého procesu

Ako server bol použitý voľne dostupný Post Test Server V2[3]. Na tomto serveri sme si vytvorili požiadavku typu GET[4], ktorá nám vráti predom stanovené dáta poskytované serverom.

Obe aplikácie Xamarin.Forms aj Android boli naprogramované tak, aby poslali požiadavky na server a spracovali prijaté dáta do textovej podoby. Všetky výsledky sú v milisekundách a tieto časy sú strojové časy.

```
for (int x = 0; x < repeat; x++){
    Stopwatch timer = Stopwatch.StartNew();
    for (int i = 0; i < SendCount; i++){
        try{
            HttpClient hTTPClient = new HttpClient();
            hTTPClient.Timeout = new TimeSpan(0,0,0,0,READ_TIMEOUT);
            var response = await hTTPClient.GetAsync(new Uri(url));
            if (response.IsSuccessStatusCode){
                string content = await response.Content.ReadAsStringAsync();
                SucessStatus++;
            }
        }
        catch { FaultStatus++; }
        finally{
            CountStatus++;
        }
    }
    timer.Stop();
    dataAverage.Add(timer);
}
```

Výpis 5.1: Požiadavky na server v Xamarin.Forms.

```
for(int x = 0; x < repeat; x++) {

final long start = System.currentTimeMillis();
for (int i = 0; i < numberOfRepeat; i++) {
        try {
            URL myUrl = new URL(stringUrl);
            HttpURLConnection connection = (HttpURLConnection)
                    myUrl.openConnection();
            connection.setRequestMethod(REQUEST_METHOD);
```

[3]Post Test Server V2 http://ptsv2.com
[4]Požiadavka typu GET (získanie dát) http://ptsv2.com/t/get/d/1/json

```
            connection.setReadTimeout(READ_TIMEOUT);
            connection.setConnectTimeout(CONNECTION_TIMEOUT);
            connection.connect();

            BufferedReader br = new BufferedReader(new
                InputStreamReader(myUrl.openStream()));
            StringBuilder sb = new StringBuilder();
            String line;
            while ((line = br.readLine()) != null) {
                sb.append(line + "\n");
            }
            br.close();
            String jsonString = sb.toString();
            connection.getResponseCode();
            this.publishProgress(connection.getResponseCode());
        } catch (IOException e) {
            e.printStackTrace();
            this.publishProgress(500);
        }
    }
    final long end = System.currentTimeMillis();
    final long result = ((end - start));
    dataToFile.add(result);
}
```

Výpis 5.2: Požiadavky na server v Android pomocou jazyka Java.

V prípade tohto testu prebehli 3 rôzne situácie, každá situácia pri spustení programu prebehla 10-krát. Tento postup sa opakoval pre lepšie výsledky 7-krát. Test sa vykonával na 3 rôzne spôsoby.

1. Na server bola odoslaná 1 požiadavka.
2. Na server sa odoslali za sebou 2 požiadavky.
3. Na server sa odoslalo za sebou 5 požiadavok.

Oba procesy bežali v aplikáciach asynchrónne, aby sa program pri zapisovaní informácií do užívateľského rozhrania nezasekával.

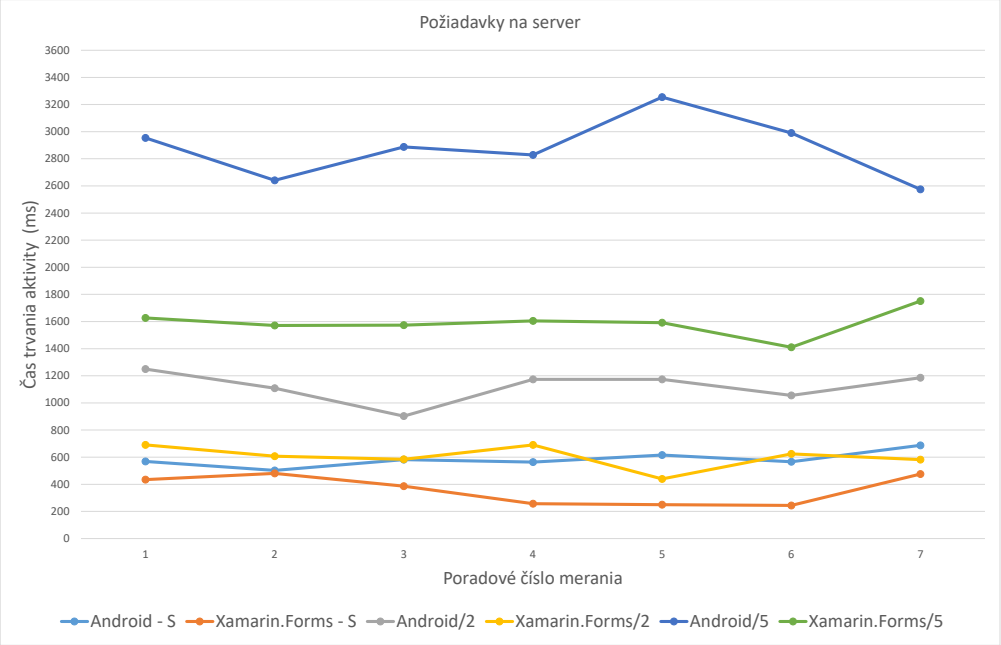
Platforma/Počet požiadaviek	1.	2.	3.	4.	5.	6.	7.
Android/1 [ms]	569,2	502,5	582	563,5	615,5	566,7	687,5
Xamarin.Forms/1 [ms]	434	480,5	386,2	257,6	249,8	244,3	475,2
Android/2 [ms]	1249,4	1108,7	902,7	1173,6	1173,6	1055,5	1185,3
Xamarin.Forms/2 [ms]	690,2	606,8	584,3	690,3	439,7	623,6	582
Android/5 [ms]	2953,4	2641,2	2888,1	2828,4	3254,7	2990,6	2574,8
Xamarin.Forms/5 [ms]	1626,6	1570,7	1573,1	1604,9	1591,8	1410,3	1751

Tabuľka 5.4: Strojový čas na odosielanie požiadaviek. Priemerné hodnoty tabuliek A.1, A.2 a A.3

Z predchádzajúcich zistení je možné posúdiť, že implementácia sieťového rozhrania je v kóde 5.1 Xamarin.Fomrs jednoduchšia ako v prípade implementácie z úkážky 5.2 pre Android. Môže sa zdať, že spomalenie Android aplikácie nastáva už pri implementácii sieťového rozhrania a požiadavky na server. Žiaľ, toto spomalenie na strane Android aplikácie vzniká pri spracovaní prichádzajúcich dát zo servera. Zatiaľ čo v prípade Xamarin.Forms požiadavku vráti priamo reťazec, pre Android je potrebné implementovať kód 5.3, ktorý dáta získa do požadovanej podoby.

```
BufferedReader br = new BufferedReader(new
    InputStreamReader(myUrl.openStream()));
StringBuilder sb = new StringBuilder();
String line;
while ((line = br.readLine()) != null) {
    sb.append(line + "\n");
}
br.close();
String jsonString = sb.toString();
```

Výpis 5.3: Načítavanie reťazca s požiadavkou na server.



Obr. 5.3: Graf zobrazujúci výsledky jednotlivých požiadaviek. Tabuľka 5.4.

Následujúci test je implementovaný iba poslaním požiadavky na server a skontrolovanie jej stavu. Medzi tabuľkami 5.5 a A.1 je možné pozorovať časový rozdiel, ktorý spomínaná časť kódu 5.3 spôsobuje.

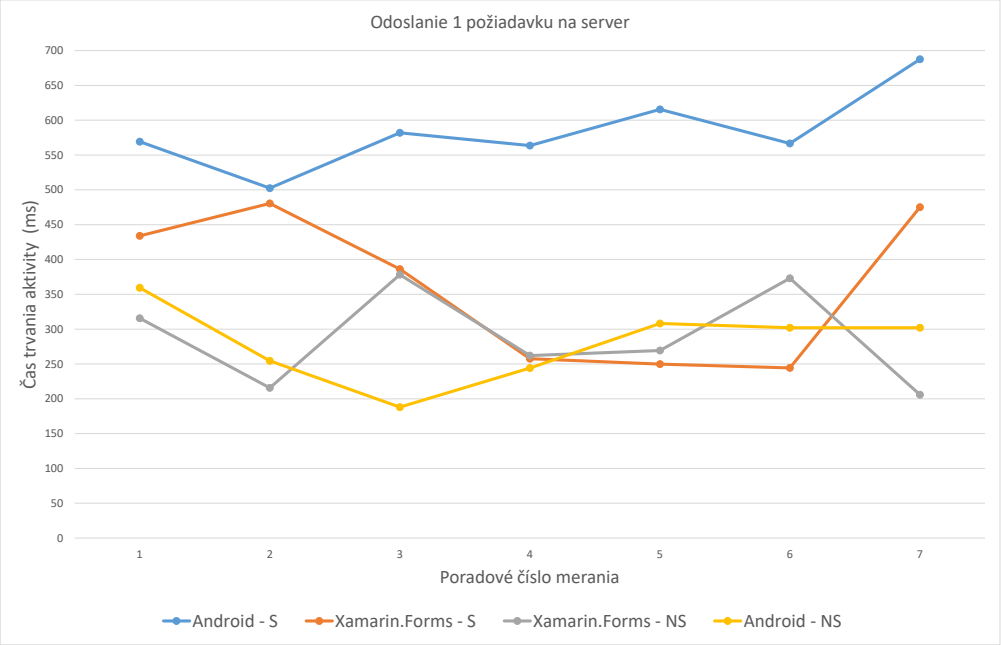
Android [ms]						
1.	2.	3.	4.	5.	6.	7.
267	240	188	267	182	225	781
177	191	234	185	290	247	188
180	185	181	181	179	738	179
182	737	187	174	183	177	738
177	178	177	181	226	184	172
741	183	183	179	183	183	178
183	183	182	734	730	732	188
735	179	180	176	180	180	177
202	291	178	190	191	179	190
750	177	190	175	737	176	230

Tabuľka 5.5: Strojový čas na odosielanie 1 požiadavky bez spracovania dát.

Pre rýchlejšie porovnanie boli všetky 3 merania spriemerované, z čoho bol následne vytvorený graf s jednotlivými krivkami. Tabuľka popisuje odosielanie jednej požiadavky na server.

Platforma	1.	2.	3.	4.	5.	6.	7.
Android - spracovanie dát [ms]	569,2	502,5	582	563,5	615,5	566,7	687,5
Android - nespracovanie dát [ms]	359,4	254,4	188	244,2	308,1	302,1	302,1
Xamarin.Forms - spracovanie dát [ms]	434	480,5	386,2	257,6	249,8	244,3	475,2
Xamarin.Forms - nespracovanie dát [ms]	315,6	215,6	378,2	262,1	269,4	373,1	205,8

Tabuľka 5.6: Priemer strojového času na odosielanie 1 požiadavky.



Obr. 5.4: Graf tabuľky 5.6.

Kapitola 6

Testovanie - chyby a ich riešenia

Hlavnou prioritou testovania je spoľahlivosť aplikácie a kompatibilita so zariadeniami iOS a Android, v druhom rade kompatibilita s Windows Phone. Ten sa po čase z testovania vyradil, nakoľko spoločnosť Microsoft prestala podporovať tento operačný systém. Počas implementácie boli použité na testovanie virtuálne zariadenia.

- iOS – virtuálne zariadenia na počítači mac mini. (napríklad iPhone 5, iPhone SE, iPhone 7 / 7plus a podobné). Neskôr sa použilo aj reálne zariadenie iPhone SE.
- Android – virtuálne zariadenia pre android nazývané AVD (Android virtual device). Reálne zariadenia Xiaomi redmi 4A, Nokia 8.
- Windows Phone – Nokia 730 Dual.

Na testovanie aplikácie neboli používané automatizované testy, a to z dôvodu, že aplikácia nie je rozsiahla, technológia Xamarin je ešte len v začiatkoch a automatizované testy by prácu na projekte znepríjemnili. Veľké množstvo chýb bolo často spôsobených práve samotnou technológiou. Zopár takýchto chýb je zahrnutých aj v tejto kapitole.

Prvé testovanie prebehlo na operačnom systéme Android, nakoľko je pôvodná aplikácia vyvinutá pre túto mobilnú platformu. Pri testoch bola objavená prvá zásadná chyba, ktorá však bola očakávaná. Aplikácie boli pomalšie najmä pri štarte vo všetkých spomínaných platformách. Tento fakt mohol užívateľov odradiť pri pohľade na biele pozadie a aplikáciu vypnúť aj napriek tomu, že sa aplikácia spúšťala. V dôsledku toho bola implementovaná úvodná obrazovka a táto implementácia musela prebehnúť samostatne pre každú platformu. Po načítaní bol beh aplikácie plynulejší do doby, kým sa nenačítal dlhší zoznam vybraných objednávok, prípadne zoznam jedálneho lístka do komponentu zvaného ListView. Problém bol vyriešený optimalizáciou komponentu o ktorý je popísaný v kapitole 6.1.

Ďalším problémom bolo aktualizovanie údajov po prechode medzi panelmi v komponente TabbedPanel. To spôsobovalo, že ak si užívateľ objednal jedlo v paneli Ponuka a prešiel následne do panelu Objednávky, tak sa objednávky v paneli neaktualizovali a objednané jedlo sa nezobrazovalo. V dôsledku toho bola vytvorená služba pre jednotlivé dáta v paneloch. V prípade, že si užívateľ objedná jedlo, táto služba sa zavolá a taktiež aktualizuje dátovú zložku v paneli Objednávky zo servera.

Aplikáciu je možné otestovať na firemnom servery anetedemo pod uživateľom kral a heslom x.

6.1 Optimalizácia ListView

Počas testovania bolo zistené, že niektoré komponenty potrebujú optimalizovať a niektoré balíčky nie je možné použiť so spojením s inými komponentami. NuGet balíček Iconize, ktorý bol spomenutý v návrhu aplikácie, nie je vhodne implementovaný na použitie v komponente ListView. Príčinou je pomalé vykresľovanie komponentov, čo spôsobuje neprirodzené načítavanie a oneskorené zobrazovanie jedálnych lístkov, histórie, objednávok.

Taktiež je potrebné sa vyhnúť v ListView parametru `HorizontalOption`, z ukážky 6.1, s nastavením `Fill`. Toto nastavenie spôsobuje, že pre každú jednu dátovú zložku ListView sa vypočítava veľkosť na šírku zobrazovacej plochy.

```
<ListView HorizontalOptions="Fill"> . . . </ListView>
```

Výpis 6.1: Nastavenie komponentu ListView na šírku rozlíšenia obrazovky.

Jedným zo spôsobom ako sa tejto problematike vyhnúť je navrhnúť `DateTemplate` v komponente ListView tak aby sa jednotlivé prvky v ListView samostatne prispôsobovali. Príkladom môže byť použitie komponentu Grid (Mriežka) v `DateTemplate` kde je možné nastaviť pomery jednotlivých stĺpcov voči šírké (`ColumnDefinition`) alebo výške (`RowDefinition`) zobrazovacieho panelu.

```
<DataTemplate>
    <ViewCell>
        <Grid VerticalOptions="Center" Padding="10">
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="5"/>
                <ColumnDefinition Width="AUTO"/>
                <ColumnDefinition Width="5"/>
                <ColumnDefinition Width="*"/>
                <ColumnDefinition Width="AUTO"/>
            </Grid.ColumnDefinitions>
            .
            .
            .
        </Grid>
    </ViewCell>
</DataTemplate>
```

Výpis 6.2: Načítavanie reťazca s požiadavkou na server.

Kapitola 7

Zverejňovanie aplikácie

Zverejňovanie aplikácie pre obchod Google a Apple nie je od bežných aplikácií veľmi odlišné. Prvým krokom je založenie si potrebných účtov. Založenie vývojárskeho účtu na oboch obchodoch je spoplatnené.

7.1 Android

Aplikácia pre Android sa zverejňuje na Google Play. Pred zverejnením je potrebné vo Visual Studiu vygenerovať balíček. Balík vygenerujeme tak, že klikneme pravým tlačidlom myši na projekt pre Android a vyberieme „Archive...“. Po vygenerovaní sú k dispozícii dve možnosti ako nahrať vytvorenú aplikáciu do obchodu:

1. Pomocou Ad Hoc je možné priradiť k balíku identifikátor (bežné aj pre vývoj natívnej aplikácie) a uložiť balíček .apk. V ďalšom kroku je nutné zložku otvoriť tam, kde sa súbor .apk nachádza. Ten sa nahrá cez webové rozhranie po prihlásení na Google Store.

2. Pomocou Google Play. Táto možnosť je o niečo jednoduchšia, ale je potrebné mať priradený vývojársky účet pre Google Play. Tak ako pre bod č. 1 pridáme identifikátor, následne vyberieme Google Play účet, na ktorý chceme aplikáciu nahrať, potom už len stačí vybrať typ zverejnenia (Alpha, Beta, Production).

Po nahratí aplikácie do Google Play je potrebné počkať na overenie a následné zverejnenie aplikácie. Tento proces trvá približne jednu hodinu. Aplikáciu Mobilní Objednávání Kredit 2, vyvíjanú v tejto práci, je možné stiahnuť na Google Play[1].

7.2 iOS

V prípade iOS je iba jediná možnosť zverejňovania aplikácie, a to pomocou balíčka .ipa. Pokiaľ je záujem o balíček .ipa, je potrebné ho zapnúť v nastaveniach projektu pre iOS pod kategóriou „iOS IPA Options“. Po preložení aplikácie sa súbor .ipa nachádza na zariadení MAC, kde sa aplikácia ukladá alebo sa nachádza priamo v binárnych súboroch

[1] Mobilní Objednávání Kredit 2 - Android
https://play.google.com/store/apps/details?id=anete.droid.mobilkredit&hl=cs

na Windowse. IPA balíček je potrebné nahrať pomocou aplikácie „Application Loader“ na zariadeniach MAC. Aplikáciu pre zariadenia iOS je možné stiahnuť z iTunes[2].

[2]Mobilní Objednávání Kredit 2 - iOS
https://itunes.apple.com/us/app/mobilkredit-2/id1308812320?l=sk&ls=1&mt=8

Kapitola 8

Záver

Táto práca sa zaoberá novou technológiou vytvorenia aplikácie pre viaceré mobilné platformy a tvorbou funkčného programu pod technológiou Xamarin. Aplikácia bola úspešne vytvorená a zverejnená pre užívateľov mobilných zariadení Android a iOS. Okrem toho bola porovnávaná platforma Android s platformou Android v technológii Xamarin.Forms. Táto platforma je podľa výsledkov bakalárskej práce vhodná na produkčný vývoj aplikácií pod podmienkou, že aplikácia nie je náročná a nezaoberá sa prílišnými špecifickými vlastnosťami na jednu z programovaných platforiem. Pre vývoj pod Xamarin.Froms je dobré si aplikáciu navrhnúť na obecnej úrovni, najmä v prípade zobrazovaných komponentov, nakoľko budú s užívateľom komunikovať. Ak sa užívateľ rozhodne pre konkrétnu aplikáciu, ktorá bude dôležitá pre neho, či rozvoj spoločnosti, odporúčam takúto aplikáciu programovať pod pôvodnou technológiou danej platformy. Najväčšia nevýhoda v Xamarin.Forms čelí problému príchodu noviniek na platformách, pričom v technológii Xamarin sa s vysokou pravdepodobnosťou objavia tieto novinky až o pár mesiacov neskôr.

Xamarin.Forms nie je jedinou technológiou, ktorou je možné programovať viac platformové aplikácie. V poslednej dobe sa rozšírili aj technológie React Native a Ionic 2, ktoré sa uchytili u firiem, aby znížili čas a náklady na vývoj mobilných aplikácií [4]. Nedá sa presne určiť, ktorá z týchto technológií je optimálna. Taktiež záleží na postoji, skúsenostiach a preferenciách jednotlivých programátorov k danej technológii. Xamarin.Forms je lepšou voľbou v prípade, ak sa firma chce zamerať na Microsoft technológie, nakoľko v Xamarin.Forms dokáže programátor vytvoriť aplikáciu pre Android, iOS, Windows Phone, prípadne programy pre desktopové počítače bežiace na operačnom systéme Windows. Pri väčšej znalosti technológie je možné aplikácie optimalizovať aj na zariadenia IoT (Internet of Thinks), nakoľko Microsoft ponúka operačný systém Windows IoT.

Literatúra

[1] Brian Lagunas: *Prism*. [Online; navštíveno 22.04.2018].
URL https://github.com/PrismLibrary/Prism

[2] Deitel, P. D. H. M.: *Android^{TM} How to Program, Third Edition*. Pearson, 2016, ISBN 978-0-13-448918-6.

[3] Hermes, D.: *Xamarin Mobile Application Development: Cross-Platform C# and Xamarin.Forms Fundamentals 1st ed. Edition*. Apress, 2015, ISBN 978-1484202159.

[4] Kate Fastrich: *Introduction to Portable Class Libraries*. 2018, [Online; navštíveno 10.05.2018].
URL https://belitsoft.com/react-native-development/react-native-vs-xamarin-vs-ionic

[5] Microsoft: *Introduction to Portable Class Libraries*. [Online; navštíveno 25.04.2018].
URL https://docs.microsoft.com/en-us/xamarin/cross-platform/app-fundamentals/pcl?tabs=vsmac

[6] Mono Project: *Mono*. [Online; navštíveno 09.04.2018].
URL https://www.mono-project.com/docs/about-mono/

[7] Petzold, C.: *Creating Mobile Apps with Xamarin.Forms*. Microsoft Press A Division of Microsoft Corporation One Microsoft Way Redmond, Washington 98052-6399, 2014, ISBN 978-1-5093-0297-0.

[8] Siddiqi, R. V. M. S.: *MVVM Survival Guide for Enterprise Architectures in Silverlight and WPF*. Packt Publishing, 2012, ISBN 978-1-84968-343-2.

Prílohy

Príloha A

Tabulky výsledkov pre štatistiky

Android [ms]							Xamarin.Forms [ms]						
1.	2.	3.	4.	5.	6.	7.	1.	2.	3.	4.	5.	6.	7.
421	950	460	374	953	365	935	1527	232	193	205	186	207	226
387	331	316	331	901	334	316	190	209	184	788	194	190	741
873	365	887	322	318	877	323	193	745	174	199	186	178	198
328	884	324	340	883	321	949	177	741	257	188	179	174	184
320	332	358	320	910	882	399	184	760	180	181	184	186	819
341	321	958	877	321	330	888	192	191	983	196	228	184	180
1441	884	324	901	351	923	317	750	191	770	187	775	180	737
349	325	322	903	881	879	1507	191	752	186	246	190	184	181
332	313	428	948	315	345	918	187	754	180	199	189	776	739
900	320	1443	319	322	411	323	749	230	755	187	187	184	747

Tabuľka A.1: Strojový čas na odosielanie 1 požiadavky.

Android [ms]							Xamarin.Forms [ms]						
1.	2.	3.	4.	5.	6.	7.	1.	2.	3.	4.	5.	6.	7.
1879	858	717	1829	1829	731	695	1223	962	402	439	418	967	588
1230	802	1236	1210	1210	652	1823	946	924	959	400	364	367	369
699	670	656	664	664	1261	1289	420	376	986	397	372	503	366
1785	1809	651	1244	1244	719	1231	376	409	373	1506	944	964	1146
755	656	1273	1237	1237	666	683	380	382	375	371	410	1012	359
1320	670	688	667	667	747	1205	371	372	1249	923	369	932	922
1240	1223	1201	1216	1216	1845	680	926	375	385	378	363	381	387
1248	1370	1220	664	664	690	665	922	935	369	367	368	380	366
1232	1798	707	1780	1780	2351	1777	957	401	372	1158	398	361	931
1106	1231	678	1225	1225	893	1805	381	932	373	964	391	369	386

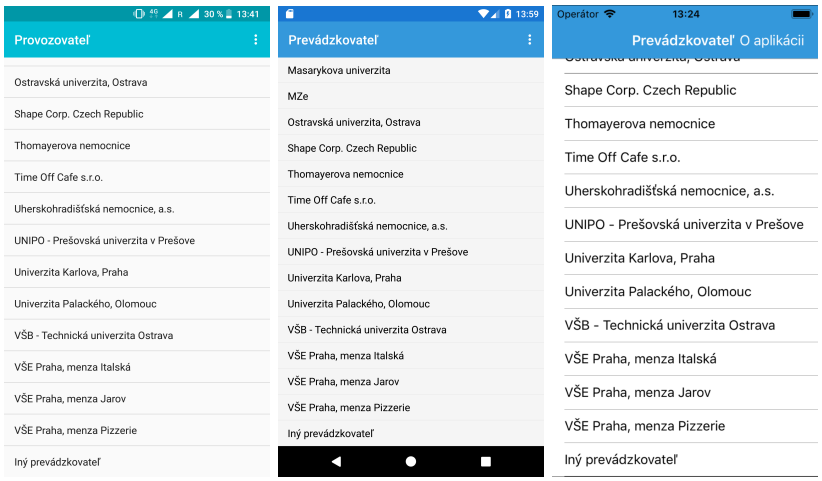
Tabuľka A.2: Strojový čas na odosielanie 2 požiadaviek.

Android [ms]							Xamarin.Forms [ms]						
1.	2.	3.	4.	5.	6.	7.	1.	2.	3.	4.	5.	6.	7.
2758	2240	2561	1755	2777	2742	3451	1 791	1574	994	1487	921	903	940
2763	3334	2245	3311	2210	3433	2990	1 565	978	1937	2873	1517	1532	2075
2767	2219	3465	2743	3857	2311	1755	1 529	932	1512	1522	1497	956	950
2756	1634	3348	2922	3317	2761	2274	1 515	1582	2054	967	2047	2088	1515
2802	2770	2641	3347	3957	2397	2270	2 640	2060	1523	1016	2679	1628	3240
4334	2758	2877	2815	4564	2853	3342	990	1801	1499	1517	1028	1224	2080
2187	2849	2854	3047	2760	2782	2820	1 507	1617	1575	1521	963	1751	2108
2843	2740	3378	2200	2434	3925	2803	1500	2173	1481	2065	1608	951	1519
2211	2430	2779	3320	3906	2211	2411	2301	1483	2200	1023	1603	2061	995
4113	3438	2733	2824	2765	4491	1632	928	1507	956	2058	2055	1009	2088

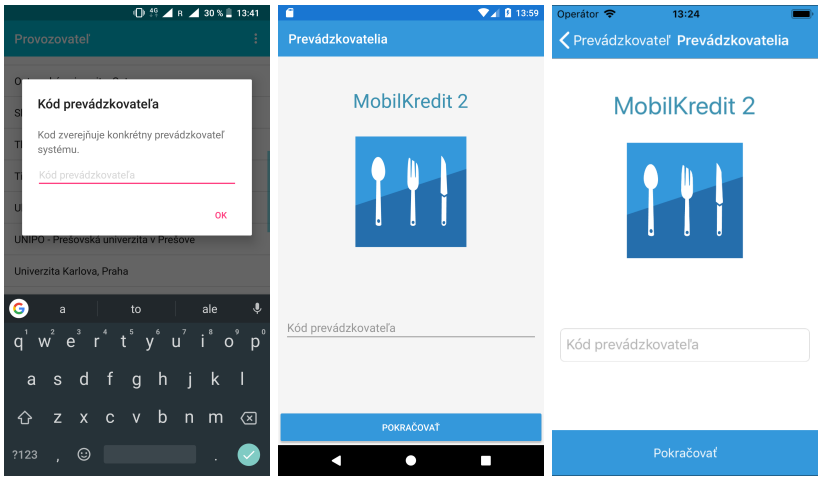
Tabuľka A.3: Strojový čas na odosielanie 5 požiadaviek.

Príloha B

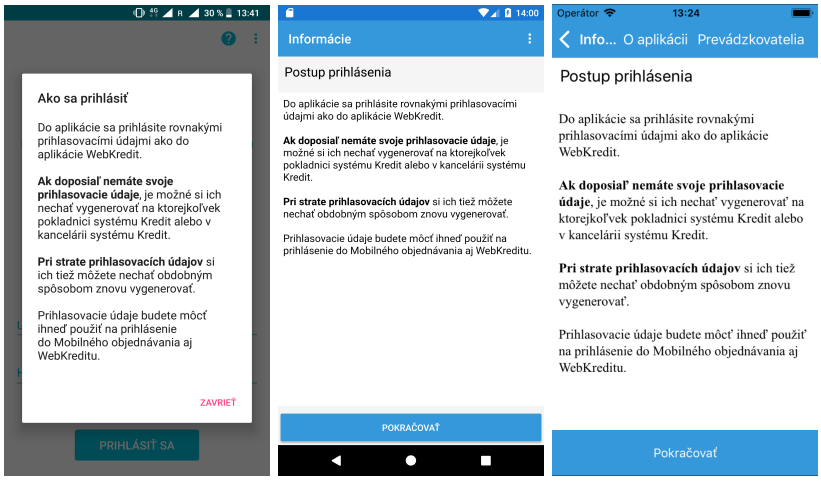
Obrázky Mobilní Objednávání Kredit 2



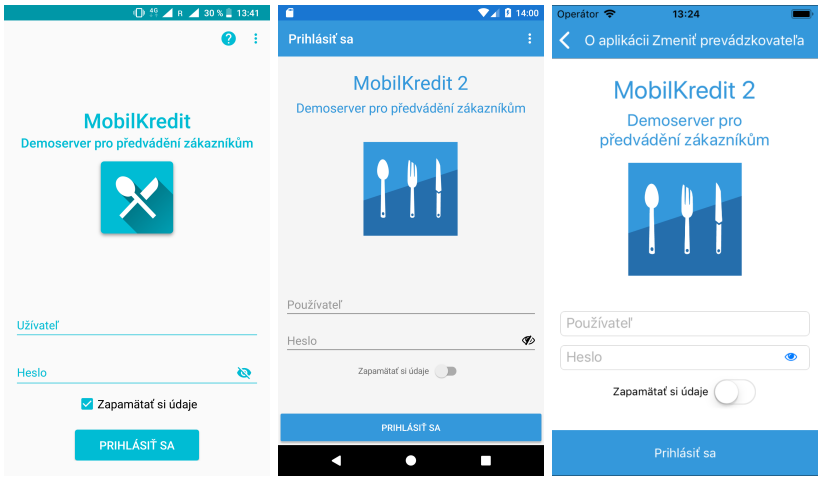
Obr. B.1: Výber poskytovateľa.



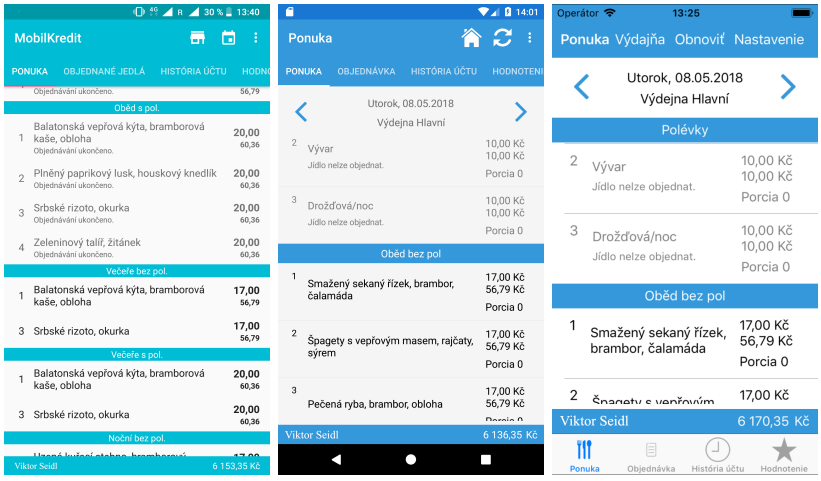
Obr. B.2: Výber poskytovateľa, ktorý nie je v zozname.



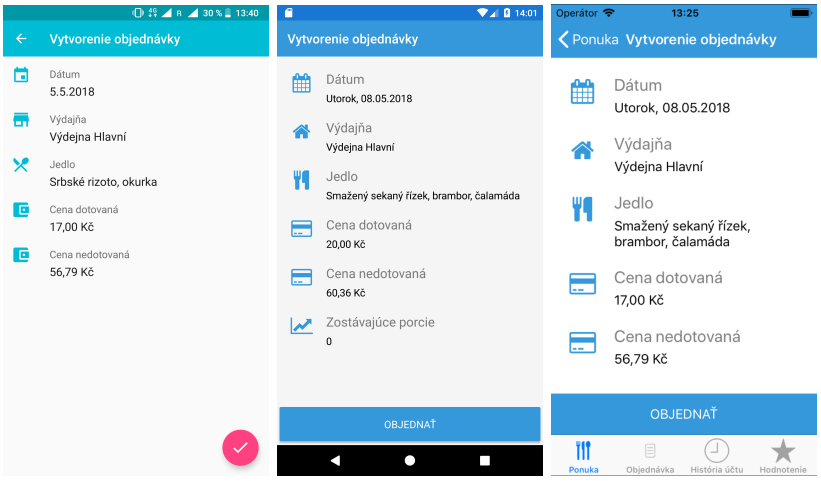
Obr. B.3: Pokyny pre úspešné prihlásenie u vybraného poskytovateľa.



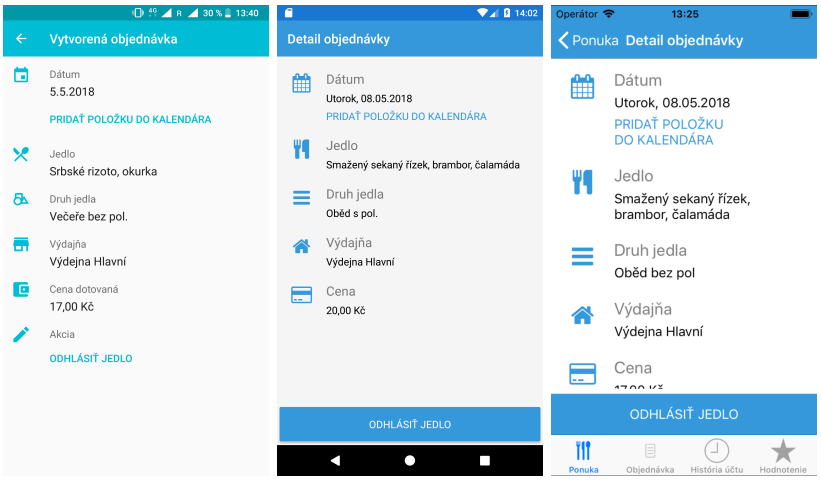
Obr. B.4: Prihlasovanie.



Obr. B.5: Výber jedla z ponuky.



Obr. B.6: Vytvorenie objednávky.



Obr. B.7: Vytvorená objednávka.

Príloha C

Obsah pamäťového média

Priložené pamäťové médium obsahuje:

- Zdrojové súbory textu bakalárskej práce.
- Bakalársku prácu vo formáte .pdf.
- Zdrojové kódy aplikácie Mobilní Objednávání Kredit 2.
- Zdrojové kódy aplikácie TestPerformance pre Android a Xamarin.