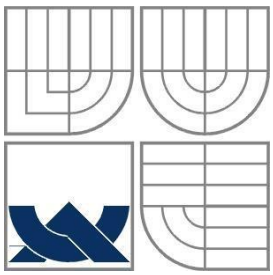
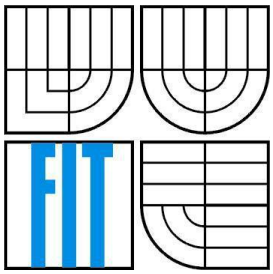


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

TERMINÁL PRO OBJEDNÁVÁNÍ JÍDEL

TERMINAL FOR ORDERING MEALS

BAKALÁŘSKÁ PRÁCE
BACHELOR‘S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN KRŠÁK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. ZBYNĚK KŘIVKA, Ph.D.

BRNO 2016

Abstrakt

Cieľom bakalárskej práce bolo vytvoriť aplikáciu pre objednávanie jedál v menze. Teoretická časť popisuje technológie WPF a UWP, návrhový vzor MVVM a PRISM Framework, ktoré sú v práci využité. Pre túto prácu bola podrobnejšie naštudovaná a využitá technológia UWP. Práca zahŕňa taktiež implementačnú časť, návrh aplikácie, analýzu, zber požiadaviek, postup tvorby aplikácie a testovanie.

Abstract

The goal was to create an application for ordering food in the canteen. The theoretical part describes UWP and WPF technology, MVVM design pattern and PRISM framework, which are used in the work. For this work was detailed preparation and use of technology UWP. The work also includes implementation part, application design, analysis, process of creating the application and testing.

Klíčová slova

UWP, WPF, PRISM, MVVM, .NET Framework, terminál, objednávanie jedál, REST

Keywords

UWP , WPF , PRISM , MVVM , .NET Framework, terminal, ordering meals, REST

Citace

Martin Kršák: Terminál pre objednávanie jedál, bakalářská práce, Brno, FIT VUT v Brně, 2016

Terminál pre objednávanie jedál

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Zbyňka Křivku, Ph.D. Další informace mi poskytl odborný konzultant firmy ANETE, s.r.o., pán Ing. Tomáš Juríček. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

19. května 2016 ……
Martin Kršák

Poděkování

Touto cestou by som sa chcel poďakovať svojmu vedúcemu bakalárskej práce pánovi Ing. Zbyňkovi Křivkovi, Ph.D. za odborné vedenie a cenné rady pri tvorbe tejto práce. Taktiež by som chcel poďakovať odbornému konzultantovi Ing. Tomášovi Juríčkovi.

Obsah

1. Úvod ... 1
2. Súčasný stav ... 2
   2.1 Microsoft Continuum ... 3
   2.2 Microsoft Windows dnes ... 4
   2.3 .NET Framework ... 5
   2.4 Windows Presentation Foundation ... 6
   2.5 Universal Windows Platform (UWP) ... 7
   2.6 Porovnanie UWP a WPF ... 8
   2.7 Životný cyklus UWP aplikácie ... 9
   2.8 MVVM architektúra ... 11
   2.9 PRISM Framework ... 12
3. Prehľad podobných aplikácií ... 13
   3.1 MobilKredit ... 13
   3.2 WebKredit ... 14
4. Zber požiadaviek ... 15
5. Analýza požiadaviek ... 16
6. Návrh aplikácie ... 17
   6.1 Klient - server architektúra ... 17
   6.2 Komunikačný protokol ... 18
   6.3 Návrh zobrazenia informácií ... 19
   6.4 Návrh GUI ... 20
7. Implementácia ... 21
   7.1 Použité technológie ... 21
      7.1.1 Jazyk C# ... 21
      7.1.2 Vývojové prostredie ... 21
      7.1.3 Resharper ... 22
      7.1.4 Code Metrics ... 22
      7.1.5 GitHub ... 22
   7.2 Aplikácia ... 23
      7.2.1 Štruktúra aplikácie ... 23
      7.2.2 Model ... 24
      7.2.3 ViewModel ... 25
      7.2.4 View ... 25
      7.2.5 DialogService ... 25
      7.2.6 DialogResult ... 27
      7.2.7 NavigationService ... 27
      7.2.8 Štruktúra GUI ... 28
8. Možnosti rozšírenia ... 36
9. Záver ... 37
10. Literatúra ... 38

1. Úvod

Svet, v ktorom žijeme sa neustále vyvíja a spolu s ním aj technológie. Len prednedávnom bolo zvykom objednávať si jedlo vo firmách alebo v menzách orazením si čísla vybratého jedla na lístku na nasledujúci deň. V súčasnosti je možné objednávať jedlá z jedálneho lístka prostredníctvom vstavaného terminálu priamo v menze alebo prostredníctvom svojho osobného počítača cez internet. Avšak v dobe plnej nových technológií, ktoré uľahčujú prácu ľuďom, sa očakáva objednávanie jedla prostredníctvom iných zariadení, ako je napr. mobilný telefón, či tablet a objednávanie jedla z rôznych miest, či už je to z domova alebo z nákupného strediska.

Dnešná doba prináša so sebou rôzne typy zariadení, na ktorých je možné tieto objednávky sprostredkovať. Ich implementácia je však často krát náročná. Pri tvorbe jednej aplikácie na rôzne zariadenia, je potrebné vytvoriť viac verzií pre pracovnú plochu, pre mobilný telefón, či tablet.

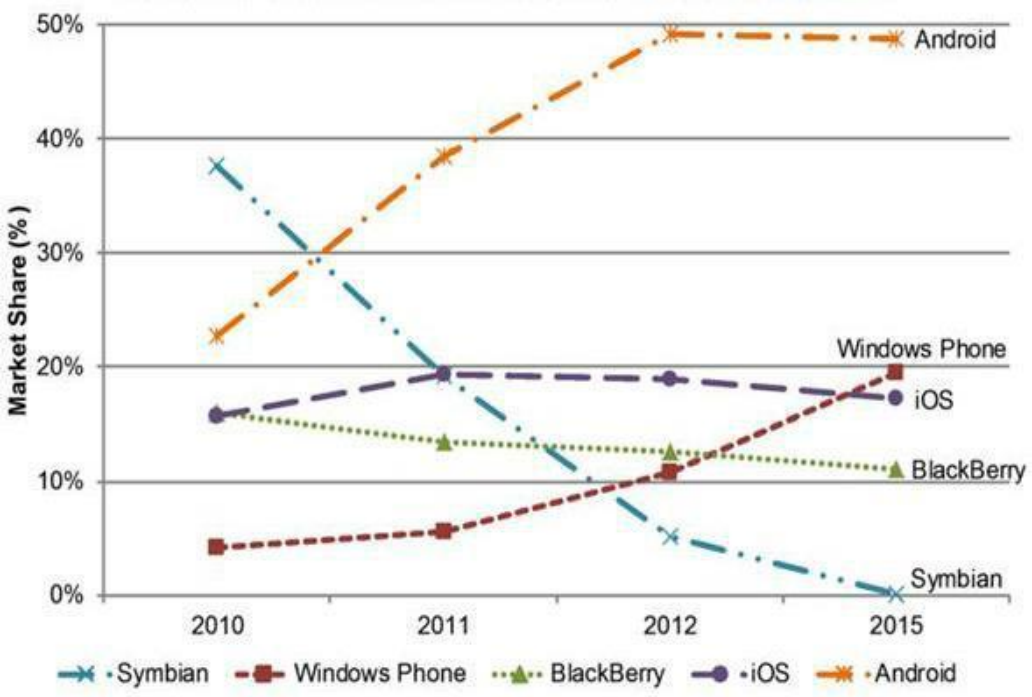
Cieľom tejto bakalárskej práce je vytvoriť mobilnú aplikáciu pre objednávanie jedál pre operačný systém Windows 10, ktorý umožňuje pri správnom návrhu vytvorenie iba jednej aplikácie, ktorá bude schopná fungovať na rôznych zariadeniach. Aplikácia umožní používateľom vyhľadať ponúkané jedlá na rôznych výdajných miestach, objednávanie jedál, rušenie, zmenu výdajne objednaného jedla. Aplikácia bude uchovávať históriu objednaných jedál a informácie o používateľovi, ako sú jeho meno a priezvisko, stav účtu, disponibilný zostatok a iné.

Windows 10 ponúka novú technológiu UWP, ktorá umožňuje funkčnosť jednej dobre navrhnutej aplikácie na rôznych zariadeniach ako je mobilný telefón, tablet, notebook, stolný počítač a iné zariadenia. Podmienka je, že dané zariadenie musí pracovať na operačnom systéme Windows 10. Výsledná aplikácia bude spustená na tablete, ktorý bude k dispozícii v menze. Taktiež bude prístupná na mobilné telefóny a iné zariadenia s operačným systémom Windows 10 pripojené na internet.

Úvodná kapitola sa zaoberá analýzou trhu aplikácií, ktoré sa využívajú na viacerých zariadeniach a technológiou ich tvorby. Ďalšie kapitoly sú zamerané na návrh, analýzu procesu tvorby aplikácie, implementáciu a testovanie.

2. Súčasný stav

Na obrázku 1 možeme vidieť podiel na trhu jednotlivých mobilných operačných systémov (ďalej OS) vo svete. Svetovým lídrom mobilného OS trhu je Android s podielom 64 %, ktorý stále rastie o 2% mesačne. Windows Phone, ktorý v roku 2010 znamenal 5% svetového podielu na trhu Smartfónov, je dnes druhým najpredávanejším operačným systémom s podielom na trhu zvýšeným na 19 %. Operačný systém iOS, ktorý len prednedávnom bol za Androidom druhým najpredávanejším mobilným OS je dnes na treťom mieste s podielom 16 % na svetovom trhu.



Obrázok 1: Vývoj používania mobilných OS [1]

K výraznému stúpaniu popularity predaja Windows Phone operačného systému pomohlo niekoľko noviniek, ktoré prináša spoločnosť Microsoft. S vývojom nových technológií doba prináša mobilné telefóny, ktoré svojou výkonnosťou nahradzujú krabicové stolné počítače, či notebooky. Spoločnosť Microsoft využila túto možnosť a vyvíja prostredie, v ktorom bude možné celý stolný počítač nosiť so sebou vo vrecku v mobilnom telefóne. Vytvára prepojenie medzi mobilným telefónom a obrazovkou, ktoré by malo nahradiť plne funkčný novodobý počítač s prípadným pripojením ďalšieho vstupno-výstupného zariadenia, ako je napríklad klávesnica, či myš.

Táto novinka prišla spolu s novým operačným systémom Windows 10, ktorý umožňuje tvorbu aplikácie spustiteľnej na viacerých zariadeniach. Zvyšuje popularitu a je zatiaľ bezkonkurenčná. Na základe týchto údajov je možné vidieť stúpajúcu popularitu Windows Phone OS. To odpovedá na otázku, prečo vytvárať aplikáciu pre Windows Phone a prečo bola práve táto práca vytvorená pre tento operačný systém. V nasledujúcej podkapitole je opísaná novinka od firmy Microsoft, ktorá tak isto posunula umiestnenie operačného systému Microsoft Phone do popredia medzi ostatnými mobilnými operačnými systémami.

2.1 Microsoft Continuum

Jednou z najočakávanejších funkcií Windows 10 Mobile je funkcia Continuum, ktorá by s trochou šťastia mohla Microsoftu opäť zabezpečiť vzostup na výslnie aj v oblasti mobilných telefónov. Rada používateľov nahradila klasické PC tabletom a mobilné telefóny preberajú čoraz viac pracovných úloh. Hoci telefóny sa svojím výkonom pomaly vyrovnajú štandardnému PC, stále narážajú pri dlhodobej práci na ergonomické nedostatky. Jednoducho povedané, stále sú (a musia byť kvôli mobilite) príliš malé. Zostáva teda otázka, keď už máme dostatočný výkon, prečo prístroj nepripojíme k ďalšiemu príslušenstvo a nevytvoríme z neho pohodlné PC? Táto myšlienka je už veľmi stará a technologickí nadšenci vždy snívali o tom, že budú môcť všetky zariadenia nahradiť jediným. A práve túto myšlienku si vzal Microsoft za svoju a v podstate ako prvý výrobca sa ju snaží aj realizovať prostredníctvom Windows 10 (Mobile).

Aby bolo možné zariadenie pripojiť k ďalším perifériám, nebudú stačiť štandardné bezdrôtové prenosy, resp. budú (Miracast pre pripojenie LCD displeja a Bluetooth pre pripojenie ďalšieho príslušenstva), ale tento spôsob nemusí byť úplne spoľahlivý a hlavne pohodlný na reálne používanie. Z tohto dôvodu Microsoft vytvoril univerzálne riešenie využívajúce nové špeciálne zariadenie s názvom Microsoft Display Dock.

Microsoft Display Dock

Microsoft Display Dock je malá čierna škatuľka vybavená celou radou portov, ktoré poslúžia na prepojenie telefónu s ďalšími zariadeniami. Z jednej strany sa pripojí pomocou klasického USB kábla (typ C) mobilné zariadenie a z druhej strany cez HDMI, či DisplayPort klasický LCD monitor alebo TV. K dispozícii je tiež dvojica klasických USB 2.0 konektorov pre pripojenie ďalších periférií, ako je myš, klávesnica, USB kľúč, tlačiareň a pod. Samotný Dock je s preplňovaným vznetovým vlastným napájacím zdrojom (USB - C), čo umožní pripojený telefón počas práce štandardne nabíjať. Na obrázku môžete vidieť aplikovanie Microsoft Continuum pomocou Microsoft Display Dock-u.

Obrázok 2: Microsoft Continuum spojený Microsoft Display Dock-om [2]

2.2 Microsoft Windows dnes

Microsoft Windows je séria niekoľkých rodín operačných systémov vyvíjaných spoločnosťou Microsoft. V súčasnosti sa odhaduje, že až viac ako 90% všetkých osobných počítačov má nainštalovanú niektorú verziu systému Microsoft Windows. Windows dosiahol túto enormnú pozíciu na trhu čiastočne vďaka dominancii operačného systému MS-DOS v minulosti, čiastočne taktiež z dôvodu, že je primárnou platformou populárneho balíka Microsoft Office.

Windows prichádza dnes predinštalovaný na väčšine počítačov (ako tzv. OEM edícia), čo z neho robí prvotnú voľbu pre používateľov. Väčšina zákazníkov, ktorí chcú používať iný operačný systém, nezmaže Windows a alternatívny systém nainštaluje po jeho boku. V nasledujúcej tabuľke (Tabuľka 1) je vidieť podiel na trhu a použitie Windows operačného systému v porovnaní s ďalšími Windows operačnými systémami a všetkými operačnými systémami.

Operačný systém	Podiel na trhu	Windows OS	Všetky OS
Windows 2000	0.01%	0.04%	0.02%
Windows XP	10.59%	9.73%	4.86%
Windows Server 2003	-	0.14%	0.07%
Windows Vista	1.61%	2.00%	1.00%
Windows 7	56.11%	56.38%	28.16%
Windows Phone 7.5	0.24%	1.70%	0.85%
Windows Phone 8	0.51%	1.70%	0.85%
Windows Phone 8.1	2.25%	1.70%	0.85%
Windows 10 Phone	0.09%	1.70%	0.85%
Windows 8	2.88%	3.38%	0.85%
Windows 8.1	11.15%	14.90%	0.85%
Windows RT 8.1	-	0.08%	0.85%
Windows 10	9.00%	11.65%	0.85%
Všetky varianty	94.44%	100.00%	0.85%

Tabuľka 1: Použitie OS Windows v porovnaní s ďalšími OS

2.3 .NET Framework

.NET Framework je zastrešujúci názov pre súbor technológií v softvérových produktoch, ktoré tvoria celú platformu, ktorá je dostupná nielen pre Web, Windows aj Pocket PC. Common Language Infra - structure je štandardizovaná špecifikácia jadra .NET .

Základným komponentom je Microsoft .NET Framework, prostredie potrebné pre beh aplikácií, ktoré ponúka spúšťacie rozhranie, ako aj potrebné knižnice.

Dostupnosť Platformy

- **Microsoft .NET Framework** je najrozšírenejšia platforma pre osobné počítače s operačným systémom Microsoft Windows od verzie Windows 98.
- **Microsoft .NET Compact Framework** je platforma určená pre vreckové počítače a mobilné telefóny s operačným systémom Windows Mobile.
- **Microsoft .NET Micro Framework** je platforma určená pre vstavané zariadenia s ešte menšou výpočtovou kapacitou a väčšími obmedzeniami, než vreckové počítače.
- **Mono** je produktom nezávislej open source iniciatívy, implementujúcej .NET Runtime pre operačné systémy UNIX-ového typu (Linux, Mac OS X).

Aktuálna verzia 4.5 je pre majiteľov operačného systému Windows k dispozícii zdarma ako samostatný komponent, ktorý sa do systému doinštaluje (býva šírený na CD, či DVD rôznych počítačových časopisov, ako súčasť rôzneho software a možno ju tiež stiahnuť samostatne alebo cez Windows Update). V nasledujúcom obrázku môžeme vidieť architektúru .Net frameworku.

Obrázok 3: .Net Framework architektúra [3]

2.4 Windows Presentation Foundation

Windows Presentation Foundation (WPF alebo) je grafický subsystém pre vykresľovanie užívateľských rozhraní v aplikáciách založených na systéme Windows od spoločnosti Microsoft. WPF, predtým známy ako "Avalon", bola pôvodne vydaná ako súčasť rozhrania .NET Framework 3.0. Skôr než sa spoliehať na staršie GDI subsystému, WPF používa DirectX. WPF sa snaží poskytnúť konzistentné programovací model pre vytváranie aplikácií a oddeľuje užívateľské rozhranie od obchodnej logiky. To sa podobá modelom objektov XML, ako sú realizované v XUL a SVG.

WPF používa jazyk XAML. XAML je relatívne jednoduchý a všestranne použiteľný deklaratívny programovací jazyk, vhodný pre výstavbu a inicializáciu objektov. XAML je XML, rozšírený o súbor pravidiel prvkov a atribútov a ich mapovanie na objekty, ich vlastnosti a na hodnoty týchto vlastností. [4]

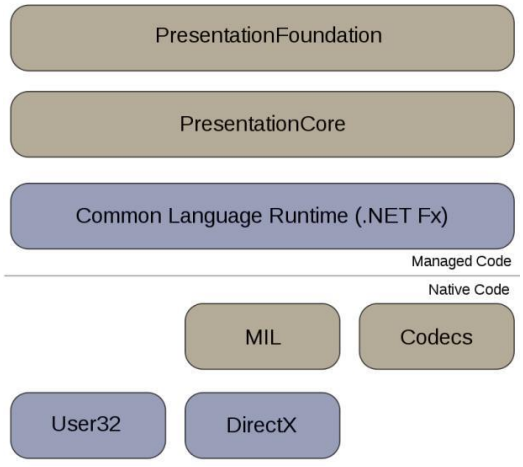
WPF aplikácie môžu byť nasadené ako samostatné desktopové programy, alebo hostované ako vložený objekt vo webových stránkach. WPF si kladie za cieľ zjednotiť niekoľko spoločných prvkov používateľského rozhrania, ako sú 2D / 3D rendering, pevné a adaptívne dokumenty typografie, vektorová grafika, runtime animácie, a pre-rendered médiá. Tieto prvky potom môžu byť spojené a je možnosť s nimi manipulovať na základe rôznych udalostí, interakcie užívateľa a viazanie dát.

WPF runtime knižnice sú súčasťou všetkých verzií Microsoft Windows od operačného systému Windows Vista a Windows Server 2008. Používatelia Windows XP SP2 / SP3 a Windows Server 2003 môžete voliteľne nainštalovať potrebné knižnice.

Microsoft Silverlight poskytuje funkcie, ktoré sú väčšinou podmnožinou WPF, ktorý poskytuje vstavané webové ovládacie prvky porovnateľné s Adobe Flash. 3D rendering runtime bol podporený v Silverlight od verzie Silverlight 5.

Architektúra WPF

Architektúra WPF je rozprestretá na dve časti súčasne, spravovaný kód (Managed Code) a natívny kód (Native Code). Avšak, verejné API je k dispozícii iba prostredníctvom spravovaného kódu. Zatiaľ čo väčšina WPF je v spravovanom kóde, composition engine, ktorý poskytuje aplikácie WPF je súčasťou natívneho kódu. Na obrázku môžete vidieť kompletnú architektúru technológie WPF.



Obrázok 4: WPF architektúra (modré elementy sú Windows komponenty, hnedé sú WPF komponenty) [5]

2.5 Universal Windows Platform (UWP)

Universal Windows Platform (UWP) umožňuje vytvárať aplikáciu, ktorá má potenciál spustiť na akomkoľvek operačnom systéme Windows zariadenia:

- **Rodina mobilných zariadení:** Windows Phones, phablets
- **Rodina stolných zariadení:** tablety, notebooky, PC
- **Rodina tímových zariadení:** Surface hub
- **Rodina IoT:** kompaktné zariadenia ako sú wearables alebo domáce spotrebiče

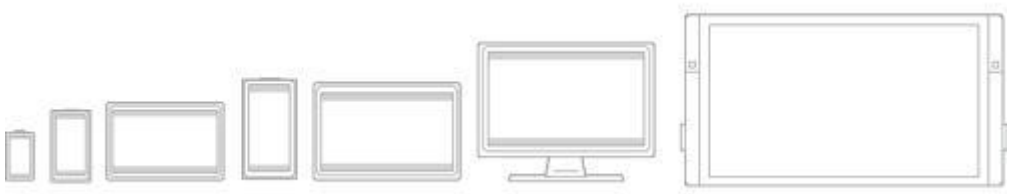
Môžete obmedziť vaše aplikácie do jediného zariadenia rodiny (napríklad mobilné zariadenia), alebo si môžete zvoliť, aby bola aplikácia k dispozícii na všetkých zariadeniach so systémom Windows. Len projektovanie aplikácie, ktorá vyzerá dobre na všetkých mobilných zariadeniach môže byť veľkou výzvou. Navrhovanie pre viac rodín zariadení vyžaduje niektoré ďalšie pozornosti, plánovanie a dizajn, ale Universal Windows Platform (UWP) poskytuje sadu vstavaných funkcií a univerzálne stavebné vlastnosti, ktoré zjednodušujú vytvorenie skvelého užívateľského zážitku pre viac zariadení.

Ak vaša aplikácia beží na Windows zariadení , systém používa algoritmus na normalizáciu spôsobu ovládania, písma a ďalšie prvky používateľského rozhrania zobrazené na obrazovke. Tento scaling algoritmus berie do úvahy pozorovaciu vzdialenosť a hustotu obrazovky (pixlov na palec) pre optimalizáciu a pre vnímanú veľkosť (fyzická veľkosť). Scaling algoritmus zabezpečí, že 24 px písmo na Surface Hub 10 feet away je rovnako čitateľné užívateľovi ako 24 px písma na 5" telefónu, ktorý je o pár palcov ďalej.

Univerzálny vstup a inteligentné interakcie

Pre UWP nie je potrebné navrhovať aplikácie pre konkrétne vstupné zariadenia, pretože UWP aplikácie používajú vstupný systém, ktorý používa " inteligentné" interakcie . To znamená, že môžete navrhnúť interakciu kliknutím bez toho, aby museli vedieť, či kliknutie pochádza zo skutočného kliknutia myšou alebo ťuknutím na obrazovku prstom.

Na obrázku je možné vidieť rôzne typy zariadení, na ktorých bude aplikovaná technológia UWP, ktorá umožňuje vytvorenie jednej aplikácie na všetky typy zariadení.



Obrázok 5: Aplikovanie UWP na rôzne typy zariadení [6]

2.6 Porovnanie UWP a WPF

WPF technológia má veľa výhod pri tvorbe aplikácií využívaných na jednom konkrétnom type zariadenia. Je to staršia technológia ako UWP, a teda je rozšírenejšia a viac dokumentovaná. UWP technológia je využiteľnejšia pri tvorbe aplikácie, ktorá bude pracovať na rôznych typoch zariadení. Je menej dokumentovaná a menej odskúšaná ako WPF. Pri tvorbe UWP aplikácie je možnosť stretnúť sa s novými technikami.

Z histórie je možné vidieť vývoj projektov jazyka C#. Prvotné programovanie bolo podobné jazyku C++, no neskôr Microsoft predstavil WPF. WPF technológia priniesla ďalší layer, ktorý priniesol mnoho zložitosti do jazyka C#. Neskôr sa stal populárny návrhový vzor MVVM. Táto metodológia je komplexná a z 3-hodinového pojektu sa stal 3-dňový.

Aktuálnym vývojom projektov jazyka C# a novinkou je technológia UWP, ktorá priniesla veľa ďalších možností, spomínaných v predchádzajúcej podkapitole. Technológia UWP nepriniesla nové veci do vývoja C# projektov ako by sa mohlo zdať, namiesto toho priniesla veľa rozdielov pri programovaní. Vývojári tejto technológie odstránili štandardné funkcie zo starších technológií ako napríklad z WPF, Silverlight, a iné. Hlavné funkcie, ktoré boli používané vo WPF a v UWP chýbajú sú opísané v ďalšej časti:

- MultiBinding je zrušený
- ConverterParameter nie je povolený pre Binding
- ICommand očakáva od programátora vyvolanie udalosti CanExecuteChanged
- DataTriggers sú zrušené
- Pribudol VisualStateManager, vďaka ktorému jediná cesta pre nastavovanie property je cez komplexné Storyboard-y
- Inštalácia UWP aplikácia je komplikovanejšia ako WPF
- UWP zrušilo DockPanel a ďalšie WPF prvky, ktoré boli nahradené novými
- X:Static binding-y sú zrušené
- X:Type binding-y sú zrušené, v UWP je potreba predávať typ cez CommandParameter

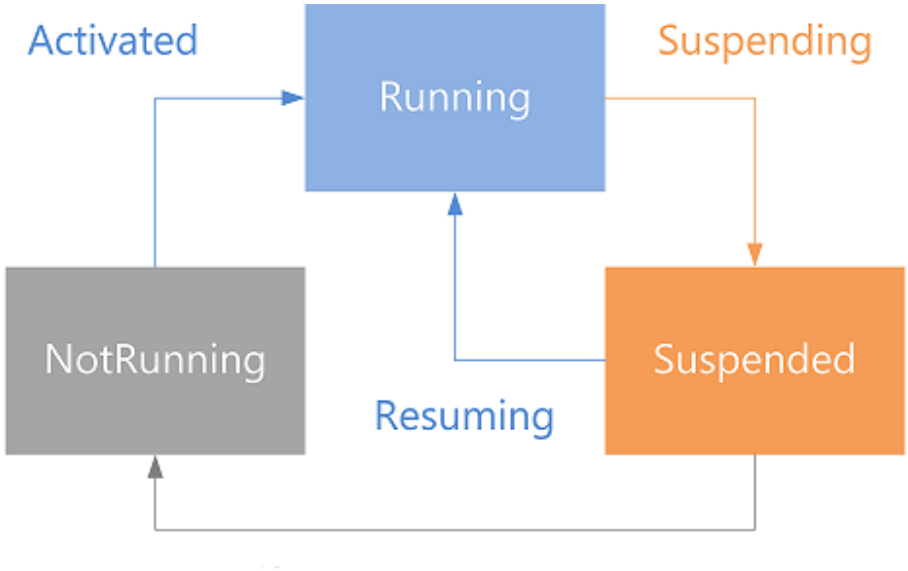
Pre tvorbu bakalárskej práce som sa rozhodol pre technológiu UWP, na základe špecifikácie výslednej aplikácie. Aplikácia má bežať na zariadení typu tablet, ktorý bude dostupný v menze pre objednávanie jedál. Taktiež má táto aplikácia fungovať aj na mobilnom zariadení, cez ktoré sa bude dať objednať jedlo napríklad na nasledujúci deň aj z domu. Táto aplikácia sa dá využiť aj na stolnom počítači. Aj napriek tomu, že technológia odstránila staré zvyklosti zo starších technológií, poskytuje tvorbu aplikácie pre rôzny typy zariadení, čo je dôležitý faktor pri voľbe technológie.

Pri zvolení technológie WPF by bolo potrebné vytvoriť rôzne aplikácie pre mobil, tablet a iné zariadenia, čo nie je efektívne z hľadiska návrhu. Technológia UWP rieši tento problém a vďaka nej je možné vytvoriť aplikáciu na rôzne zariadenia.

2.7 Životný cyklus UWP aplikácie

Táto podkapitola popisuje životný cyklus Universal Windos Platform (UWP) aplikácií, od okamžiku, kedy je aktivovaná, až po jej uzavretie. Veľa používateľov rozšírilo svoju prácu na rôznych typoch zariadení. Používatelia môžu očakávať, že aplikácia si bude pamätať svoj stav aj počas multitaskingu. Napríklad keď sa stránka aplikácie zoscrolluje a vráti do pôvodnej pozície, tak by mala byť v rovnakom stave ako predtým. Vďaka exekučným stavom do ktorých sa aplikácia dostáva je tento druh chovania bezproblémový.

Nasledujúci obrázok predstavuje prechod medzi stavmi, do ktorých sa aplikácia môže dostať. Životný cyklus UWP aplikácie pozostáva zo stavov NotRunning, Running a Suspended, do ktorých sa aplikácia dostane cez zahájenie, pozastavenie a obnovenie.



Obrázok 6: Exekučné stavy UWP aplikácie [7]

V nasledujúcej časti sú popísané jednotlivé časti životného cyklus UWP aplikácie.

Štart aplikácie

Aplikácia sa spustí, keď je v NotRunning stavu. Do tohto stavu sa môže dostať aplikácia ak ešte nebola spustená, alebo bola spustená, ale potom sa zrútila, alebo bola pozastavená (Suspended) a nemohla byť uložená v pamäti a tak bola ukončená systémom. Po spustení aplikácie, Windows zobrazí úvodnú obrazovku (Splash Screen). Zatiaľ, čo je úvodná obrazovka spustená, aplikácia by mala pripravovať užívateľské rozhranie. Hlavnými úlohami aplikácie je registrácia obsluhy udalostí (event handler) a nastavenie vlastného UI, ktoré potrebuje pre načítanie úvodnej stránky. Tieto úlohy by mali trvať len niekoľko sekúnd. Po dokončení aktivácie aplikácie, vstúpi bo bežiaceho stavu (Running) a úvodná obrazovka zmizne (všetky jej zdroje a objekty sú vymazané).

Aktivácia aplikácie

Aplikácia môže byť používateľom aktivovaná pomocou kontraktov. Aktivačný kód aplikácie môže otestovať prečo bola aplikácia aktivovaná a či bola aktivovaná v stave Running. Aplikácia môže obnoviť predtým uložené dáta pri aktivácií v prípade, že operačný systém ukončí aplikáciu a používateľ ju následne obnoví. Windows môže ukončiť aplikáciu potom, čo bola pozastavená z niekoľkých dôvodov. V prípade, že používateľ spustí aplikáciu Windows potom, čo bola ukončená, používateľ vidí úvodnú obrazovku pokiaľ nedôjde k aktivácii aplikácie.

Pozastavenie aplikácie

Aplikácia sa do stavu pozastavenia (Suspend) dostane vždy, keď sa používateľ prepne do inej aplikácie, alebo na plochu, alebo na Štart. Aplikácia môže byť prerušená aj vtedy, keď zariadenie prejde do stavu nízkej spotreby energie. Systém obnoví aplikáciu vždy, keď sa používateľ prepne späť k nej. Po návrate na aplikáciu, obsah premenných a dátových štruktúr je rovnaký ako bol predtým, ako systém pozastavil aplikáciu. Systém obnoví aplikáciu tam, kde skončila, takže sa javí používateľovi ako keby bežala na pozadí.

Medzi stavmi Running a Suspended nastane prechodný stav, kedy je aplikácia na pozadí, ale Windows ešte niekoľko sekúnd čaká, aby sa uistil, že používateľ sa nechce vrátiť späť do aplikácie. Je to z dôvodu, že je určitá pravdepodobnosť, kedy používateľ sa z iného programu, prípadne domovskej obrazovky, prepne znovu na aplikáciu. V tomto prípade je aplikácia na pozadí systémom vnímaná ako bežiaca, nedochádza k jej pozastaveniu a po prepnutí je nábeh okamžitý.

Po uplynutí tohto časového intervalu systém automaticky presunie aplikáciu do úsporného režimu. Aplikácia v stave pozastavenia je zavedená v pamäti, ale v tomto stave nemôže vykonávať žiadnu činnosť. V priebehu prechodu môže aplikácia napríklad uložiť neuložené dáta do úložného zariadenia. Na toto uloženie dát má maximálne 5 sekúnd. V prípade, že systém má dostupný dostatok prostriedkov, umožní vám previezť aj dlhšie trvajúcejšiu operáciu.

Pokračovanie aplikácie

Aplikácia sa do stavu pokračovania (Resume) dostane keď sa používateľ prepne na ňu, alebo keď sa dostane zariadenie z nízkej spotreby energie. Keď aplikácia pokračuje od Suspend stavu, vstúpi do Running stavu a pokračuje od miesta, kde bola aplikácia, keď bola prerušená. Žiadne dáta z aplikácie nie sú stratené. Pokiaľ má aplikácia obsah alebo internetové pripojenie, ktoré je potrebné obnoviť, mali by byť aktualizované, keď nastane Resume aplikácie. Ak je aplikácia v stave Suspend a má byť aktivovaná, obdrží najprv udalosť pre Resume a následne až aktivačnú udalosť. [7]

2.8 MVVM architektúra

Model-View-ViewModel je architektonický vzor, ktorého hlavným zámerom je uľahčenie oddelenia vývoja grafického užívateľského rozhrania (či už ako značkovacieho jazyka alebo GUI kódu) od rozvoja biznis logiky alebo back-end logiky (model dát). ViewModel je zodpovedný za vystavenie dátových objektov z Modelu na View, čím sú objekty ľahko spravovateľné. Komponenty MVVM sú triedy, oddelené z abstraktných tried Model, View a ViewModel. [8]

Model

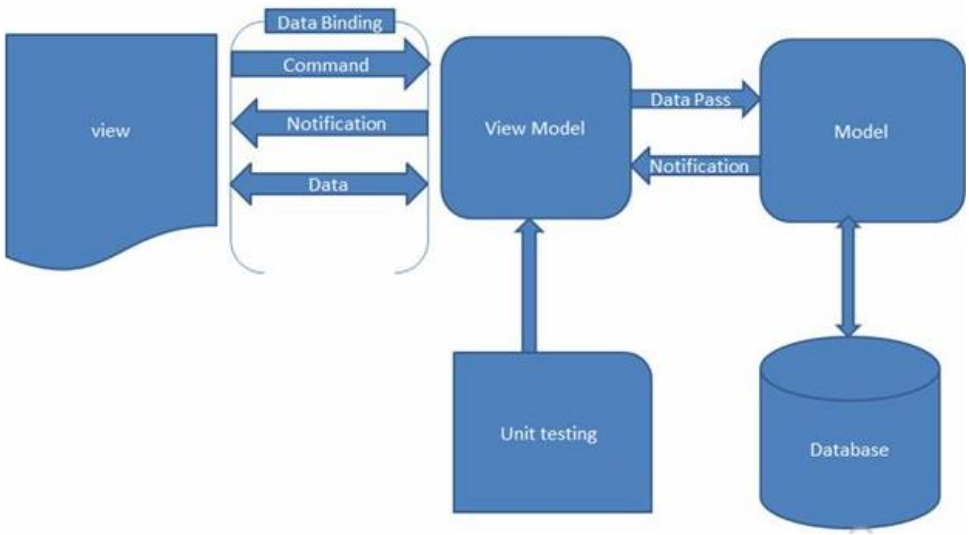
Obsahuje výpočty, databázové dotazy, validáciu, logiku celej aplikácie. Model nie je spojený s výstupom. Jeho funkcia spočíva v prijatí parametrov a následné vydanie dát iným komponentom MVVM vzoru. V klientskych aplikáciách komunikujúcich s aplikačným serverom model obsahuje zoznam entít prichádzajúcich zo servera.

View

Definuje štruktúru, rozloženie a vzhľad, ktorý používateľ vidí na obrazovke. Najčastejšie sa jedná o WPF alebo UWP technológiu, ktorá využíva tagy značkovacieho jazyka XAML s obmedzeným kódom na pozadí, ktorý neobsahuje biznis logiku. View štandardne má jeden ViewModel, ktorý však v sebe môže obsahovať aj ďalšie vnorené ViewModel-y. Umožňuje staticky nastavovať atribúty jednotlivým komponentom aplikácie, ako aj naviazať hodnotu atribútu s ViewModel-om daného View pomocou Binding.

ViewModel

Je chýbajúci prvok, ktorý vytvára prostredníka medzi View a Modelom. View a Model navzájom nekomunikujú a ViewModel sa stará o získanie dát z Modelu, ktoré spracuje a nastaví pravidlá predania View. ViewModel drží celý systém pohromade a komponenty prepája. Existuje veľa prístupov, najčastejšie má každý View jeden ViewModel a každý ViewModel spravuje práve jeden View pre správne navrhnutý MVVM vzor. ViewModel, ako aj Model, by mali byť testovateľné. Pre ViewModel-y sa štandardne vytvárajú Unit testy pre testovanie metód využitých vo ViewModel-i.



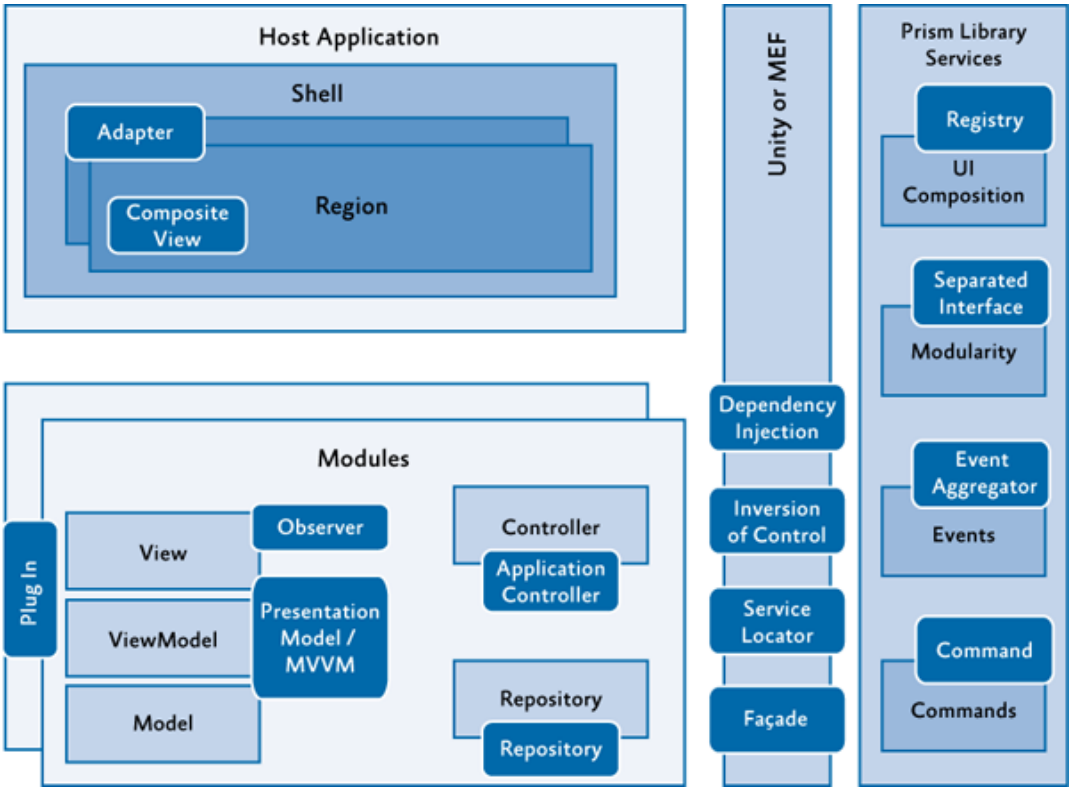
Obrázok 7: MVVM architektúra [9]

2.9 PRISM Framework

Prism poskytuje návod, ktorý pomáha jednoduchšie navrhovať a vytvárať flexibilné a ľahko udržiavateľné klientske aplikácie, ktoré bežia na Windows Runtime,WPF alebo UWP desktop, Silverlight alebo Windows Phone. Tieto aplikácie môžu vznikať ako malé aplikácie a neskôr sa môžu zväčšovať. Prism pomáha navrhovať a vytvárať aplikácie, ktoré vytvárajú interakciu medzi prezentačnou a biznis logikou s back-end systémom a jeho službami a pomocou vrstvenej architektúry, môžu byť fyzicky rozmiestnené v niekoľkých vrstvách. Pri vývoji cez framework Prism sa očakáva, že sa aplikácia bude výrazne vyvíjať v priebehu svojej životnosti v reakcií na nové požiadavky a obchodné príležitostí. Z toho vyplýva, že aplikácie sú stavané na výdrž a pripravené na rôzne zmeny. [10]

Hlavnou výhodou tohto Framework-u je použitie modularity prístupu, kde je aplikácia rozdelená do rôznych modulov. Tento prístup pomáha pri súčasnom vývoji aplikácií, kde niektoré tímy môžu pracovať na rovnakom projekte. Toto umožňuje významne zrýchliť vývoj aplikácií a tiež minimalizuje závislosti medzi rôznymi tímami a umožňuje rôznym tímom sústrediť sa na ich oblasť záujmu, ako dizajn užívateľského rozhrania, implementácia biznis logiky a vývoj kódu.

Spolu s Model-View-ViewModel (MVVM) architektonickým štýlom, Prism používa aj iné návrhové vzory ako Adapter, Application Controller, Command, Composite a Composite View, Event Aggregator, Façade, Observer, Registry, Repository, Separated Interface, Plug-in, Service Locator, Dependency Injection (DI), Inversion of Control (IC), Separation of Concerns (SoC) a iné. Nasledujúci obrázok ilustruje štandardnú kompozitnú aplikačnú architektúru s použitím Prism knižnice. Aplikácie by mali využívať niektoré z týchto návrhových vzorov.



Obrázok 8: Príklad kompozitnej aplikačnej architektúry so štandardnými návrhovými vzormi [11]

3. Prehľad podobných aplikácií

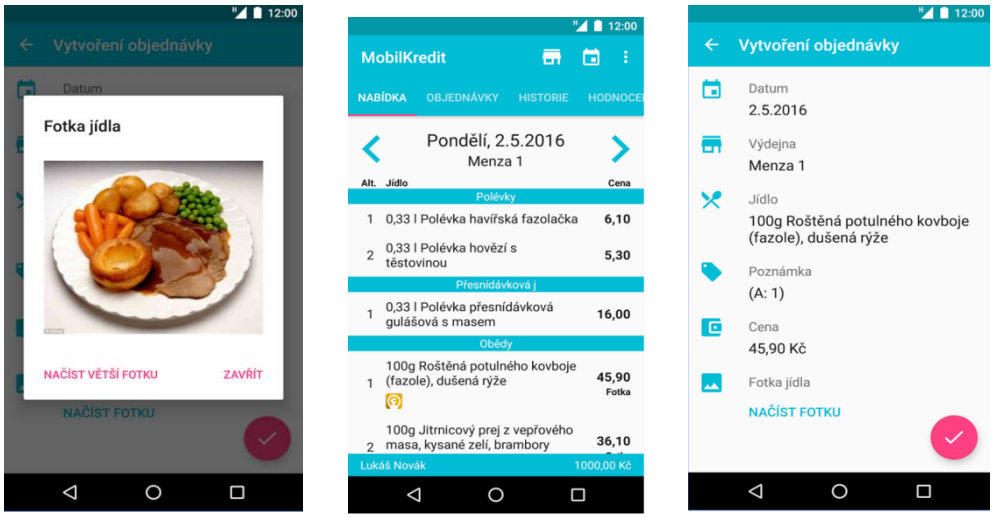
Pred návrhom aplikácie bolo prínosné preskúmať aplikácie s rovnakou alebo podobnou tematikou. Pred vývojom mi bola poskytnutá dokumentácia k obsluhe a implementácií beta-verzie 4.0.0 Android aplikácie MobilKredit, ako aj manuál k webovej stránke WebKredit. Taktiež bola voľne dostupná Android aplikácia MobilKredit starej verzie 3.0.0 na Google Play, ktorá mi rovnako pomohla pri tvorbe návrhu výslednej aplikácie. Všetky spomínané aplikácie a webová stránka vyvíja firma Anete s.r.o., pre ktorú som vyvíjal túto prácu. Z verejných vysokých škôl využíva systémy od firmy Anete 11 škôl. Ak škola využíva tento systém, tak pre všetky jej menzy.

3.1 MobilKredit

Aplikácia MobilKredit je momentálne aplikáciou poskytujúcou objednávanie jedál do 6 vysokých škôl. Ide o mobilnú alternatívu internetového objednávania WebKredit. Aplikácia je dostupná na Android cez Google Play a pre iOS cez AppStore.

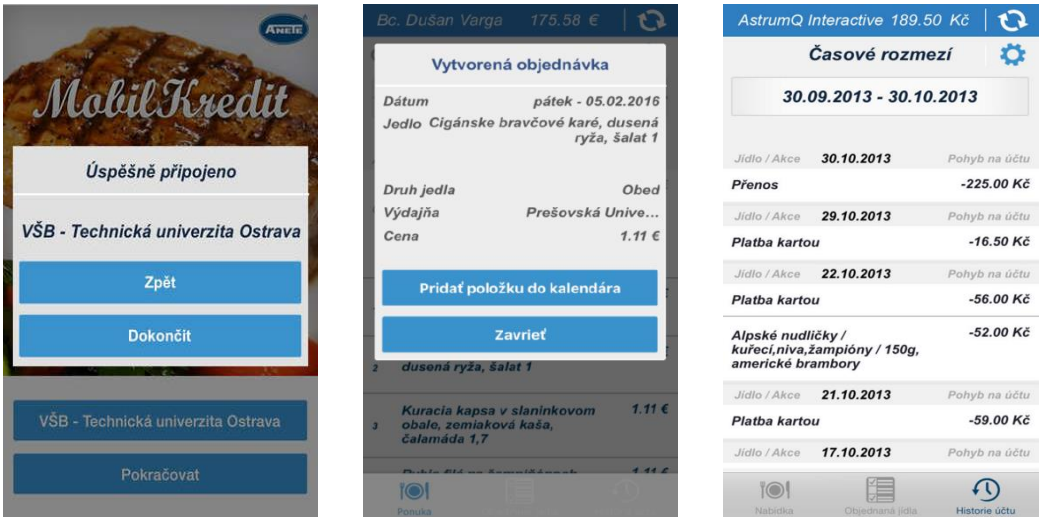
Funkcie tejto aplikácie sú objednávanie jedla pre zvolený dátum a výdajňu a prípadné uloženie do kalendára, zoznam objednávok a história účtu, odhlasovanie jedál, práca s burzou jedál, možnosť pridania objednávky do kalendára, informácie o zostatku na účte, lokalizácia aplikácie do angličtiny, nemčiny a slovenčiny. Obsahuje funkcie sprístupnené iba niektorým stravovacím zariadeniam ako zobrazovanie fotiek jedál, zobrazovanie posledných porcií, možnosť ohodnotiť výdajne a objednané jedlá.

Aplikácia MobilKredit pre Android má pekný dizajn, rýchla dostupnosť dát a nie je preplnená zbytočnými funkciami. Pri spustení aplikácie je potrebné prihlásenie, z ktorého aplikácia prejde na hlavnú stránku s ponukou jedla na aktuálny deň a s vybratou menzou, ktorú má používateľ prednastavenú, alebo ak nemá, tak mu zobrazí prvú menzu, ktorá je dostupná pre konkrétneho prevádzkovateľa podľa nastavení používateľa. Zo stránky ponuky jedla sa dá dostať na stránky: objednávky jedla, ktorá obsahuje zoznam objednaných jedál, na stránku histórie účtu, kde je zoznam akcií na účte a na stránku s hodnotením, v ktorom používateľ môže ohodnotiť aplikáciu, kvalitu jedla alebo výdajne. Ďalšou možnosťou aplikácie je výber výdajne prostredníctvom tlačítka, v ktorej je zoznam platných výdajní a tlačítko Kalendár, ktorí zobrazuje kalendár a informácie o objednaných jedlách. Grafické znázornenie Android aplikácie je na obrázku 9.



Obrazok 9: MobilKredit pre Android [12]

Mobilná aplikácia pre iOS má starší dizajn, ale funkcionalitou sa nelíši od Android verzie. Grafické znázornenie iOS aplikácie MobilKredit je na obrázku 10.



Obrázok 10: MobilKredit pre iOS [13]

3.2 WebKredit

Je Webová aplikácia, ktorá umožňuje obsluhu stravovacieho účtu. Aplikáciu využíva 11 vysokých škôl. Možnosťami sa veľmi nelíši od mobilnej aplikácie pre objednávanie jedla. Umožňuje objednanie, alebo rušenie jedál, prípadné ponúkanie do burzy, zisťovanie finančného zostatku, alebo počtu vyčerpaných dotácií tzv. "bonov", prehľadávanie jedálnička, kontrola cien, história účtu, obsluha finančných prostriedkov SUPO kredit. Na nasledujúcim obrázku je ukážka z aplikácie pri štandardnom používaní. Grafické znázornenie webovej aplikácie WebKredit je na obrázku 11.

Obrázok 11: webová aplikácia WebKredit [14]

4. Zber požiadaviek

Pre tvorbu aplikácie je dôležité špecifikovať požiadavky: aké vlastnosti má mať aplikácia, do akých stavov sa má dostať, ako aj kontrola, aby sa aplikácia nedostala do nedefinovaných stavov. Aplikácia má slúžiť ako terminál pre objednávanie jedál spustiteľný na tablete vstavanom v menzách. Táto aplikácia má byť spustiteľná aj na mobilnom zariadení, tablete a desktope a iných zariadení s operačným systémom Windows 10.

Aplikácia spustená na tablete vstavanom v menze má mať odlišné správanie ako aplikácia spustená na tablete používateľa. V aplikácií využívanej v menze musí byť nastavený nekonečný životný cyklus aplikácie. Prihlasovanie má fungovať na základe priloženia stravovacej karty k čítačke kariet. V aplikácií, ktorú používateľ spustí na vlastnom tablete, sa musí prihlásiť cez prihlasovacie meno a heslo a musí byť schopný aplikáciu vypnúť.

Aplikácia má kontrolovať prihlásenie používateľa na určitú dobu, podľa nastavení prevádzkovateľa. Medzi ďalšie špecifikácie patrí zobrazovanie zoznamu jedál (pri výbere jedla možnosť objednať), zobrazenie zoznamu objednaných jedál a zoznam histórie účtu, t.j. zoznam jedál, ktoré boli objednané a zároveň aj vydané používateľovi a iné akcie nad účtom ako napríklad prevod peňazí. V zozname objednaných jedál po vybratí konkrétnej objednávky by mala byť k dispozícií možnosť odhlásenia jedla. V zozname histórií by mal byť k dispozícií detailnejší popis jednotlivých akcií. K dispozícií by mal byť výber medzi rôznymi platnými výdajňami, zobrazenie informácie o používateľovi, ako sú meno a stav účtu, prípadne disponibilný zostatok na účte podľa nastavení prevádzkovateľa. Správanie aplikácie by mala byť závislá na systémových nastaveniach. Napríklad niektoré menzy zobrazujú počet zvyšných jedál na objednanie, zobrazenie informácie, že jedlo na objednávku je z burzy. V popise objednávky by mal byť popis jedla, z ktorej výdajne je objednávka, dátum, cena, zobrazenie obrázku jedla, zloženie jedla, prípadne aj detailnejšie informácie o jedle.

Aplikácia musí byť správne navrhnutá pre reálne využitie, ako napríklad možnosť odhlásenia jedla nesmie vzniknúť určitý počet dní pred výdajom jedla, definovaný prevádzkovateľom, rovnako tak aplikácia nesmie dovoľovať objednať jedlo po určitej dobe pred vydaním. Musí reagovať na všetky možné stavy, ako napríklad pre niektorý dátum nemusí byť vytvorená ponuka jedál. Aplikácia by mala byť efektívne graficky navrhnutá na všetky typy zariadení. Okrem toho, aby aplikácia tieto požiadavky spĺňala, musí byť pre používateľa intuitívne ovládateľná a prehľadná.

5. Analýza požiadaviek

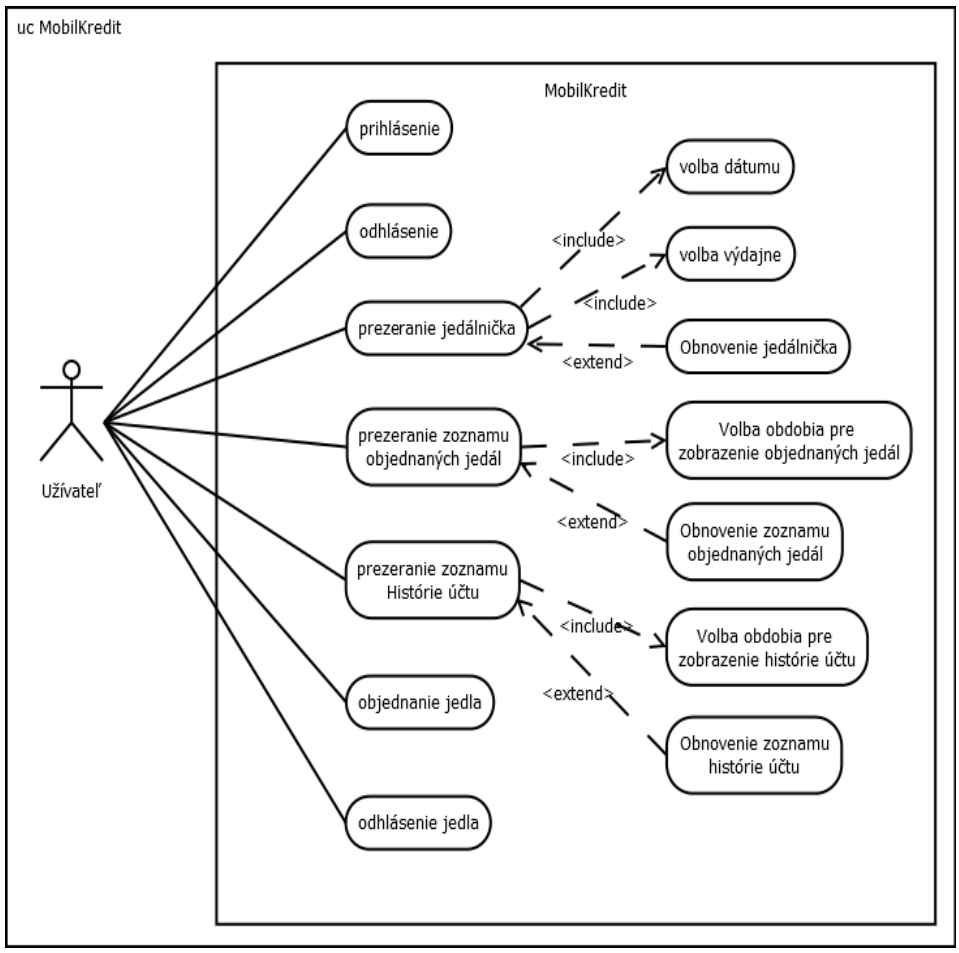
Pri tvorbe aplikácie budeme potrebovať rozdeliť aplikácie na viac modulov, na seba nadväzujúcich. Aplikácia bude rozdelená na 3 hlavné komponenty:

- Model
- View
- ViewModel.

Model bude tvorený dvoma triedami. Jedna trieda bude slúžiť na vytváranie objektov reprezentujúcich používateľa aplikácie, v ktorej budú informácie o používateľovi, pri odhlásení sa zruší objekt a pri prihlásení ďalšieho používateľa sa vytvorí nový tejto triedy. Druhá trieda bude vytvorená podľa návrhového vzoru Singleton, ktorá obsahuje zoznam volaní na aplikačný server.

View bude obsahovať štruktúru a prvky GUI aplikácie. GUI by malo byť pre používateľa jednoduché na ovládanie a intuitívne.

ViewModel je prostredník medzi View a Modelom, ktoré spája a nastavuje pravidlá zobrazovania dát do View získaných z Modelu. K týmto trom častiam patria aj Unit testy ktoré budú prevažne testovať metódy ViewMode-u. Na nasledujúcom obrázku je znázornený diagram prípadov použitia.



Obrázok 12: Diagram prípadov použitia

6. Návrh aplikácie

Táto kapitola nadväzuje na prechádzajúcu kapitolu a má za úlohu špecifikovať návrh aplikácie MobilKredit, návrh architektúry, návrh zobrazenia informácií, komunikačné protokoly, návrh užívateľského rozhrania. Pri tvorbe aplikácie bola využitá klient – server architektúra, ktorá je detailnejšie popísaná v nasledujúcej podkapitole.

6.1 Klient - server architektúra

Aplikácia ako je MobilKredit vyžaduje vykonávanie časovo a pamäťovo náročných operácií mimo aplikácie a ich výsledky následne posielať a prezentovať v aplikácií. Z toho dôvodu bola zvolená architektúra Klient – server.

Model Klient – server je forma distribuovaného spracovania, kedy klient komunikuje so serverom a odovzdávajú si informácie. Klient prostredníctvom užívateľského rozhrania komunikuje s používateľom, prekladá používateľovu požiadavku tak, aby bol zrozumiteľná serveru, čaká na odpoveď od serveru a odpoveď prekladá späť tak, aby bola zrozumiteľná používateľovi a prezentuje ju cez užívateľské rozhranie. Server čaká na požiadavku od klienta, potom ju spracuje a odpoveď pošle klientovi.

Najznámejšie sú dva typy Klient – server architektúr, dvojvrstvová a trojvrstvová. Dvojvrstvová zabezpečuje sieťovú komunikáciu medzi dvomi komponentmi, a to je klient (prezentačná vrstva) a server (dátová vrstva). Klient teda obsahuje užívateľské rozhranie a aplikačnú logiku a server obsahuje relačnú databázu. Trojvrstvová architektúra obsahuje jednu komponentu navyše a tou je aplikačný server (aplikačná vrstva), v ktorej sa nachádza aplikačná logika. Aplikácia MobilKredit bude obsahovať trojvrstvovú Klient – server architektúru:

Prezentačná vrstva

Je časť, ktorá je viditeľná pre používateľov, zaisťuje vstup požiadaviek a prezentáciu výsledkov. Je závislá na platforme (napr. webová aplikácia, Aplikácia pre Windows, Android aplikácie a iné). Môže byť teda rôzna pre rôzne zariadenia alebo platformu. Prezentačnú vrstvu v tomto projekte reprezentuje klientska aplikácia MobilKredit.

Aplikačná vrstva (funkčná)

Prostredná vrstva modelu, zabezpečuje výpočty a operácie vykonávané medzi vstupno-výstupnými požiadavkami a dátami. Obsahuje aplikačnú logiku. Tiež nazývaná ako aplikačný server. Aplikačnú vrstvu v tomto projekte reprezentuje aplikačný server, ktorý je implementovaný na serveroch, od ktorých bude aplikácia MobilKredit žiadať dáta.

Dátová vrstva (databázová)

Najnižšia vrstva modelu, zaisťuje prácu s dátami, teda so systémom riadenia bázy dát (SRBD) a základné dátovo-funkčné operácie zaisťujúce ukladanie, výber, agregáciu, predspracovanie, integritu a audit dát. Dátová vrstva tohto projektu sa rovnako ako aplikačná vrstva nachádza na serveroch v podobe úložiska dát.

6.2 Komunikačný protokol

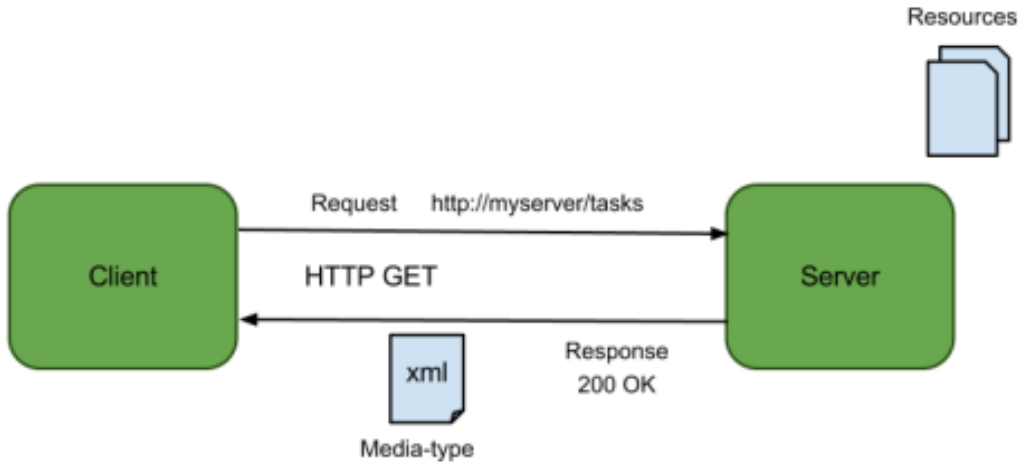
Pre správny návrh aplikácie musí byť taktiež správne navrhnutý komunikačný protokol medzi klientom a serverom. Aplikácie sú zväčša postavené tak, aby pristupovali k vzdialeným službám, ktoré poskytujú dáta vyžiadané od používateľa. Tieto webové služby bývajú zvyčajne založené na SOAP (Simple Object Access Protocol) alebo REST (Representational State Transfer). Pre tento projekt bola zvolená webová služba REST.

REST

Je architektonický štýl umožňujúci jednoducho vytvoriť, čítať, editovať alebo mazať informácie zo serveru pomocou jednoduchých HTTP volaní. Táto architektúra je navrhnutá pre distribuované prostredie. REST na rozdiel od SOAP nie je orientovaný procedurálne, ale dátovo. Webové služby definujú vzdialené procedúry a protokol pre ich volania, REST určuje, ako sa pristupuje k dátam.

Rozhranie REST je použiteľné pre jednotný a ľahký prístup k zdrojom (resources). Zdroje môžu byť dáta, tak ako aj stavy aplikácie (ak sa dajú popísať konkrétnymi dátami). Všetky zdroje majú vlastný identifikátor URI a REST definuje štyri základné metódy pre prístup k nim, známe pod pojmom CRUD. To je vytváranie dát (Create), získavanie požadovaných dát (Retrieve), editácia (Update) a mazanie (Delete). Tieto metódy sú implementované pomocou odpovedajúcich metód HTTP protokolu, ktoré sú POST (Create) GET (Retrieve), PUT (Update) DELETE (Delete).

Klientská aplikácia MobilKredit bude využívať REST API pre zabezpečenie komunikácie medzi serverom firmy, ktorá poskytuje prístup k potrebným dátam, ktoré má klient spracovať a následne aplikácia MobilKredit prezentovať používateľovy. Aplikácia MobilKredit začne komunikáciu a pristúpi k zdroju na serveri použitím unikátnej adresy (URI). Server následne vráti reprezentáciu zdroja vo formáte na základe požiadavky. Aplikácia MobilKredit využíva JSON.



Obrázok 13: Architektúra REST [15]

6.3 Návrh zobrazenia informácií

Návrh štruktúry aplikácie vychádza zo špecifikácie požiadaviek obsiahnutej v kapitole 4 (zber požiadaviek). Celá aplikácie bude rozdelená do stránok (Page). Medzi jednotlivými stránkami sa bude dať prepínať cez hlavné menu. Hlavné menu bude obsahovať stránku pre zobrazenie jedálnička, stránku pre zoznam objednaných jedál, zoznam histórie účtu a stránku, ktorá bude slúžiť na prihlásenie a odhlásenie používateľa. Bližšia definícia stránok:

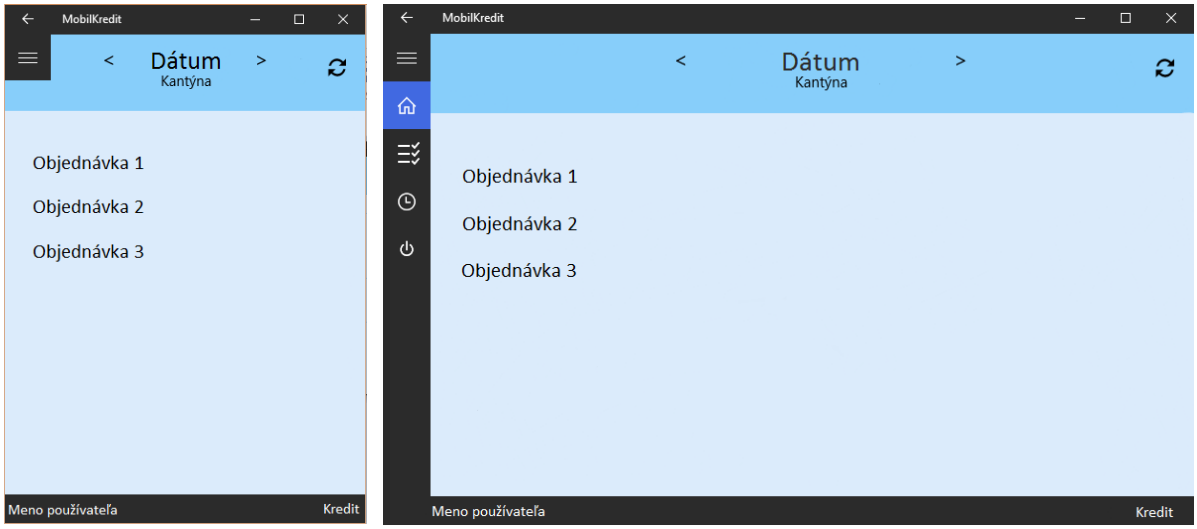
- **Jedálniček –** stránka, ktorá bude v aplikácií zobrazovať jedálny lístok v zadanom dátume a výdajni. Táto stránka bude zobrazovať jednotlivé jedlá zjednotené do skupín ako je napríklad obed, večera, pizza, týždenná ponuka atď. Po zvolení konkrétneho jedla sa zobrazí okno s detailnejšími informáciami o jedla ako je zloženie jedla. V tomto okne bude k dispozícií tlačítko pre objednanie jedla alebo zrušenie okna. Po objednaní jedla sa otvorí nové okno s detailnejšími informáciami o objednanom jedle ako aj čas objednávky. Stránka Jedálniček bude obsahovať tlačítko pre výber výdajne, ktoré zobrazí okno s možnosťou výberu výdajne a tlačítko pre obnovenie ponuky pre vybratý deň a výdajňu. Táto stránku bude taktiež umožňovať prepínanie medzi dátumami pre zobrazenie zoznamu jedál.
- **Objednané jedlá –** stránka, ktorá bude zobrazovať zoznam objednaných jedál a podobne ako v stránke Jedálniček bude obsahovať jednotlivé objednávky zjednotené do skupín podľa dátumu objednávky. Tak isto obsahuje tlačítko pre obnovenie zoznamu a stránku umožňuje prepínať medzi časovým obdobím zobrazených objednaných jedál. Po zvolení konkrétnej objednávky sa zobrazí okno s bližšími informáciami o objednanom jedle a bude obsahovať tlačítko pre odhlásenie objednávky.
- **História účtu –** stránka, ktorá bude zobrazovať zoznam histórie účtu a podobne ako v stránke Objednané jedlá bude obsahovať jednotlivé akcie z histórie účtu zjednotené do skupín podľa dátumu prevedenia akcie. Tak isto obsahuje tlačítko pre obnovenie zoznamu a stránka umožňuje prepínať medzi časovým obdobím zobrazených akcií z histórie účtu. Po zvolení konkrétnej akcie sa zobrazí okno s bližšími informáciami o akcii.
- **Prihlásenie/Odhlásenie –** stránka, ktorá slúži pre prihlásenie a odhlásenie. Má viac variant. Prvá je pre aplikácia dostupná pre verejnosť na stiahnutie, v ktorej používateľ zadá svoj login a heslo, prípadne zvolí prevádzkovateľa. Druhá varianta je pre tablet vstavaný v menzách, kde na stránke Prihlásenie/Odhlásenie bude uvítacia obrazovka a prihlásenie sa uskutočňuje cez čítačku kariet, prostredníctvom karty používateľa.

Okrem spomínaných stránok a hlavného menu bude aplikácia obsahovať informácie o používateľovi, ktorý bude prihlásený. Je to meno používateľa a jeho stav na účte. V prípade, že používateľ nebude prihlásený do aplikácie, nebude možné prepínať medzi stránkami. Aplikácia bude taktiež kontrolovať pripojenie k serveru. V prípade, že vypršal časový limit pre pripojenie na server, alebo server prestal komunikovať s aplikáciou, používateľ je o tom informovaný prostredníctvom správy, ktorá ho presmeruje na prihlasovaciu stránku.

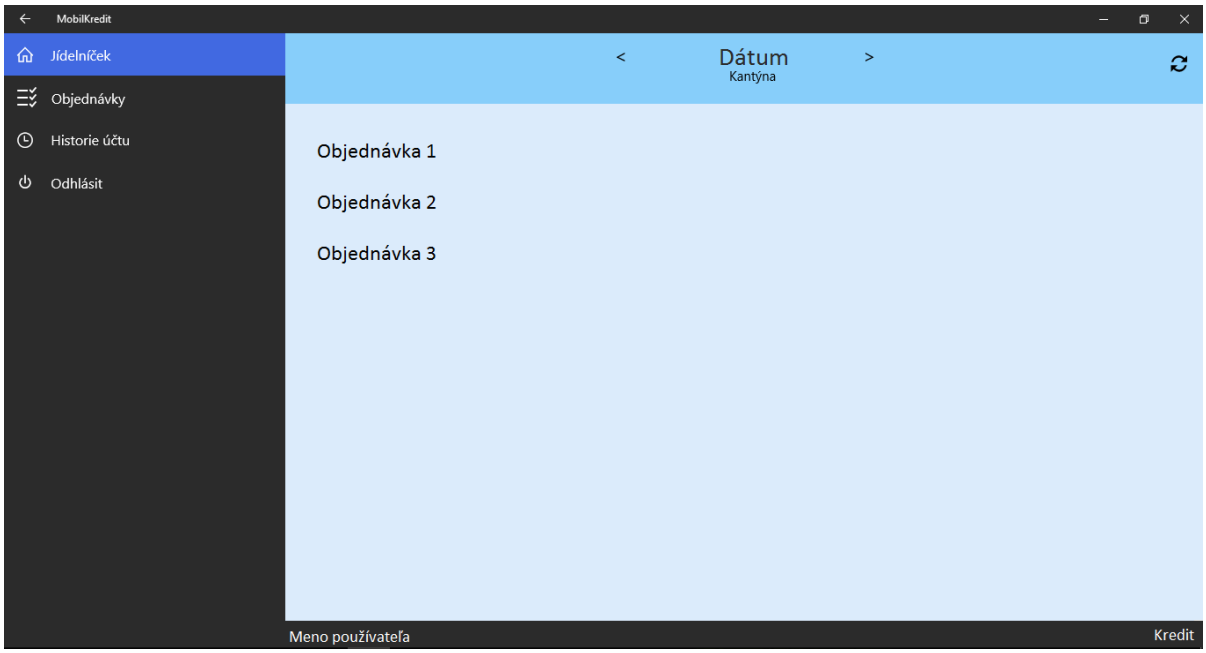
6.4 Návrh GUI

Pri návrhu grafického užívateľského rozhrania bolo potrebné zohľadniť niekoľko vlastností výslednej aplikácie MobilKredit, ktoré boli bližšie definované v kapitole 4 (Zber požiadaviek). Aplikácia má byť spustiteľná na rôznych typoch zariadení, ako je mobilné zariadenie, tablet, stolný počítač, prípadne na veľkoplošných LCD monitoroch. Preto bude potrebné vytvoriť hlavné menu vo forme navigačnej lišty prispôsobujúce sa výslednému zariadeniu podľa šírky a výšky displeja.

Na nasledujúcich obrázkoch je možné vidieť prvotný návrh grafického rozhrania aplikácie MobilKredit pre rôzne typy zariadení. Je možné vidieť prispôsobenie hlavného menu na rôzne typy zariadení. Rozhranie je navrhnuté tak, aby bolo pre používateľa prehľadné, intuitívne a poskytovalo rýchly a jednoduchý prístup k všetkým funkciám aplikácie MobilKredit.



Obrázok 14: návrh aplikácie pre mobilné zariadenie a tablet



Obrázok 15: návrh aplikácie pre stolný počítač

7. Implementácia

Táto kapitola bližšie špecifikuje technológie, ktoré sú potrebné pre implementáciu aplikácie MobilKredit, rovnako ako aj popis problémov pri vývoji, štruktúru aplikácie, implementáciu jednotlivých tried. V tejto kapitole je rovnako zahrnuté aj možné rozšírenie aplikácie.

7.1 Použité technológie

Pri tvorbe Windows aplikácií sa využívajú nástroje, zamerané na uľahčenie vývoja, testovania a ladenia aplikácie. Pre vývoj aplikácie MobilKredit bol zvolený.Net Framework s jazykom C# a vývojové prostredie Visual Studio. Tieto technológie sú ďalej popísané v nasledujúcich podkapitolách ako aj doplnky k uľahčeniu vývoja aplikácie.

7.1.1 Jazyk C#

Okrem základných objektovo orientovaných princípov C# zjednodušuje vývoj softwarových komponent vďaka niekoľkým inovačným jazykovým konštrukciám, ako sú:

- Zapuzdrené podpisy metód nazývaných delegáti, ktoré povoľujú typovo bezpečné oznámenia o udalostiach.
- Vlastnosti, ktoré slúžia ako prístupové objekty pre premenné súkromných členov.
- Atribúty, ktoré poskytujú deklaratívne metadáta o typoch za behu.
- Dokumentačné komentáre XML
- LINQ (Language-Integrated Query), ktorý poskytuje preddefinované možnosti rôznych dátových zdrojov.

Proces zostavenia C# je jednoduchší, ako v jazykoch C a C++ a flexibilnejší ako v jazyku Java. Neexistujú žiadne samostatné hlavičkové súbory a žiadna požiadavka, aby metódy a typy boli deklarované v určitom poradí. Zdrojový súbor C# môže definovať ľubovoľný počet tried, štruktúr, rozhraní a udalostí.

7.1.2 Vývojové prostredie

Pre tvorbu aplikácie je dôležitý výber vhodného vývojového prostredia, ktoré dokáže zjednodušiť prácu na vývoji aplikácie, ladení a testovaní.Vývojové prostredie Visual Studio 2015 umožňuje vývoj aplikácií pre rôzne zariadenia, aplikácie pre pracovné plochy, pre web a v cloude. Taktiež poskytuje IDE pre tvorbu aplikácie pre iOS, Android a Windows. Podporuje tvorbu hier pre viac platforiem v jazyku C# pomocou Unity, tvorbu univerzálnych aplikácií pre všetky zariadenia s operačným systémom Windows 10. Umožňuje zobrazovať mapy kódu a grafy závislostí. Nástroje pre ladenie a diagnostiku. Microsoft Blend pre Visual Studio je ďalšou pomôckou pre vývojára pri vytváraní UI, je to nástroj pre tvorbu grafického rozhrania. Obsahuje veľa ďalších technológií, ktoré uľahčujú vývoj v tomto vývojovom prostredí. Jednými z nich sú Resharper a Code metrices, ktoré boli využité pri tvorbe tejto práce a budú detailnejšie popísané v nasledujúcich podkapitolách.

7.1.3 Resharper

Táto podkapitola je zameraná na výhody a uľahčenie vývoja pomocou nástroju Resharper pre vývojové prostredie Visual Studio. Vlastnosti tohto nástroja sú rozdelené do častí:

- **Analýza kvality kódu –** analýza kvality kódu je k dispozícií v jazyku C#, VB.NET, XAML, ASP.NET, ASP.NET MVC, JavaScript, TypeScript, CSS, HTML a XML. Oznamuje v prípade, že sa kód môže vylepšiť a ponúkne možnosti pre zmenu.
- **Eliminácia chýb v kóde –** Resharper nielen dokáže varovať vývojára, keď je chyba v kóde, ale poskytuje stovky rýchlych opráv pre automatické riešenie problému. Vo väčšine prípadov je možné si vybrať najlepšiu rýchlu opravu z rôznych možností.
- **Bezpečne meniť základ kódu (Refactoring)** – Refactoring umožňuje bezpečne zmeniť základ kódu, či už je potrebné obnoviť starý kód alebo dodržiavať štruktúru aplikácie v poradí.
- **Okamžité prechádzanie celého riešenia (Solution) –** Je možné okamžite prechádzať a vyhľadávať v celom Solution. Skočiť na akýkoľvek súbor, typ alebo typový člen, alebo prejsť z určitého symbolu k jeho použitiam, odvodeným symbolom alebo implementáciám.

7.1.4 Code Metrics

Nástroj pre vývojové prostredie Visual Studio je sada softwarových meraní, ktoré poskytujú lepší pohľad na kód pre vývojárov. Vďaka tomuto nástroju vývojári vidia, ktoré časti kódu by mali prepracovať, alebo dôkladnejšie otestovať. Code Metrics umožňuje identifikovať možné riziká. K jednotlivým metódam a iným častiam kódu Code Metrics priradí známku, ktorá ukazuje zložitosť daného úseku kódu. Čím väčšia známka tým väčšie riziko a potrebné adekvátne otestovanie. Medzi hlavné merania patria Index Udržateľnosti, cyklomatická zložitosť, hĺbka dedičnosti, spájanie tried, počet riadkov kódu.

7.1.5 GitHub

GitHub je webová služba podporujúca vývoj softwaru pomocou verzovacieho nástroja Git. GitHub poskytuje bezplatný webhosting pre open source projekty. Súkromné repozitáre sú k dispozícií po zaplatení mesačného poplatku. GitHub poskytuje ďalšie služby ako Gist, ktorý je súčasťou GitHubu, umožňuje verzovanie a rýchle zdieľanie kratšieho kódu.

Nástroj Git bol využitý pri tvorbe tejto práce. Zabezpečoval predovšetkým ukladanie zmien práce, jednotlivé zmeny boli prístupné v zozname histórie zmien, kde bolo možné prechádzať medzi jednotlivými zmenami, ako aj ich obnovenie. Nástroj poskytoval zdieľanie práce s firmou, pre ktorú bola práca vyhotovená. Zdieľanie tak pomohlo pri rýchlosti vývoja aplikácie, ktoré zabezpečovalo rýchlu odozvu od firmy v prípade porušenia špecifikácie, ktorá nebola zahrnutá vo všeobecnej špecifikácií, ktorá je popísaná v kapitole 4 (Zber požiadaviek).

7.2 Aplikácia

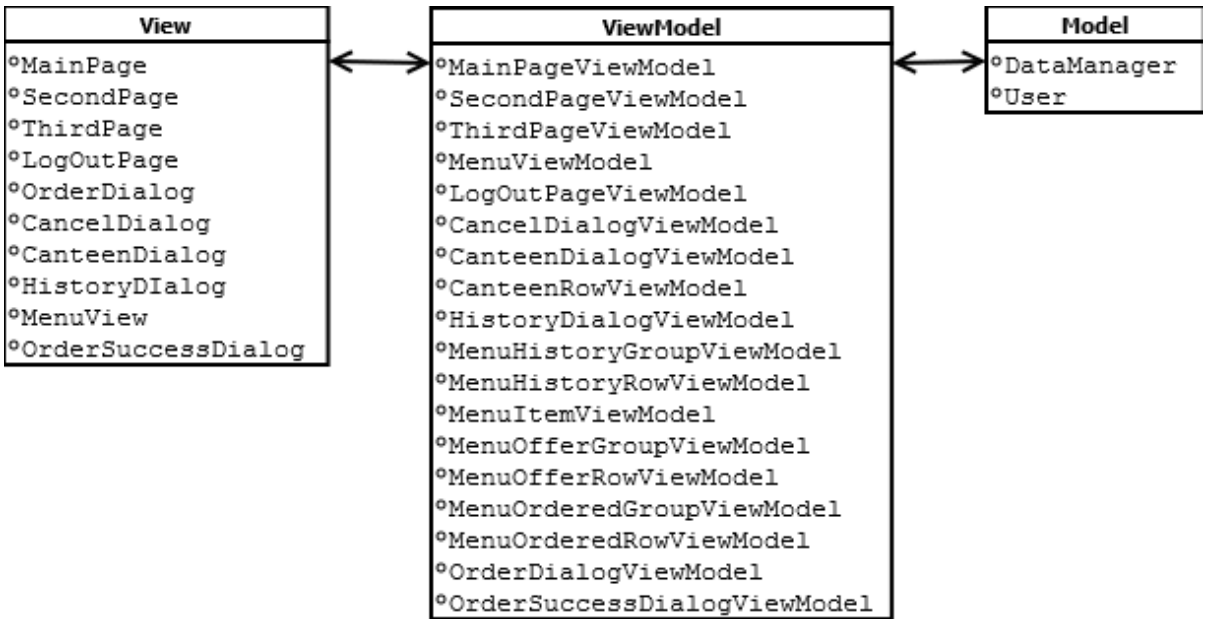
Úlohou tejto podkapitoly je znázorniť vnútorný pohľad na vývoj aplikácie, ktorým je jej základná vnútorná štruktúra, použitie jednotlivých udalostí, návrhových vzorov a celkový vývoj dizajnu aplikácie a podobne.

7.2.1 Štruktúra aplikácie

Zdrojové kódy aplikácie operačného systému Windows sú usporiadané do stromovej štruktúry adresárov. Koreňom aplikácie je Solution, ktorý môže obsahovať viac projektov. Tieto projekty obsahujú adresáre, ktoré obsahujú zdrojové kódy aplikácie. Adresáre sa môžu vnorovať. Aplikácia MobilKredit sa skladá z 3 projektov obsiahnutých v jednom Solution. Celé rozdelenie aplikácie na jednotlivé adresáre a ich popis je uvedený v nasledujúcich častiach podkapitoly:

- **Solution 'Ordering' –** Solution aplikácie, ktorý obsahuje 3 projekty.
  - **Ordering.Client –** Projekt, ktorý obsahuje zdrojové kódy aplikácie MobilKredit
    - **Properties –** Základné informácie o assembly projektu (AssemblyInfo).
    - **References –** Referencie na ďalšie projekty, prípadne knižnice (Ordering.Utils, Ordering.Domain a ďalšie).
    - **Events –** V tomto adresári sa nachádzajú udalosti pre načítanie karty na čítačku (CardReadEvent, CardReadEventArgs).V tejto verzií je to postupnosť stlačených kláves po sebe s časovým limitom, tento postup modeluje priloženie karty.
    - **Services –** Adresár obsahuje servis pre čítanie karty (CardReader, CardReaderState, ICardReader) a pre prideľovanie View dialógového okna k patričnému ViewModel-u (DialogService).
    - **Strings –** Obsahuje podadresáre označujúce jazyk v ktorom majú byť zobrazované časti aplikácie (en, cz, sk). Podadresáre obsahujú reťazce využívané v aplikácií.
    - **Themes –** Obsahuje grafické témy aplikácie. V tomto adresári sa nachádza XAML súbor v ktorom je definovaný dizajn celej aplikácie (SplitViewSampleStyle). Aplikácie štandardne obsahujú viac štýlov a jednoducho sa medzi nimi prepína.
    - **Models –** Prvá časť návrhového vzoru MVVM, v ktorej sa nachádzajú Modely aplikácie. V prípade tejto práce ide o dáta manažér a prihlásený používateľ (User). Modely budú detailnejšie popísané v nasledujúcich podkapitolách.
    - **ViewModels –** Druhá časť návrhového vzoru MVVM, v ktorej sa nachádzajú ViewModel-y aplikácie. Prostredník medzi View a Model-om. Zdrojové kódy ViewModel-u prijímajú dáta od Modelu a zobrazujú cez View.
    - **Views –** Tretia časť návrhového vzoru MVVM, v ktorej sa nachádzajú View-y aplikácie. Definujú grafický dizajn zobrazenia jednotlivých častí aplikácie. Dáta prijímajú od ViewModel-u.
    - **AppShell -** trieda obsahujúca základný rám aplikácie.
    - **PageTokens –** trieda, v ktorej sú konštanty názvov stránok, využívané pri navigácií medzi stránkami.
  - **Ordering.Domain –** Projekt, v ktorom sa nachádzajú volania na server. Obsahuje Properties a References adresáre, v ktorých sa nachádzajú podobne ako v predchádzajúcom projekte assembly projektu a referencie.
    - **Models** – Definujú dátové triedy využívané pri volaniach na server.
    - **Services** – Definujú metódy pre komunikáciu so serverom.

Hlavná časť aplikácie MobilKredit, ktorá tvorí návrhový vzor MVVM je znázornená na nasledujúcom obrázku. Nachádza sa v projekte Ordering.Client.



Obrázok 16: Rozdelenie aplikácie podľa MVVM

V ďalších podkapitolách sa nachádza detailnejší popis častí MVVM vzoru projektu Ordering.Client.

7.2.2 Model

Model obsahuje 2 triedy, dáta manažer (DataManager) a prihlásený používateľ (User). Prijíma dáta z aplikačného servera cez metódy v triede DataManager a poskytuje ich ViewModel-om.

DataManager

DataManager je trieda, ktorá je implementovaná ako návrhový vzor Singleton. Obsahuje privátny konštruktor a statickú property, ktorá inicializuje inštanciu tejto triedy. Okrem toho obsahuje zoznam asynchrónnych metód, ktoré reprezentujú jednotlivé volania na server. Metódy obsahujú blok try - catch, ktorý zabezpečuje oznámenie používateľovi o ukončení spojenia so serverom, alebo pri chybnej komunikácií so serverom. V prípade, že blok try – catch zachytí výnimku, zavolá sa privátna metóda OpenMessageLogout(), v ktorej je používateľ informovaný o ukončení spojenia so serverom pomocou dialógového okna. Okno obsahuje jedno tlačítko, ktoré používateľa následne presmeruje na prihlasovaciu stránku aplikácie prostredníctvom NavigationService.

Trieda DataManager ďalej obsahuje property typu User, ktorá sa nastavuje pri prihlásení a odhlásení používateľa do aplikácie. Pri prihlásení sa property naplní dátami definovanými v triede User a pri odhlásení obsahuje null. Trieda obsahuje privátnu premennú proxy typu IOrderingProxy, ktorá sprostredkováva služby servera a privátnu premennú navigationService typu INavigationService, ktorá sprostredkováva navigáciu medzi stránkami aplikácie.

User

Trieda reprezentujúca prihláseného alebo odhláseného používateľa. V stave odhlásenia obsahuje hodnotu null a v stave prihlásenia naplní premenné v konštruktore.

7.2.3 ViewModel

Časť MVVM vzoru, ktorá v tejto práci preberá parametre z modelu a nastavuje ich jednotlivým property ViewModel-ov, ktoré sa volajú vo View. ViewModel zabezpečuje aktualizáciu Modelu na základe interaktivity používateľa s aplikáciou.

Aplikácia MobilKredit obsahuje ViewModel-y pre hlavné stránky aplikácie (Jedálniček, Objednané jedlá, História účtu a Prihlásenie/Odhlásenie), pre hlavné menu, v ktorom sa nachádzajú stránky, pre každé dialógové okno v aplikácií. Okrem týchto tried obsahuje ViewModel-y pre skupiny jedál a dátumov (napríklad MenuHistoryGroupViewModel) a ViewModel-y pre jednotlivé položky skupiny (napríklad MenuOfferRowViewModel).

Podľa špecifikácie aplikácie, bolo potrebné vytvoriť zoznam jedál, ktorý by obsahoval skupiny ako sú obed, raňajky a podobne. Do týchto skupín by boli pridelené jedlá na základe parametrov dotyčného jedla. Pri tvorbe tohto zoznamu bolo potrebné vytvoriť triedu MenuOfferGroupViewModel, ktorá obsahovala kolekciu objektov triedy MenuOfferRowViewModel (trieda reprezentujúca konkrétne jedlo pre objednávku) a property MealKindText (reťazec typu jedla, napríklad obed). Trieda MenuOfferRowViewModel implementuje kontroly zobrazenia informácií (jedlo je z burzy, počet zvyšných jedál, jedlo nie je možné objednať) a definuje property pre dané jedlo.

Okrem spomínaných tried, do ViewModel-u patria triedy definované v Services. Jedná sa o služby pre načítanie karty, DialogService a DialogResult, ktoré sú opísané v ďalších podkapitolách. Triedy v adresári by podľa logického návrhu mali byť zahrnuté do časti Model v MVVM, no táto aplikácia je stavaná tak, že sú súćasťou ViewModel-u. Ide o udalosti pre zachytávanie načítania karty.

7.2.4 View

View je prezentačná časť MVVM vzoru, ktorá nastavuje grafický dizajn aplikácie staticky, ako aj pomocou Binding-u property, ktorá sa nachádza v príslušnom ViewModel-y daného View. Ku každému View tejto práce prislúcha práve jeden ViewModel. Komponenty View využívajú štýly zo statických zdrojov definovaných v adresári Themes v XAML súbore SplitViewSampleStyles.

Podľa pravidiel MVVM by View nemal obsahovať kód na pozadí. V tejto práci bolo potrebné vytvoriť kód na pozadí, a to konkrétne vo View dialógových oknách, v ktorých sa nachádza viac ako jedno tlačítko. Dôvodom bolo notifikovať stránku, z ktorej bolo okno spustené o konkrétnom tlačítku, ktoré bolo stlačené (napríklad jedno tlačítko je pre objednanie jedla a druhé pre zrušenie dialógového okna). Detailnejšie informácie sa nachádzajú v nasledujúcej podkapitole.

7.2.5 DialogService

Prideľovanie ViewModel-ov k View jednotlivým stránkam (Page) zabezpečuje PRISM framework. Problém nastal pri spájaní View a ViewModel-u dialógových okien. Bolo potrebné vytvoriť triedu DialogService, ktorá mala na starosti spájaciu službu a vytvorenie View. Jej metóda CreateView() slúži na vytvorenie View do kontajnera IUnityContainer, ktorý predstavuje návrhový vzor Dependency Injection (DI). Trieda implementuje dve metódy:

ShowAsync()

Vytvára View k danému ViewModel-u, pomocou privátnej metódy CreateView(). View je typu ContentDialog, nad ktorým zavolá funkciu ShowAsync(), čím sa zobrazí dialógové okno. Po jeho ukončení sa okno zavrie a aplikácia sa vráti na stránku, z ktorého bolo okno spustené. Metódou ShowAsync() sú spustené dialógové okná ako napríklad: HistoryDialog, OrderSuccessDialog.

ShowAsyncWithResult()

Rovnaká funkcionalita ako u metódy ShowAsync() rozšírená o vrátenie hodnoty pôvodnej stránke, z ktorej bolo okno spustené. View je typu ContentDialog, ktorý navyše implementuje rozhranie ICustoMDialogResult() (popis v nasledujúcej podkapitole). Po pridelení ViewModel-u k View sa zavolá asynchrónna medóda ShowAsync() nad týmto View. Po jej ukončení (ukončenie dialógového okna) vráti Result stránke, z ktorého bolo spustené okno. Metódou ShowAsync() sú spustené dialógové okná ako napríklad: OrderDialog, CancelDialog, CanteenDialog. Popis výsledku, ktorý vráti dialógové okno je popísané v nasledujúcej podkapitole.

```
public class DialogService
{
    public DialogService(IUnityContainer container)
    {
        this.container = container;
    }

    private IUnityContainer container;

    public async Task<bool> ShowAsyncWithResult<TView>(INotifyPropertyChanged viewModel)
        where TView : ContentDialog, ICustomDialogResult
    {
        var view = CreateView<TView>();
        view.DataContext = viewModel;
        await view.ShowAsync();
        return view.Result;
    }

    public void ShowAsync<TView>(INotifyPropertyChanged viewModel)
        where TView : ContentDialog
    {
        var view = CreateView<TView>();
        view.DataContext = viewModel;
        view.ShowAsync();
    }

    private TView CreateView<TView>()
        where TView : ContentDialog
    {
        return container.Resolve<TView>();
    }
}
```

Obrázok 17: Trieda DialogService

7.2.6 DialogResult

Pri tvorbe dialógových okien, ktoré mali dve tlačítka pre rozdielnu funkčnosť, bolo potrebné vytvoriť rozhranie ICustomDialogResult, ktorý sa nachádza v časti ViewModel MVVM. Rozhranie obsahovalo jeden predpis property Result typu Bool. Triedy View, ktoré implementovali toto rozhranie tak definovali práve dve metódy, jedna pre jedno tlačítko s pridelením hodnoty true do property Result a druhá pre druhé tlačítko s hodnotou false property Result. Tlačítko zruší dialógové okno a stránke z ktorej bolo spustené okno následne vráti Result o informácií, ktoré tlačítko bolo stlačené. To zabezpečuje metóda ShowAsyncWithResult(), ktorá je opísaná v predchádzajúcej podkapitole. Na nasledujúcom obrázku je možné vidieť príklad View, ktoré obsahuje dve tlačítka. Konkrétne ide o dialógové okno, prostredníctvom ktorého sa dá objednať konkrétne jedlo.

```
public sealed partial class OrderDialog : ContentDialog, ICustomDialogResult
{
    public OrderDialog()
    {
        InitializeComponent();
    }
    private void Objednat_OnClick(object sender, RoutedEventArgs e)
    {
        var btn = sender as Button;
        var command = btn.Tag as ICommand;
        if (command != null) command.Execute(btn.CommandParameter);
        Result = true;
        Hide();
    }
    private void Zavriet_OnClick(object sender, RoutedEventArgs e)
    {
        var btn = sender as Button;
        var command = btn.Tag as ICommand;
        if (command != null) command.Execute(btn.CommandParameter);
        Result = false;
        Hide();
    }
    public bool Result { get; private set; }
}
```

Obrázok 18: trieda OrderDialog

7.2.7 NavigationService

Pre navigáciu vo frameworku PRISM existuje rozhranie INavigationService, ktoré vytvára navigačnú službu pre Windows Store aplikáciu. Štandardná implementácia tohto rozhrania je FrameNavigationService trieda, ktorá používa triedu, ktorá implementuje IFrameFacade, pre poskytnutie navigácie stránkam. Rozhranie obsahuje predpis metódy Navigate() s parametrom pageToken typu string. Pre jednoduchšiu editáciu kódu a pridávanie nových stránok preto bola vytvorená trieda PageTokens, ktorá bude obsahovať konštanty typu string jednotlivých stránok. Pri navigácií zo stránky na inú tak stačí použiť metódu Navigate() s parametrom konštanty cieľovej stránky. Ak navigácia uspeje, metóda Navigate() vráti hodnotu true, v opačnom prípade false. Na nasledujúcom obrázku je možné vidieť triedu PageTokens aplikácie MobilKredit.

```
public static class PageTokens
{
    public const string MainPage = "Main";
    public const string SecondPage = "Second";
    public const string ThirdPage = "Third";
    public const string LogOut = "LogOut";
}
```

Obrázok 19: Trieda PageTokens

Okrem metódy Navigate() sa v rozhraní INavigationService nachádzajú predpisy ako:

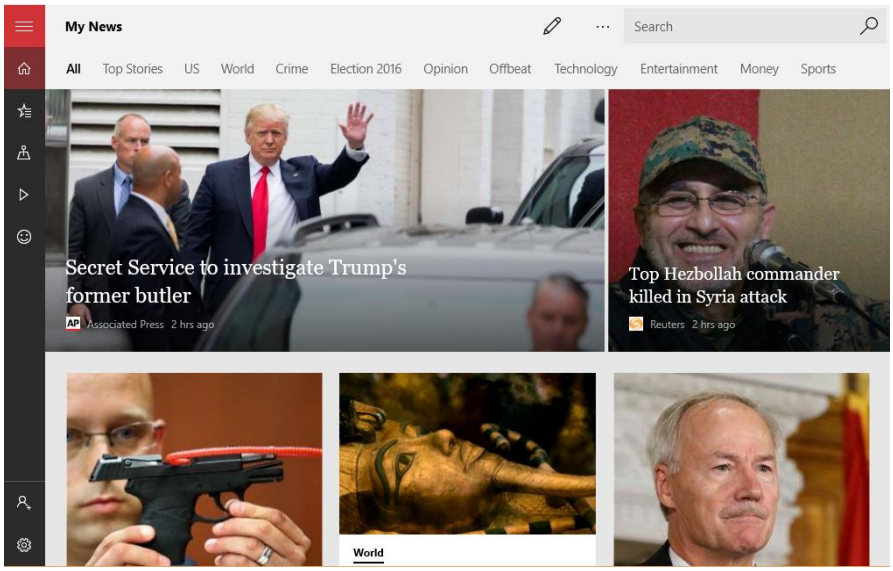
- GoBack() - vrátiť sa na predchádzajúcu stránku.
- CanGoBack() - metóda, pre zistenie či sa môže vrátiť na predchádzajúcu stránku.
- ClearHistory() - vymazanie histórie navigácie.
- RestoreSavedNavigation() – Obnovenie uloženej navigácie.
- Suspending() – Používa sa pre navigáciu od aktuálneho ViewModel-u, na základe udalosti Suspend. Týmto spôsobom je možné previezť ďalšiu logiku pre spracovanie tejto udalosti.

7.2.8 Štruktúra GUI

Graphical User Interface (GUI) je skratka pre používateľské grafické rozhranie. Umožňuje ovládanie aplikácie na základe interaktívnych prvkov ako sú tlačítka, zoznamy, formuláre a ďalšie.

Všetky prvky používateľského rozhrania operačného systému Windows sa nachádzajú v XAML súboroch. Staršiu technológiu WPF (Windows Presentation Foundation) pre GUI Windows aplikácií nahrádza UWP (Universal Windows Platform). V tejto práci je použitá technológia UWP. Celé používateľské rozhranie je obsiahnuté v jednej časti MVVM vzoru, vo View.

Inšpiráciou pri vytváraní dizajnu GUI bol trend dnešných Windows 10 aplikácií. Sú voľne dostupné pre operačný systém Windows 10. Ide o aplikácie Novinky, Počasie, Mapa od Microsoftu. Nasledujúci obrázok je príklad takejto aplikácie, konkrétne ide o aplikáciu News (Novinky).

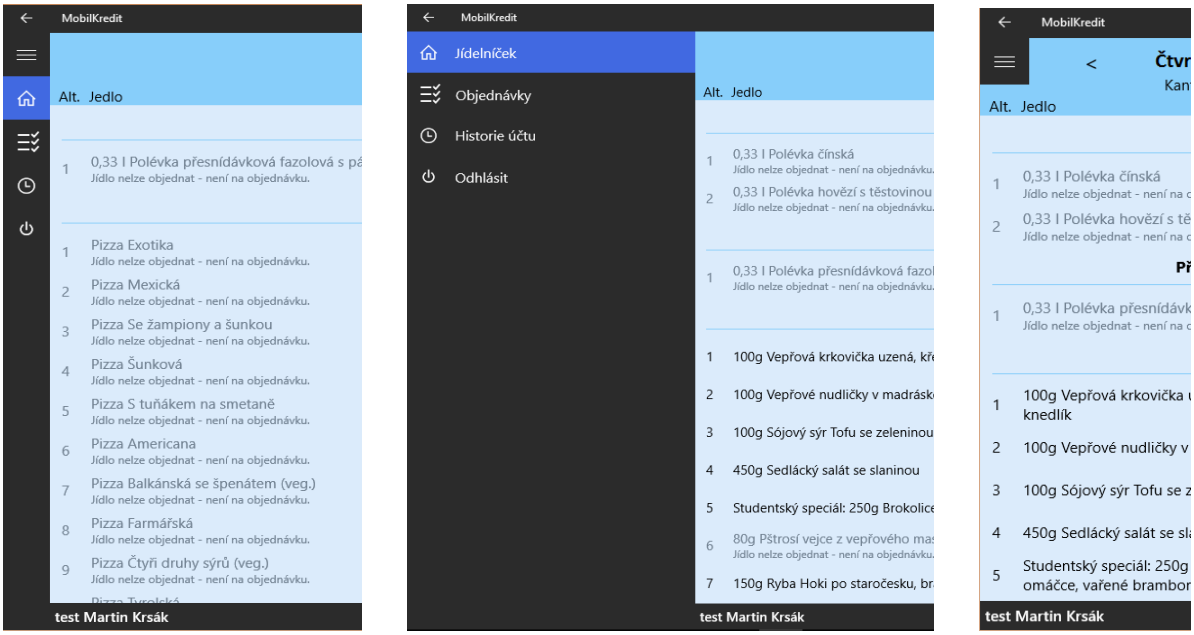


Obrázok 20: Aplikácia News od Microsoft-u

Hlavné menu

Hlavné menu aplikácie MobilKredit je navigačný prostriedok medzi stránkami. Grafické rozhranie je definované v AppShell, ktoré prispôsobuje zobrazenie informácií v hlavnom menu podľa šírky obrazovky v pixeloch. V XAML súbore MenuView časti View, je definované jedno tlačítko, ktoré slúži na prepínanie medzi stránkami. Toto View je prepojené s ViewModelom MenuItemViewModel, ktorý definuje property pre zobrazenie, ako sú napríklad DisplayName (Názov stránky), FontIcon (Ikonka stránky), Command (delegovanie na príslušnú metódu).

Hlavné menu je vytvárané v MenuViewModel, v ktorom sa vytvárajú jednotlivé tlačítka pre stránky a nastavujú pravidlá pre stav tlačítok ako Disabled, Enabled, pomocou privátnych premenných canNavigateToMain, canNavigateTioSecond. Na nasledujúcom obrázku je možné vidieť hlavné menu prispôsobené na rôznu šírku displeja.

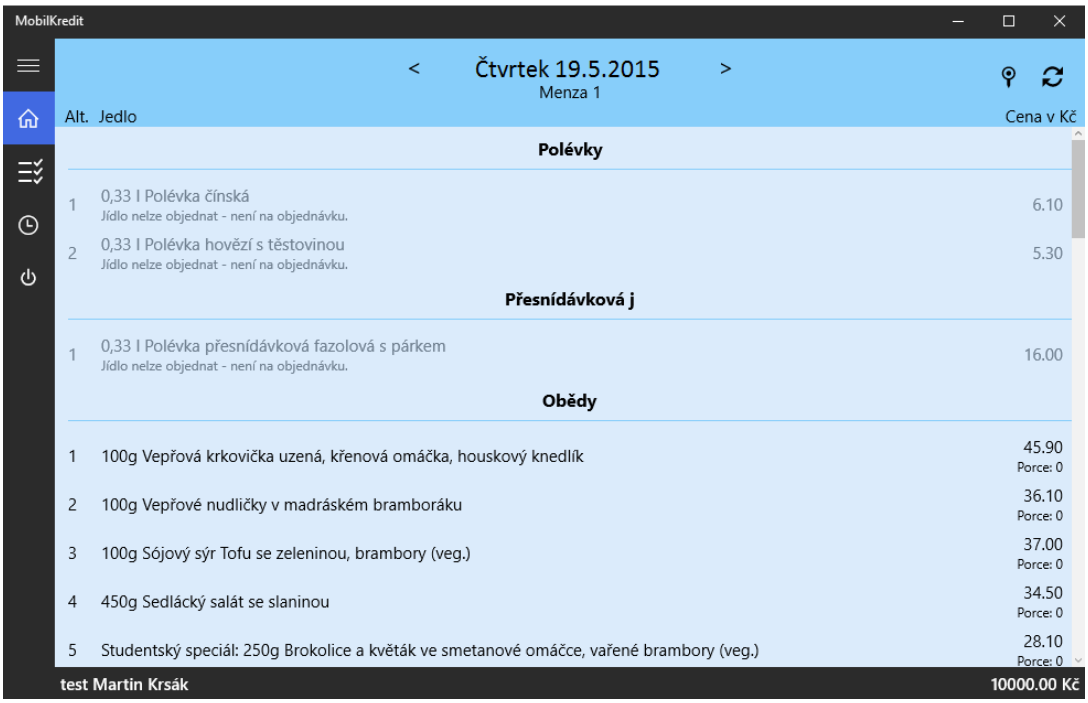


Obrázok 21: Hlavné menu prispôsobené mobilnému zariadeniu, tabletu, pracovnej ploche

V ďalších častiach budú opísané grafické prvky jednotlivých stránok, rozloženie a ich implementácia.

Jedálniček

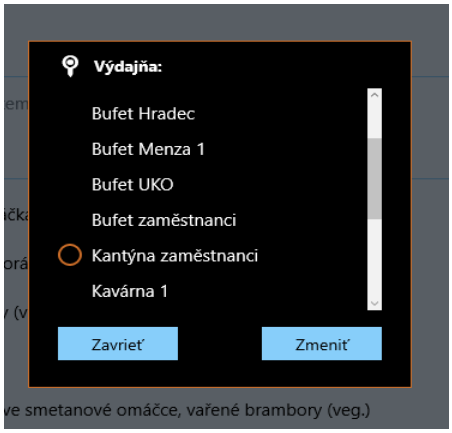
Implementácia grafického rozhrania stránky Jedálniček sa nachádza v XAML súbore MainPage časti View a je prepojená s MainPageViewModel. Definícia GUI tejto stránky je rozdelená vertikálne na 3 časti, tie sú rozdelené v grafickom prvku Grid. Na nasledujúcom obrázku je znázornená stránka Jedálniček.



Obrázok 22: Stránka Jedálniček

Prvá časť zhora sa skladá z informácie o vybranom dátum, tlačítok pre prepínanie medzi dátumami a tlačítok pre obnovenie stránky a navigáciu na dialógové okno pre výber výdajne. Ďalej sa tam nachádzajú nadpisy, ktoré označujú popis stĺpca v ďalšej časti (Alt., Jedlo, Cena v Kč).

Implementácia grafického rozhrania dialógového okna pre výber výdajne a jeho štruktúra je definované v XAML súbore CanteenDialog, ktorý je prepojený s CanteenDialogViewModel. Obsahuje zoznam grafických prvkov typu RadioButton, ktoré zobrazujú dostupné výdajne pre daný dátum. Obsahuje dve tlačítka, "Zavrieť", ktoré zavrie okno a "Zmeniť", ktoré nastaví používateľovi vybratú výdajňu, uzavrie toto dialógové okno a aktualizuje zobrazovanú výdajňu na stránke Jedálniček. Na nasledujúcom obrázku je znázornené dialógové okno Vybrať výdajňu.



Obrázok 23: Dialógové okno Výber výdajne

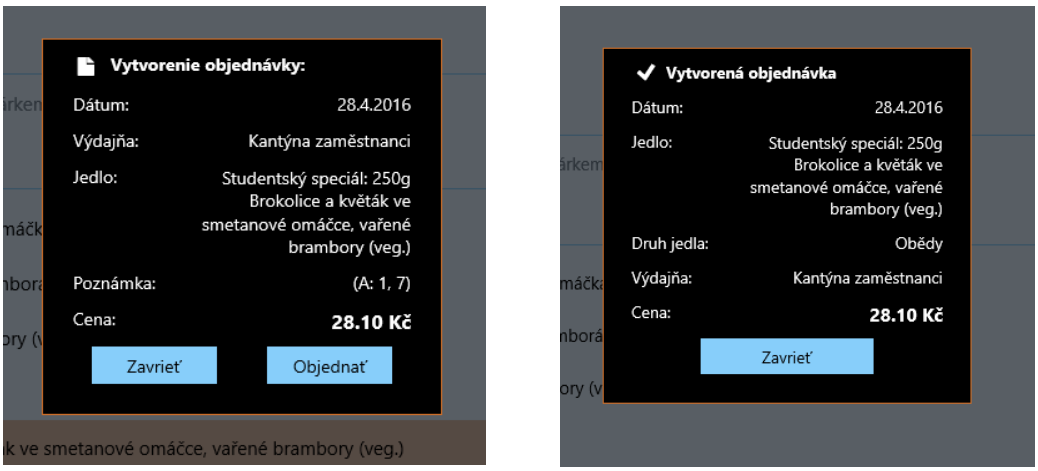
Druhá časť obsahuje grafický prvok ListView (zoznam), ktorý zobrazuje jednotlivé položky pre objednanie. ListView berie kolekciu od property DailyOfferView, ktorá je definovaná v MainPageViewModel. ListView vytvára skupiny, ktoré sú definované vo OfferGroupViewModel. Zobrazuje názov skupiny, ktorý sa nachádza v property MealKindText.

Políčko ListView je definované grafickým prvkom Grid, ktoré obsahuje ďalšie prvky typu TextBlock. Popisujú informácie o jedle ako napríklad názov jedla, číslo objednávky, informácia o dostupnosti jedla, počet porcií, cena. Celé políčko malo dva návrhy na implementáciu. Buď by políčko prezentoval grafický prvok Button alebo Grid. Grid som si vybral z dôvodu, aby štýl jednotlivých riadkov v ListView zostal rovnaký. Preto bolo potrebné nastaviť reakciu stlačenia políčka, tým že sa zavolá príkaz CmdOpenOrderingPopUp pomocou EventTriggerBehavior. Po kliknutí otvorí dialógové okno OrderDialog prostredníctvom príkazu (Command). Na nasledujúcom obrázku je ukážda implementácie udalosti pre stlačenie políčka.

```
<interactivity:Interaction.Behaviors>
    <core:EventTriggerBehavior EventName="Tapped">
        <core:InvokeCommandAction Command="{Binding CmdOpenOrderingPopUp}"/>
    </core:EventTriggerBehavior>
</interactivity:Interaction.Behaviors>
```

Obrázok 24: Implementácia obsluhy kliknutia na políčko

Príkaz CmdOpenOrderingPopUp zabezpečí spustenie dialógového okna pre objednanie jedla. Grafické rozhranie je definované v OrderDialog spojené s OrderDialogViewModel. Obsahuje prvky typu TextBlock o detailnejších informáciách o jedle na objednávku, rozdelené do grafického prvku Grid. Obsahuje dve tlačítka "Zavrieť", zavrie dialógové okno a "Objednať", objedná jedlo, zavrie toto okno a spustí okno, ktoré oznamuje o objednaní jedla. Okno o informácií objednania obsahuje informácie o objednanom jedle. Na nasledujúcom obrázku je zobrazené dialógové okno pred objednávkou a po objednávke.



Obrázok 25: Dialógové okno objednávky pred objednaním a po objednaní

Tretia časť sa skladá z dvoch TextBlock prvkov, ktoré získavajú obsah prostredníctvom property AccountName a AccountCredit.

Objednané jedlá

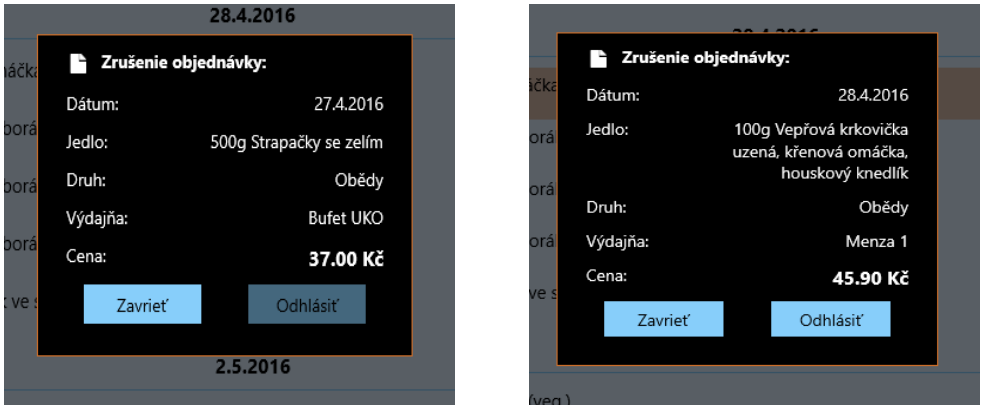
Implementácia grafického rozhrania stránky Objednané jedlá je rovnako vertikálne rozdelená na 3 časti ako stránka Jedálniček. GUI je definované v XAML súbore SecondPage prepojené s SecondPageViewModel. Na nasledujúcom obrázku je znázornená stránka Objednané jedlá.

Obrázok 26: Stránka Objednané jedlá

Prvá časť zhora sa skladá z informácie o rozmedzí dátumov pre zobrazenie objednaných jedál. Ďalej obsahuje tlačítko pre obnovenie stránky. Nachádzajú sa tam nachádzajú nadpisy, ktoré označujú popis stĺpca v ďalšej časti (Alt., Objednávka).

Druhá časť je zložená rovnako ako stránka Jedálniček z grafického prvku ListView, ktorý obsahuje skupiny rozdelené podľa dátumu objednania. Pre definíciu skupiny je vytvorený ViewModel OrderedGroupViewModel a pre jednotlivý riadok zoznamu je OrderedRowViewModel. Property, ktorá nastavuje zobrazenie dátumu skupiny sa nazýva DateText. ListView zobrazuje kolekcie objednávok, získavaných cez property OrderedView.

Políčko z ListView je definované grafickým prvkom Grid, ktoré obsahuje ďalšie prvky typu TextBlock. Popisujú informácie o objednávke, ako napríklad názov objednávky, číslo a výdajňa. Celé políčko je nastavené tak, aby reagovalo na kliknutie pomocou EventTriggerBehavior, ktoré bolo znázornené v predchádzajúcej časti stránky Jedálniček. Po kliknutí sa otvorí dialógové okno CancelDialog prostredníctvom príkazu CmdOpenCancelPopUp. Okno obsahuje informácie o objednávke rozdelené pomocou prvku Grid. Nachádzajú sa tam dve tlačítka, "Zavrieť", zavrie dialógové okno a "Odhlásiť", odhlási danú objednávku, uzavrie okno a aktualizuje stránku Objednané jedlá. ViewModel tohto dialógového okna kontroluje, či je možné objednávku zrušiť. V prípade že áno, tak je tlačítko Odhlásiť v stave Enable a keď nie je možné odhlásiť objednávku, tak je v stave Disable. Obvykle je to v prípadoch, kedy objednávku nie je možné zrušiť deň pred vyzdvihnutím. Na nasledujúcom obrázku je vidieť dialógové okno pre zrušenie objednávky v stave Disable a Enable.

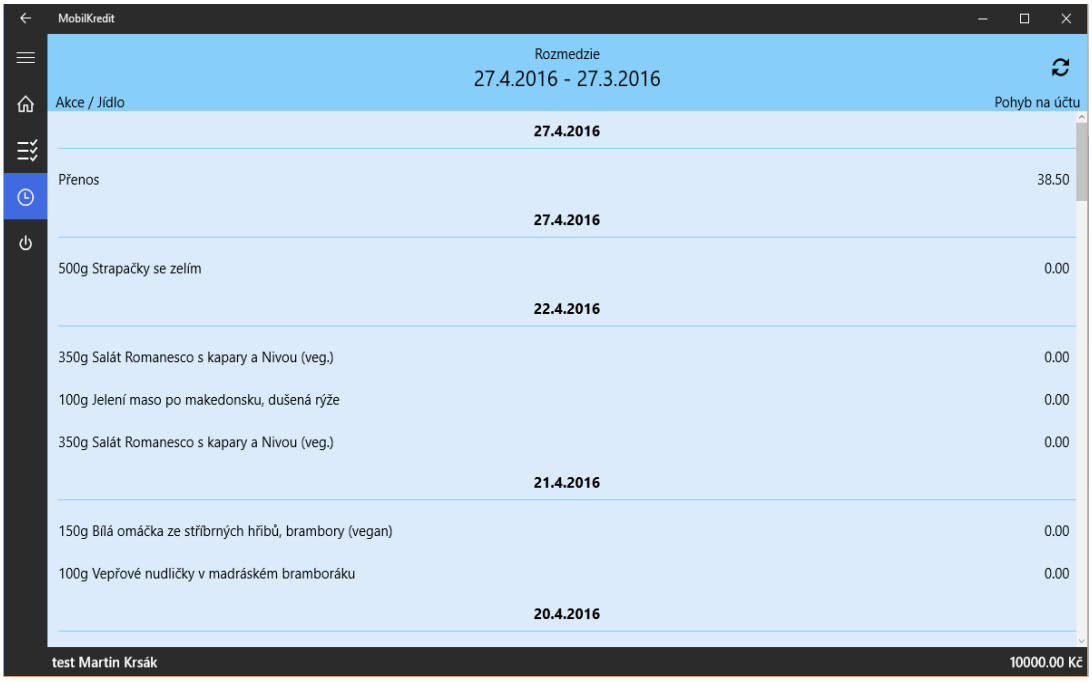


Obrázok 27: Dialógové okná pre zrušenie objednávky (Disable a Enable)

Tretia časť sa skladá z dvoch TextBlock prvkov, ktoré získavajú obsah prostredníctvom property AccountName a AccountCredit. Rovnako ako u stránky Jedálniček, jeden TextBlock zobrazuje menu používateľa a druhý jeho kredit.

História účtu

Implementácia grafického rozhrania stránky História účtu je rovnako vertikálne rozdelená na 3 časti ako predošlé stránky Jedálniček a Objednané jedlá. GUI je definované v XAML súbore ThirdPage prepojené s ThirdPageViewModel. Na nasledujúcom obrázku je znázornená stránka História účtu.

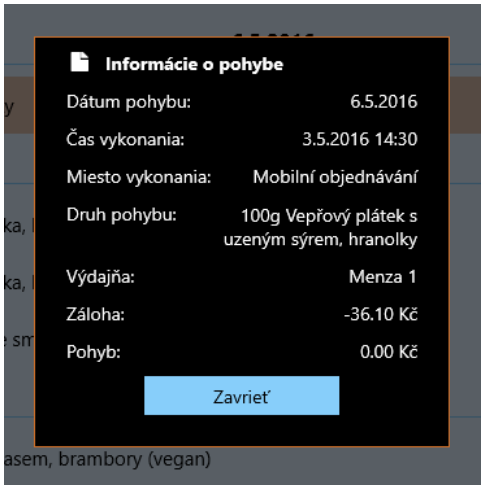


Obrázok 28: Stránka História účtu

Prvá časť zhora sa skladá z informácie o rozmedzí dátumov pre zobrazenie akcií histórie účtu. Ďalej obsahuje tlačítko pre obnovenie stránky. Nachádzajú sa tam nadpisy, ktoré označujú popis stĺpcov ďalšej časti (Akcie/Jedlo, Pohyb na účte).

Druhá časť obsahuje rovnako ako stránky Jedálniček a Objednané jedlá grafický prvok ListView, ktorý obsahuje skupiny rozdelené podľa dátumu uskutočnenia akcie histórie účtu. Pre definíciu skupiny je vytvorený ViewModel HistoryGroupViewModel a pre jednotlivý riadok zoznamu je HistoryRowViewModel. Property, ktorá nastavuje zobrazenie dátumu skupiny sa nazýva DateText. ListView zobrazuje kolekcie objednávok, získavaných cez property HistoryView.

Políčko z ListView je definované grafickým prvkom Grid, ktoré obsahuje dva prvky typu TextBlock. Prvý zobrazuje informáciu o akcií cez property ActionText a druhý cena akcie cez property PriceText. Celé políčko je nastavené tak, aby reagovalo na kliknutie pomocou EventTriggerBehavior, ktoré bolo popísané v časti stránky Jedálniček. Po kliknutí sa otvorí dialógové okno HistoryDialog prostredníctvom príkazu CmdOpenHistoryPopUp. Okno obsahuje informácie o akcií histórie účtu rozdelené pomocou prvku Grid. Nachádza sa v ňom jedno tlačítko "Zavrieť", ktoré zruší dialógové okno. Nasledujúci obrázok zobrazuje dialógové okno akcie histórie účtu.



Obrázok 29: Dialógové okno akcie histórie účtu

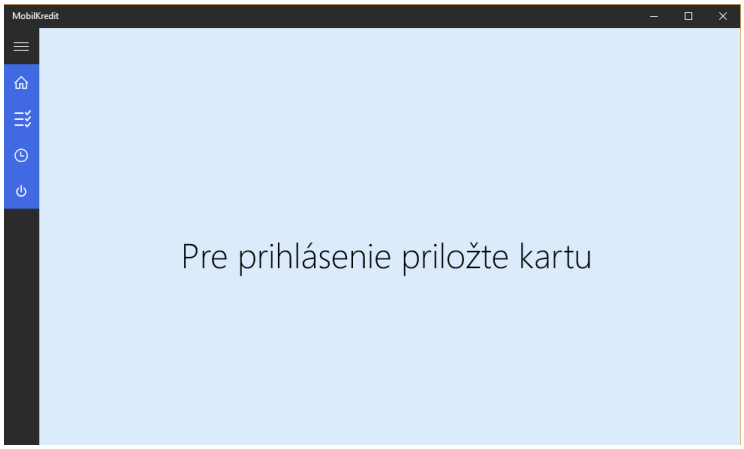
Tretia časť sa skladá z dvoch TextBlock prvkov, ktoré získavajú obsah prostredníctvom property AccountName a AccountCredit. Rovnako ako u predošlých stránok, jeden Textblock zobrazuje menu používateľa a druhý jeho kredit.

Prihlásenie/Odhlásenie

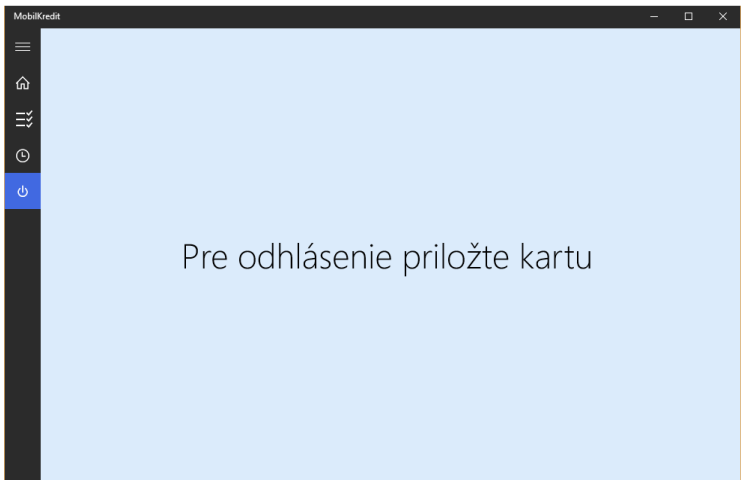
Táto stránka je prvá, ktorá sa používateľovi zobrazí pri spustení aplikácie. Obsahuje zatiaľ jeden TextBlock, ktorý informuje používateľa o priložení karty k čítačke kariet. V ďalších vývojoch aplikácie sa pridá obrázok a informácie o poskytovateľovi a ďalšie. Implementácia grafického rozhrania je v XAML súbore LogOutPage a je prepojená s ViewModel-om LogOutPageViewModel.

Pri implementácií čítania karty nebola k dispozícií čítačka kariet, ale možnosťou bolo vytvoriť simuláciu priloženia kartu v podobne postupnosti kláves na klávesnici s určitým časovým limitom. Táto postupnosť je definovaná v triede CardReader v adresári Services. Udalosť, ktorá nastane po úspešnom zadaní postupnosti znakov je definovaná v triede CardReadEvent a CardReadEventArgs. Ide o znaky ";" (štandardne tlačítko pod ESC), ľubovolné číslo od 0 do 20 a Enter. Po správnom zadaní postupnosti používateľ prejde do prihláseného stavu aplikácie, kde ho NavigationService presmeruje na stránku Jedálniček.

V prípade, že sa používateľ chce odhlásiť, musí prejsť na stránku Prihlásenie/Odhlásenie cez hlavné menu a znova stlačiť rovnakú postupnosť znakov, ktoré simulujú priloženie karty. V nasledujúcich obrázkoch sú znázornené stavy stránky Prihlásenie/Odhlásenie keď používateľ nie je prihlásený a keď je prihlásený.

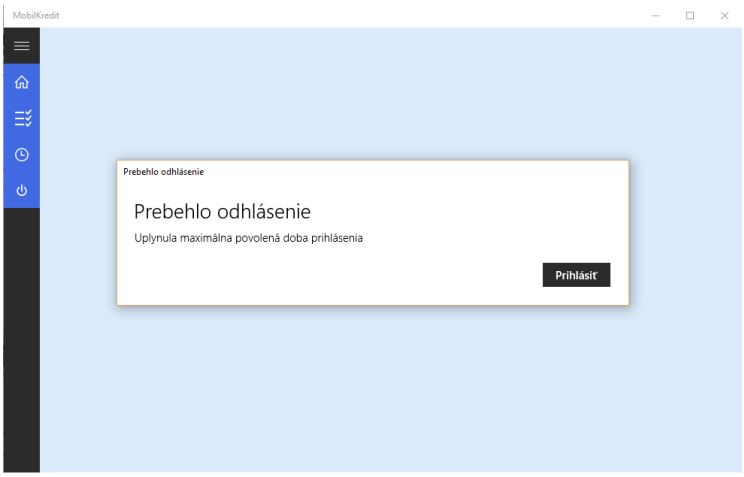


Obrázok 30: Stránka Prihlásenie/Odhlásenie v stave odhlásenia



Obrázok 31: Stránka Prihlásenie/Odhlásenie v stave prihlásenia

Pri uplynutí časovača prihlásenia používateľa do aplikácie sa pri prvom volaní na server zobrazí MessageBox, ktorý informuje používateľa o odhlásení a pre znovu prihlásenie. MessageBox po stlačení tlačítka Prihlásiť naviguje cez NavigationService na stránku Prihlásenie/Odhlásenie a dostáva aplikáciu do stavu odhlásenia. Nasledujúci obrázok zobrazuje tento prípad.



Obrázok 32: MessageBox pri odhlásení používateľa

8. Možnosti rozšírenia

Jednou z možností rozšírenia aplikácie MobilKredit je vytvorenie kalendára pre objednávanie jedál. Používateľ otvorí kalendár, v ktorom môže vidieť poznámky pri určitých dňoch. Poskytlo by to ľahšiu dohľadateľnosť pri objednávaní na vzdialenejší dátum. Pri objednávaní by informácie o objednávke obsahovali obrázky posielané z aplikačného servera, prípadne ďalšie informácie o objednávke, ktoré v aplikácií MobilKredit nie sú, ako napríklad zloženie jedla rozdelené do cukrov, tukov a ďalšie.

Na toto by sa dalo naviazať ďalšie rozšírenie, ktoré by si ukladalo zloženie objednaných jedál a každému používateľovi by vytvorilo osobný jedálniček s navrhovanými jedlami na základe zloženie už minulých objednaných jedál (používateľ si často objednáva ako prílohu hranolky, tak medzi navrhovanými by boli v sekcií prílohy ako prvé hranolky).

Aplikácia by mohla ukladať telefónne číslo používateľa a po objednaní jedla by mohla informovať prostredníctvom SMS používateľa v deň, ktorý má objednávku vyzdvihnúť. Ďalším rozšírením by bola burza jedál, ktorá by bola umiestnená na ďalšej stránke pod názvom Burza. V nej by klienti mohli poskytovať objednávku za nižšie ceny v dobe, keď konkrétnu objednávku nejde zrušiť a nemôžu sa dostaviť pre vyzdvihnutie objednávky.

Pre spätnú väzbu menz a vývojárov aplikácie MobilKredit, by mohla byť jednou z rozšírení stránk,a na ktorej by používatelia mohli ohodnotiť služby aplikácie, menzy. Ohodnotiť konkrétne jedlo, prípadne pridať komentár pre vylepšenie aplikácie, alebo pre navrhnutie lepšieho chodu menzy.

Ďalšia z možností rozšírenia by mohlo byť dobíjanie kreditu klienta cez aplikáciu MobilKredit. Uplatnilo by sa to v aplikácií, ktorá je k dispozícií cez Google Play, alebo voľne na stiahnutie z internetu. Pre aplikáciu, ktorá bude vstavaná v menze by toto rozšírenie bolo problematické a mohlo by spôsobiť rady pred miestom, na ktorom bude tablet s aplikáciou MobilKredit, čo spomalí chod menzy.

Rozšírenie, ktoré je potrebné pre aplikáciu MobilKredit, je rozdelenie aplikácie na dva typy, pre menzy a voľne dostupné na stiahnutie. Aplikácia, ktorá bude vstavaná v menze bude musieť byť neustále spustená a nebude možnosť ju vypnúť, pričom u aplikácie, ktorá bude voľne dostupná to bude štandardné správanie. Okrem toho aplikácia, ktorá bude voľne dostupná na stiahnutie bude poskytovať prihlásenie do nej prostredníctvom loginu a heslom, kde u aplikácie, ktorá ma byť v menze sa klient bude prihlasovať prostredníctvom čítačky kariet po priložení platnej karty.

9. Záver

Cieľom tejto bakalárskej práce bolo vytvoriť aplikáciu, ktorá umožní stravníkom menz objednávať jedlá, prehľadávať a rušiť ich, ako aj prehľadávať históriu objednaných jedál a iných akcií nad ich účtom.

Aplikácia bola navrhnutá ako online program tak, aby ju mohli využívať skoro všetci používatelia s operačným systémom Windows 10. To je zabezpečené vďaka optimalizácií pre rôzne rozlíšenia a veľkosti obrazoviek rôznych typov zariadení, ako sú napríklad mobilné zariadenia, tablety alebo pracovné plochy.

Funkcionalita aplikácie, rovnako ako aj technológie, ktoré sa využili pri tvorbe aplikácie, boli vopred určené zadaním práce a bližšou špecifikáciou od firmy Anete s.r.o.. Pri vytváraní nebolo možné splniť všetky časti zadania. V časti doplnenia zadania bola špecifikovaná implementácia prevádzky aplikácie v krízových situáciách (výpadok spojenia so serverom a podobne). Táto funkcionalita nebola možná s výslovným zákazom od externého zadávateľa. Nebolo možné objednávať jedlo v režime Offline. Jeden z dôvodov je, že všetky objednávky na sebe závisia a každá objednávka má určitý počet porcií.

Prvotnú časť vývoja aplikácie som venoval analýze a návrhu dátového modelu, architektúry celej aplikácie a jej štruktúry. Bolo potrebné navrhnúť komponenty MVVM vzoru, v ktorom View bude preberať informácie od časti ViewModel a ten bude získavať dáta prostredníctvom parametrov od Modelu. Druhá časť vývoja, ktorá bola najrozsiahlejšia, bola implementácia jednotlivých tried MVVM architektúry a grafického používateľského rozhrania. Treťou a poslednou časťou vývoja bolo testovanie na rôznych typoch zariadení, ako aj testovanie intuitívnosti aplikácie na vybraných osobách.

Aplikácia bola otestovaná na vybratých osobách. Testovala sa intuitívnosť a prehľadnosť aplikácie. Prevládali prevažne dobré hodnotenia od testovaných osôb, kde bol vyzdvihnutý hlavne dizajn aplikácie, ktorý získal veľký úspech. Okrem testovaní intuitívnosti a prehľadnosti bola aplikácie otestovaná na troch zariadeniach s operačným systémom Windows 10 (mobilné zariadenie, tablet a pracovná plocha).

V súčasnosti je celé riešenie nasadené na testovacom serveri. Po úplnom dokončení sa predpokladá jeho nasadenie a použitie ako náhrada existujúceho objednávania jedál prostredníctvom starých prostriedkov.

Bakalárska práca bola tiež prostriedkom, prostredníctvom ktorého som sa oboznámil s programovaním v jazyku C# a obšírnejším používaním vývojového prostredia Visual Studio. Naučil som sa pracovať na aplikácií, ktorá bude reálne využívaná vo firme. Oboznámil som sa s novou technológiou UWP a s návrhovým vzorom MVVM, s ktorým som sa predtým nestretol. Prácou na tejto bakalárskej práci sa zlepšili moje programátorské zručnosti.

10. Literatúra

[1] "Analysis of the Current Apps Available in Online Vendor Markets," [Online]. Available: http://mhealth.jmir.org/2013/2/e24/. [Accessed 15 5 2016].

[2] "Microsoft's Display Dock will turn your phone into a Windows 10 PC," [Online]. Available: http://www.stuff.tv/news/microsofts-display-dock-will-turn-your-phone-windows-10-pc. [Accessed 15 5 2016].

[3] ".Net Framework," [Online]. Available: http://www.academictutorials.com/microsoft.net/dotnet-framework.asp. [Accessed 15 5 2016].

[4] A. Nathan, "XAML Demystified," in *Windows Presentation Foundation Unleashed (WPF) 1st Edition*, Sams Publishing, 2006, p. 656.

[5] "Introduction Of WPF," [Online]. Available: http://r4r.co.in/wpf/01/tutorial/basics/introduction_to_wpf.shtml. [Accessed 15 5 2016].

[6] "Device primer for Universal Windows Platform (UWP) apps," [Online]. Available: https://msdn.microsoft.com/en-us/windows/uwp/input-and-devices/device-primer. [Accessed 15 6 2016].

[7] "App lifecycle," [Online]. Available: https://msdn.microsoft.com/en-us/windows/uwp/launch-resume/app-lifecycle. [Accessed 15 5 2016].

[8] "The MVVM Pattern," [Online]. Available: https://msdn.microsoft.com/en-us/library/hh848246.aspx. [Accessed 15 5 2016].

[9] "MVVM (Model View ViewModel) Introduction," [Online]. Available: http://www.c-sharpcorner.com/UploadFile/0b73e1/mvvm-model-view-viewmodel-introduction-part-1/. [Accessed 15 5 2016].

[10] "Prism: patterns & practices," [Online]. Available: https://compositewpf.codeplex.com/. [Accessed 15 5 2016].

[11] "Patterns in the Prism Library 5.0 for WPF," [Online]. Available: https://msdn.microsoft.com/en-us/library/ff921146(v=pandp.40).aspx. [Accessed 15 5 2016].

[12] "Mobilní objednávání Kredit," [Online]. Available: https://play.google.com/store/apps/details?id=cz.novaklukas.droidkredit&hl=cs. [Accessed 15 5 2016].

[13] "MobilKredit," [Online]. Available: http://www.appdownloader.net/iOS/App/735214126/com.astrumq.mobilkredit. [Accessed 15 5 2016].

[14] "WebKredit 8.161.1.13," [Online]. Available: http://sur.ly/i/jidelna.fnol.cz/. [Accessed 15 5 2016].

[15] "Testovacie prostredie REST," [Online]. Available: http://www.slideshare.net/FutureProcessing/rodowisko-testowe-pod-resta. [Accessed 15 5 2016].