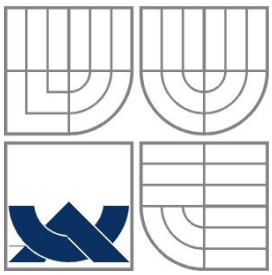
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

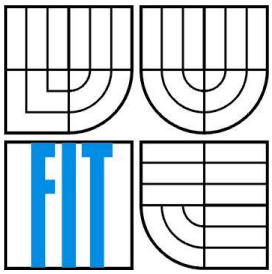
Fakulta informačních technologií
Faculty of Information Technology

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

Brno, 2016 Tomáš Kello



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

TERMINÁL PRO VYDÁVÁNÍ JÍDEL

TERMINAL FOR MEAL HAND OUT

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — Tomáš Kello
AUTHOR

VEDOUCÍ PRÁCE — Ing. Zbyněk Křivka, Ph.D.
SUPERVISOR

BRNO 2016

Abstrakt

Práca rieši problematiku softvérovej aplikácie, ktorá sa používa na výdaj jedál vo veľkokapacitných jedálňach pre zákazníkov s použitím čipovej karty. Projekt je založený na univerzálnej platforme pre operačný systém Windows so spustením na rôznych zariadeniach od mobilov až po stolové počítače. Komunikačným prostriedkom medzi aplikáciami a serverom je internetová sieť, vďaka ktorej môžeme využiť vzdialené ovládanie.

Abstract

The main topic of this paper is meal serving in large canteen with client wireless card. Where is necessary to make two applications, one for boarder and one for service. Project is based on universal platform for operating system Windows. It allows us to run this application on different types of devices from smart phones to desktop computers. Communication between these two applications and server is internet network. It offers us possibility to control running of applications remotely.

Klíčová slova

Výdaj jedál, UWP, C#, WCF, univerzálna platforma od Microsoftu, Windows 10, MVVM, Prism

Keywords

Meal hand out, UWP, C#, WCF, Microsoft universal platform, Windows 10, MVVM, Prism

Citace

Tomáš Kello: Terminál pro vydávání jídel, bakalářská práce, Brno, FIT VUT v Brně, 2016

Terminál pro vydávání jídel
Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Zbyněk Křivka, Ph.D.
Další informace mi poskytli konzultant Ing. Tomáš Juříček a pracovní kolega Ing. Karel Král.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Tomáš Kello
8.5.2016

Poděkování

Veľmi rád by som sa poďakoval vedúcemu práce Ing. Zbyněk Křivka, Ph.D. za jeho pomoc a cenné rady. Pri vývoji mi poskytol pomoc ja Ing. Tomáš Juříček, ktorému by som sa touto cestou tiež rád poďakoval.

Obsah

1 Úvod

Bakalárska práca sa venuje softvérovej aplikácii na výdaj jedál vo veľkokapacitných jedálňach s použitím zákazníckej čipovej karty. V prvej časti práce je rozpracovaná problematika softvérovej aplikácie s možnosťami vývojovej platformy. Následne ich porovnanie a odporučenie najvýhodnejšej varianty. Druhá časť práce je zameraná na samotnú implementáciu vo veľkokapacitných jedálňach. Zahŕňa komunikáciu medzi aplikáciami, návrh grafického vzhľadu a overenie testovaním unit a integračnými testami. Pôjdeme od základov a princípov programovacieho jazyka až k samotnej komplexnejšej aplikácii. Záverom práce zhodnotíme celkový produkt a vyslovíme odporúčania na ďalšie možné smerovanie vývoju. Práca môže byť využívaná ako pomocný mechanizmus pri výdaji jedál. Jej úlohou je spracovať priloženú užívateľskú kartu a zobraziť jedlá, ktoré majú byť pre daného užívateľa vydané. K použiteľnosti aplikácie v prevádzke je potrebná aj aplikácia na objednávanie jedál, ktorá nie je náplňou tejto práce.

Aplikácia je konštruovaná na zariadenia s displejom ako sú tablety, mobily, počítače a iné zobrazovacie zariadenia. Primárne je aplikácia vyvíjaná pre operačný systém Windows 10. Na použitie aplikácie je nevyhnutné pripojiť čítačku kariet a pripojenie k sieti internet. Výpočtový výkon zariadení nie je nevyhnutným faktorom, keďže záťaž aplikácie na výpočty nie je veľká. Základnou funkcionalitou aplikácie je prostredníctvom čítačky kariet rozpoznať užívateľa (stravníka), ktorý prišiel po objednanú stravu do jedálne. Po priložení čipovej karty k čítačke sa zobrazia informácie na displeji o užívateľovi a jedlách, ktoré má na daný deň objednané. Dôležité pri zobrazovaní informácii na displeji je adaptovanie sa aplikácie k zariadeniu podľa jeho rozlíšenia. Nezobrazovať príliš veľa informácii, udržovať dostatočný kontrast a veľkosť textu a zároveň nepripraviť stravníka o informácie, ktoré sú pre neho dôležité.

Paralelne s týmto zariadením je druhé zariadenie, ktoré je umiestnené v miestnosti vydávania jedál. Úlohou zariadenia je zobraziť obsluhe jedálne číslo jedla, ktoré majú vydať zákazníkovi. Rozmiestnenie zariadení môžeme vidieť na obrázku 1.1. Aby bola aplikácia použiteľná a úspešná je potrebné správne a prehľadné zobrazenie potrebných údajov, čo je náplňou tejto práce.

Obrázok 1.1: Umiestnenie zariadení v prevádzke

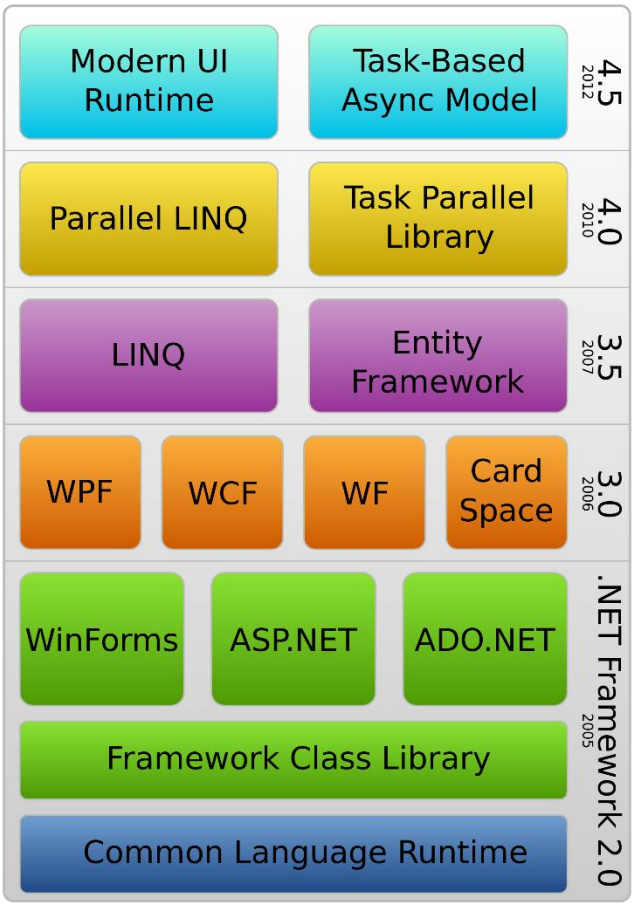
Podľa charakteristiky aplikácie sa očakáva používanie v zatvorených miestnostiach počas celého dňa. Pri výdaji obedov je potrebné brat do úvahy dopad slnečných lúčov na zariadenie a umelé osvetlenie, ktoré môžu zhoršovať čitateľnosť zariadenia.

2 Analýza

Náplňou práce je vytvoriť riešenie, ktoré urýchli a zefektívni výdaj jedál vo veľkokapacitných jedálňach. V prvom kroku je zoznámenie sa s možnosťami vývoja aplikácii pre operačný systém Windows 8 a 10 na mobilných zariadeniach, získanie znalostí s frameworkom .NET a jazykom C#. Ďalší krok obsahuje návrh a spôsob komunikácie medzi aplikáciami, návrh grafického vzhľadu, ktoré sa prispôsobí zariadeniu. Po úspešnom vytvorení aplikácie pokračuje zadanie práce v jeho otestovaní unit a integračnými testami. V závere práce je našou úlohou zhodnotiť celkovú implementáciu a navrhnúť možnosti rozšírení softvérovej aplikácie.

2.1 Vývoj frameworku .NET od Microsoftu

Vývoj aplikácii pod operačným systémom Windows je založený na podpornom jadre .NET. Jeho vývoj začal pred vyše desiatimi rokmi. Prvá používanejšia verzia bola 2.0, ktorá prišla spolu s frameworkom WinForms, ktorý sa staral o základnú logiku aplikácie. Od počiatkov sa vývoj aplikácii posunul ďalej. Veľkou zmenou vo vývoji aplikácii bolo zavedenie .NET 3 spolu s veľmi populárnym frameworkom WPF (Windows Presentation Foundation), ktorému sa venuje kapitola 2.2. Aj v dnešnej dobe je stále často používaný. Prehľad vývoja .NET je zobrazený na obrázku 2.1. Momentálne sa na trh uviedol .NET4.6, spolu s operačným systémom Windows 10. Platforma je v častých úpravách, keďže je na trhu len od leta 2015 (Na obrázku 2.1 nie je zobrazená).



Obrázok 2.1: vývoj frameworku .NET [1]

2.2 Windows Presentation Foundation (WPF)

Framework, spolu s grafickým podsystémom pomáha vývojárom vytvoriť atraktívny a efektívny grafický vzhľad aplikácie. Prvé používanie začalo s predstavením .NET 3. WPF. Poskytuje konzistentný programovací model k vývoju aplikácii a oddeleniu vývoja grafickej časti aplikácie od výpočtovej.

2.2.1 Výhody Windows Presentation Fondation

Výhodou aplikácie je, že spája niekoľko rôznych druhov aplikácii do jednej komplexnej platformy, v ktorej je možné vytvoriť viaceré druhy projektov. Prehľadnú tabuľku môžeme vidieť v tabuľke 2.1. Spája návrh aplikácii pre dokumenty, video, obrázky, dvoj rozmerné a 3D aplikácie, ktoré doteraz bolo potrebné vytvárať s rozdielnymi vývojovými prostriedkami.

[1] Wikimedia Foundation, 2005. Dostupné online: https://upload.wikimedia.org/wikipedia/commons/thumb/d/d3/DotNet.svg/2000px-DotNet.svg.png

OBSAH aplikácie	Windows Forms	PDF	Windows Forms/GDI+	Windows Media Player	Direct 3D	WPF
Grafické rozhranie	**X**					**X**
Dokumenty	**X**					**X**
Dokumenty s formátovaním		**X**				**X**
Obrázky			**X**			**X**
Video a hudba				**X**		**X**
2D grafika			**X**			**X**
3D grafika					**X**	**X**

Tabuľka 2.1: Podpora obsahu v aplikáciách

Ak mala aplikácia spájať viaceré skupiny obsahov, tak sa situácia vývojárovi veľmi komplikovala. Vývoj sa preto unifikoval a vytvorila sa jednotná platforma WPF pre kombináciu viacerých druhov obsahu, čím je možné vykresliť 3D objekt pomocou Direct3D spolu s formátovaným obsahom vo formáte PDF v jednej aplikácii.

2.2.2 Dizajn Windows Presentation Fondation

Na opis a návrh dizajnu sa využíva jazyk XAML. Pri vývoji grafiky je jednoduchší a prehľadnejší ako programovací jazyk. Ponúka väčšie možnosti definovania vzhľadu, štýlov a tém aplikácii. Po vytvorení dizajnu aplikácie v XAML môže programátor využívať jeho jednotlivé prvky ako sú tlačidlá, textové polia, upravovať farby pozadia, viditeľnosť a iné. Vďaka spomínaným prvkom dokážu dizajnéri a programátori pracovať súčasne na jednom projekte.

2.2.3 Prepojenie dizajnu a aplikácie

Prepojenie týchto dvoch vrstiev je založené na posielaní správ. Napríklad dizajnér nastaví obsah textového poľa na určitú hodnotu premennej v programe. V momente, keď sa hodnota premennej zmení, vygeneruje sa signál o zmene a zobrazenie reaguje na zmenu a prekreslí obsah na novú hodnotu.

Rovnako funguje aj opačná komunikácia z dizajnových prvkov smerom k aplikácii. Po kliknutí na niektorý objekt definovaný v XAML sa vygeneruje signál, ktorý zachytí aplikácia a vykoná obslužnú funkciu pre dané tlačidlo.

2.2.4 Zhrnutie Windows Presentation Fondation

Platforma je často používaná v aktuálnych projektoch. Spája veľké množstvo funkcionality a dáva vývojárom veľké možnosti pri návrhu grafického rozhrania, rovnako ako aj pri podpore rôznych rozšírení v podobe 3D objektov, hudby, videa, komunikácie cez internet a mnohé ďalšie.

Samotná funkcionalita na stolovom počítači nestačí. Výkony prenosných zariadený stúpajú a požiadavky na spustenie rovnakých aplikácii na počítači, notebooku, tablete či mobilnom telefóne sú vyššie.

Okrem malých prenosných zariadení sa dostávajú do popredia aj rôzne zariadenia určené na prezentáciu a interaktivitu. Samotný projektor, ktorý zobrazí obsah prezentácie sa stáva nedostačujúcim a vznikajú interaktívne tabule, prepojené s počítačom ovládané dotykmi. Dajú sa prirovnať k projektorom s vlastným výpočtovým výkonom.

Napríklad zariadenie Surface Hub od spoločnosti Microsoft, je porovnateľné k interaktívnej tabuli, ktorá nepotrebuje k používaniu počítač alebo notebook, z ktorého čerpá dáta na zobrazenie. Funguje autonómne s vlastným výpočtovým jadrom.

Zariadenia boli jasnou požiadavkou k vytvoreniu novej vývojovej platformy, ktorá by dokázala vytvoriť aplikáciu spustiteľnú na všetkých dostupných zariadenia. A jej dizajn by sa prispôsoboval zariadeniu.

Veľkou nevýhodou v aplikácii pod WPF je jadro, ktoré je primárne stavané na jednojadrové výpočty. V dnešnej dobe sa stretávame s počítačmi, ktoré majú 4 a viac jadier a preto aplikácia nemôže využiť celý výpočtový výkon počítača.

2.3 Universal Windows Platform (UWP)

Hlavnou myšlienkou platformy je vytvorenie grafického rozhrania, ktoré bude kompatibilné s viacerými zariadeniami Windows:

- Mobilné zariadenia: mobily s operačným systémom Windows, tablety
- Stolové zariadenia: notebooky, počítače
- Zariadenia pre skupinu ľudí: Surface hub (interaktívny projektor s vlastným procesorom)
- Kompaktné zariadenie: napríklad hodinky alebo zberače informácii zo senzorov v domácnosti

2.3.1 Kompatibilita

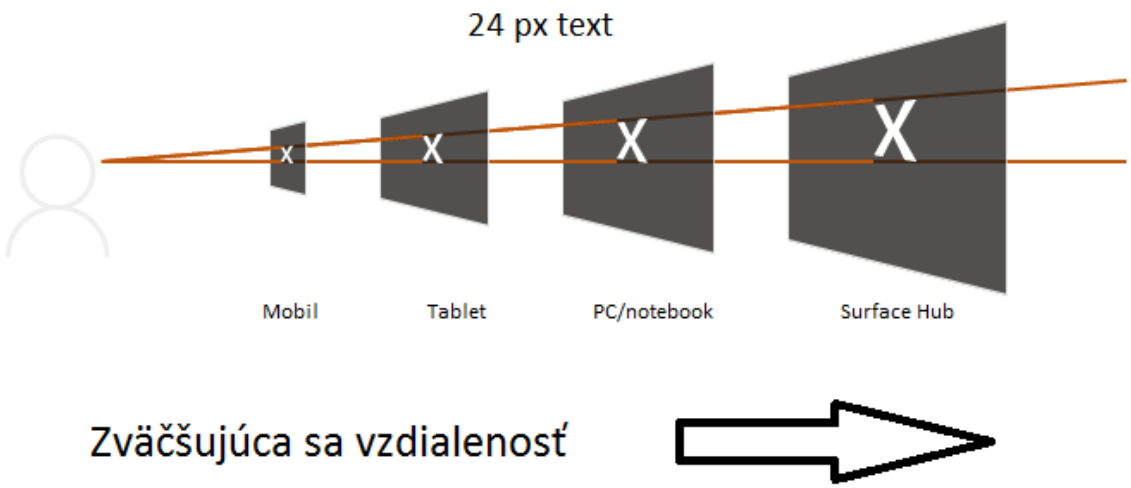
Platforma má možnosť výberu, na ktorých zariadeniach bude povolená, prípadne bez obmedzenia, kde všetky zariadenia majú povolenie k spusteniu.

Vytvorením grafického rozhrania, ktoré podporuje škálu zariadení a všetky vyrábané rozlíšenia obrazoviek, sú veľmi náročné. V dnešnej dobe sa môžeme stretnúť s malými rozlíšeniami v mobilných zariadeniach, ako napríklad BLU Win JR, model 2015 s rozlíšením 480 x 854 pixelov, až po rozlíšenia používané na počítačoch a Surface hub, kde hodnoty dosahujú aj viac ako FULL HD rozlíšenie (1,920 × 1,080 pixelov).

Pre potrebnú kompatibilitu bola vytvorená UWP platforma, ktorá sa snaží prispôsobovať dizajn zariadeniu a jeho rozlíšeniu.

2.3.2 Funkcionalita UWP

Platforma sa v prvom rade stará o čitateľnosť textu. To znamená, že veľkosť písma 24 pixelov ako môžeme vidieť na obrázku 2.2 sa prispôsobuje zariadeniu a vzdialenosti od užívateľa, v akej sa dané zariadenie zvyčajne používa.



Obrázok 2.2: Prispôsobenie textu k rozlíšeniu obrazovky [2]

Rovnako dôležité, ako zabezpečiť veľkosť textu, aby bol čitateľný, je prispôsobiť dizajn orientácii displeja. Väčšina počítačov, notebookov a projektorov funguje v orientácii displeja, kde dlhšia strana je vo vodorovnej pozícii. Tablety využívajú približne rovnako obe orientácie a mobilné zariadenia sú primárne využívané s orientáciou na výšku. Pri analýze výsledného produktu, je potrebné definovať konečného užívateľa s programovým zariadením, pre koho produkt bude a na akých zariadenia bude väčšinou spúšťaný. V prípade, že klienta chceme obmedzovať, je potrebné dizajn premyslieť o to viac a zabezpečiť rozloženie objektov na obrazovke pri použití oboch orientácii.

Ďalšou potrebnou funkcionalitou je zabezpečiť kompatibilitu medzi dotykovými zariadeniami a zariadeniami, ktoré využívajú ukazovateľ. Použitím dotykového displeja alebo ukazovateľa sa nič nemení. Dotyková obrazovka nám ponúka možnosť využívať dotykové gestá. Tieto však nie je možné reprodukovať ukazovateľom a je potrebné vytvoriť náhradu v prípade, že užívateľ nemá možnosť dotykovej obrazovky.

Okrem dizajnovo dôležitých funkcionalít ponúka platforma funkcie, ako automatické animácie, podpora viacerých jazykov alebo vysoký kontrast.

2.3.3 Prispôsobivý dizajn

Na vytvorenie grafického rozhrania, ktoré sa prispôsobí zariadeniu, jeho rozlíšeniu a funkcionalitám, ktoré obsahuje (fotoaparát, akcelerometer, GPS modul, ...) sa využívajú rôzne techniky a možnosti.

[2] MSDN, 2016. Dostupné online: https://i-msdn.sec.s-msft.com/dynimg/IC794180.png

Premiestnenie

V prípade, že užívateľ využíva zariadenie na výšku, má väčšiu aktívnu plochu vo vertikálnej polohe a jednotlivé objekty sa zobrazujú pod sebou. V prípade, zobrazenia na šírku, je pôvodné rozloženie nepraktické a neprehľadné. Preto je potrebné jednotlivé objekty premiestniť vedľa seba a využiť celú šírku zariadenia.

Obrázok 2.3: Prispôsobenie sa otočeniu displeja [3]

Zmenenie veľkosti a roztiahnutie

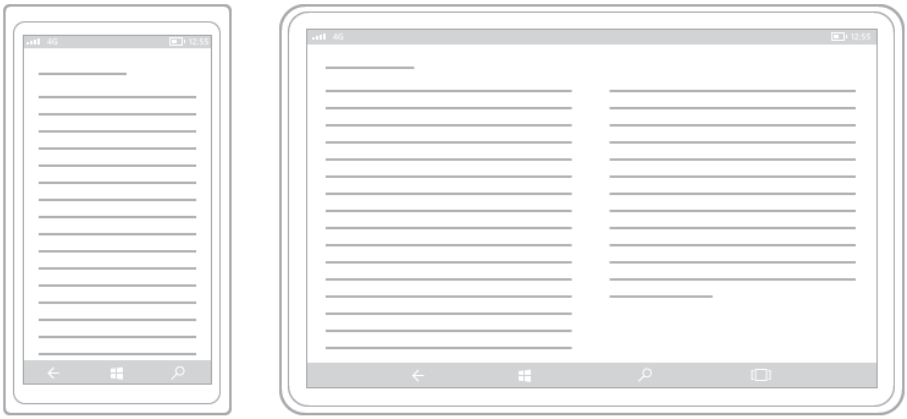
V prípade, že aplikácia nevyužíva viac objektov, ktoré by sa pri väčšej obrazovke mohli preskupiť môžeme využiť ďalšiu variantu zmeny veľkosti, ak aplikácia zobrazuje textový dokument. Najlepšou variantou pri zmene otočenia displeja, je objekt roztiahnuť bez zachovania pomerov strán, ako môžeme vidieť na obrázku 2.4

Obrázok 2.4: Rozšírenie pracovnej plochy [4]

[3] MSDN, 2016. Dostupné online: https://i-msdn.sec.s-msft.com/dynimg/IC795412.png
[4] MSDN, 2016. Dostupné online: https://i-msdn.sec.s-msft.com/dynimg/IC795414.png

Zmena počtu stĺpcov vo vykresľovaní

Možnosťou prispôsobenia sa textovému objektu pri zmene orientácie displeja zväčšením alebo zmenšením horizontálneho priestoru, je pridanie alebo odobranie stĺpca textu. Hlavnou výhodou je zachovávanie dĺžky riadkov, ako môžeme vidieť na obrázku 2.5.



Obrázok 2.5: Zmena počtu stĺpcov [5]

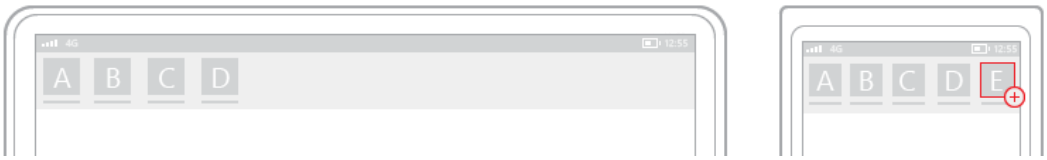
Zobrazenie funkcii na základe možností zariadenia

Keďže framework je určený viacerým druhom zariadení s rôznymi možnosťami hardvérových rozšírení ako je fotoaparát, GPS a iné príslušenstvo.

V prípade, že aplikácia má možnosť pridať fotografiu, samotné pridávanie môže mať možnosť:

- Pridať z galérie obrázkov alebo z disku
- Odfotografovať nový obrázok

V prípade, že zariadenie nemá kameru, je výhodné túto možnosť nezobrazovať.



Obrázok 2.6: Zmena hlavnej lišty [6]

Zmena rozloženia

Návrhár môže na základe veľkosti displeja úplne zmeniť rozloženie, predovšetkým ak potrebuje využiť čo najväčšiu plochu na zobrazenie.

V ukážke môžeme vidieť na mobilnom telefóne použité vertikálne rozloženie, ktoré je pri použití aplikácie skryté, aby nezaberalo voľný priestor. Po kliknutí na menu, v ľavom hornom rohu sa lišta roztiahne. Pri použití tabletu s väčším displejom, sa jeho obsah zmenší do hornej časti displeja v podobe horizontálneho rozloženia prvkov, bez možnosti skrývania.

[5] MSDN, 2016. Dostupné online: https://i-msdn.sec.s-msft.com/dynimg/IC795415.png

[6] MSDN, 2016. Dostupné online: https://i-msdn.sec.s-msft.com/dynimg/IC795416.png

Obrázok 2.7: Orientácia hlavného panela [7]

Celková zmena dizajnu

Poslednou, častou využívanou metódou pri návrhu, je definovanie špecifického zobrazenia pre jednotlivé druhy zariadení. Každé zobrazenie vychádza zo základného zobrazenia s možnosťou použiť jedno zo špeciálnych zobrazení, ktoré bude lepšie vyhovovať na dané zariadenie.

Využíva sa hlavne na mobilných zariadeniach s malým rozlíšením, kde treba jednotlivé objekty zobrazovať väčším textom na celú obrazovku. Pri použití aplikácie na počítači sa k tejto časti môžu používať aj menej dôležité objekty.

Napríklad obrázok 2.8, zobrazuje zoznam, ktorý na mobile využíva celú obrazovku len na zobrazenie menu, aby bol text čitateľný. Následne sa položka zo zoznamu, otvorí v novom okne. V počítači, ktorý ma veľký displej sa zoznam a samotný obsah vybranej položky umiestnia na jednu obrazovku a urýchľujú navigáciu. Typické použitie je v emailovej aplikácii, ktorá je súčasťou inštalácie systému Windows 10.

[7] MSDN, 2016. Dostupné online: https://i-msdn.sec.s-msft.com/dynimg/IC795417.png

Obrázok 2.8: Zmena dizajnu v závislosti na zariadení [8]

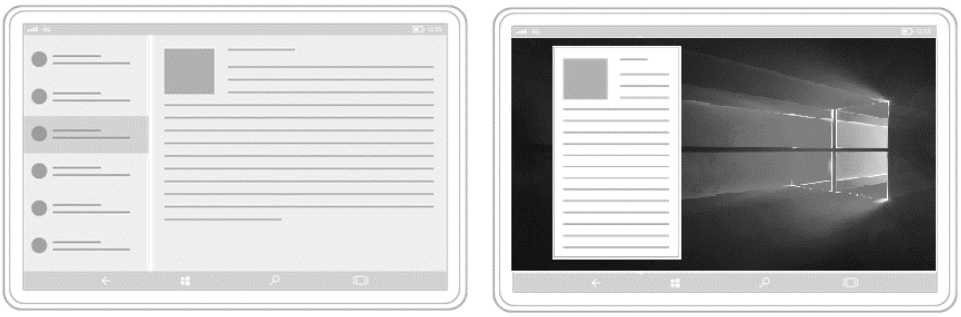
2.3.4 Zhrnutie

Windows Universal Platform vytvoril veľké možnosti v dizajne aplikácii. Sústreďuje sa hlavne na funkčnosť a prehľadnosť aplikácii, ktorá zodpovedá požiadavkám zariadenia. Pri vývoji je potrebné definovať aplikáciu, pre ktoré zariadenia je použiteľná, alebo spraviť univerzálny návrh, ktorý sa prispôsobí a bude prehľadný na všetkých rozlíšeniach a zariadeniach.

Nové možnosti prinášajú komplikácie a náročnosť pri samotnom vývoji. Veľkou výhodou je vytvorenie aplikácie a jej použiteľnosť na všetkých zariadeniach bez nutnosti akéhokoľvek zásahu do kódu. Nepriamou výhodou návrhu grafického rozhrania je pri použití na zariadeniach s vyšším rozlíšením bez použitia zobrazenia na celú obrazovku.

Napríklad pri počítačoch, keď si spustíme aplikáciu a zmenšíme jej pracovné okno na hodnoty podobné ako sú na mobile. Aplikácia bude reagovať na zmeny a zobrazí sa podobne ako na tablete alebo mobile. To znamená,že nepotrebné navigačné prvky skryje, aby aj v malom rozlíšení bol prehľadný hlavný obsah.

[8] MSDN, 2016. Dostupné online: https://i-msdn.sec.s-msft.com/dynimg/IC795418.png



Obrázok 2.9: Zmenšenie rozmerov aplikácie

2.4 Porovnanie vývojových platforiem so zameraním na projekt

V dokumente boli opísané dve najnovšie vývojové platformy od firmy Microsoft, ktoré sa aktuálne využívajú. O niečo viac bol opis zameraný na univerzálnu platformu Windows, keďže je nová a prináša mnohé zlepšenia a zmeny s ktorými sa často nestretáme, na rozdiel od WPF, ktorá je všeobecne známa medzi programátormi a veľmi často využívaná.

V súčasnosti sú väčšie požiadavky na vytvorenie univerzálnych aplikácii. Platforma UWP nie je úplne stabilná a podlieha neustálym zlepšeniam a doplnkom, aj napriek tomu prináša veľké výhody, ktoré využijeme v projekte. Budeme sa venovať hlavne platforme, na ktorej aplikácia vznikne a jej ďalším modulom, ktoré budú v aplikácii použité.

2.5 Windows Communication Foundation (WCF)

Aplikácia využíva WCF služby. WCF je framework na vytvorenie komunikačných služieb, ktoré ďalej môžu využívať aplikácie. WCF služba sa v aplikácii stará o asynchrónnu komunikáciu medzi aplikáciami. Okrem tejto základnej funkcionality ponúka veľké množstvo ďalších možností. Keďže sa jedná o komunikáciu cez verejné siete internetu, základnou požiadavkou je kódovanie a zabezpečenie pred neautorizovaným prečítaním obsahu tretou stranou (útočníkom). Okrem šifrovania nám ponúka tieto ďalšie možnosti:

- Bezpečný prenos
- Štatistiky prenosu a iné monitorovacie služby
- Okamžité odosielanie správ

Správy, ktoré sú základnou prenosovou jednotkou WCF služby môžu reprezentovať textový znak, text, ale aj celé súbory prípadne toky dát.

Služba väčšinou funguje na typológii klient - server, kde klient na základe adresy servera sa pripojí a v podobe volania funkcii s ním komunikuje.

Na použitie funkcií, ktoré nám ponúka WCF server, je potrebné, aby aplikácia vo vývojovom prostredí definovala spojenie so serverom pomocou internetovej adresy. Následne, po pripojení na

server, si vývojové prostredie vyžiada všetky funkcie, ktoré má pre klienta sprístupnené, ich parametre a návratové hodnoty. Vďaka ním klient definuje, ktoré funkcie sa za použitia aplikácie zavolajú na serveri a následne sa na pozadí vytvorí asynchrónna požiadavka, ktorá vykoná požadovanú funkcionalitu bez potrebného aktívneho čakania na dokončenie prenosu. Týmto spôsobom sa využívajú viacjadrové procesory, kde každý proces môže riešiť rozdielnu funkcionalitu. Následne sa môžu navzájom čakať k zosynchronizovaniu behu aplikácie. Prípadne nezávisle na sebe vykonávať potrebné výpočty.

2.6 Prism framework

Ďalším rozšírením, ktoré využíva aplikácia je framewok Prism. Je nadstavbou nad WPF pripadne UWP. Pôvodne bol konštruovaný pre WPF a vydaním novej platformy UWP sa prispôsobil. Momentálne ho môžeme využívať v oboch spomínaných druhoch aplikácii. Ponúka základné vzory a štruktúru programu ako postupovať pri návrhu kvalitnej a prehľadnej aplikácie. Prism využíva pri návrhu softvéru návrhový vzor MVVM. Ten sa snaží oddeliť vývoj dizajnu a logiky aplikácie do nezávislých častí, aby ich vývoj mohol prebiehať nezávisle a zároveň sa dal ľahko integrovať do jednej aplikácie.

Ďalšie výhody, ktoré prináša Prism sú navigácie medzi oknami. Služba je vstavaná a implementovaná pomocou kontajnerov, ako bude neskôr vysvetlené.

Obrázok 2.10: Navigácia medzi oknami [9]

2.6.1 Postup softwarovej implementácie

Vývoj softvéru je náročným postupom a k jeho rýchlemu a prehľadnému vývoju sa používajú zaužívané praktiky. V prípade, že dodržíme všetky najdôležitejšie pravidlá daného postupu, je samotný vývoj aplikácie rýchlejší a prehľadnejší. Prehľadnosť je obzvlášť potrebná pri veľkých projektoch, na ktorých sa zúčastňuje niekoľko vývojárov súčasne. Postupy väčšinou rozdeľujú celkovú aplikáciu na niekoľko menších nezávislých častí. Následne sa to využíva pri testovaní, kde vďaka nezávislosti, je možné testovať jednotlivé časti aplikácie osobitne. Medzi najznámejšie a najpoužívanejšie postupy sa dostali MVVM (Model-View-ViewModel), MVC (Model-View-Controller), MVP (Model-View-Presenter)

[9] MSDN, 2016. Dostupné online: https://i-msdn.sec.s-msft.com/dynimg/IC716026.png

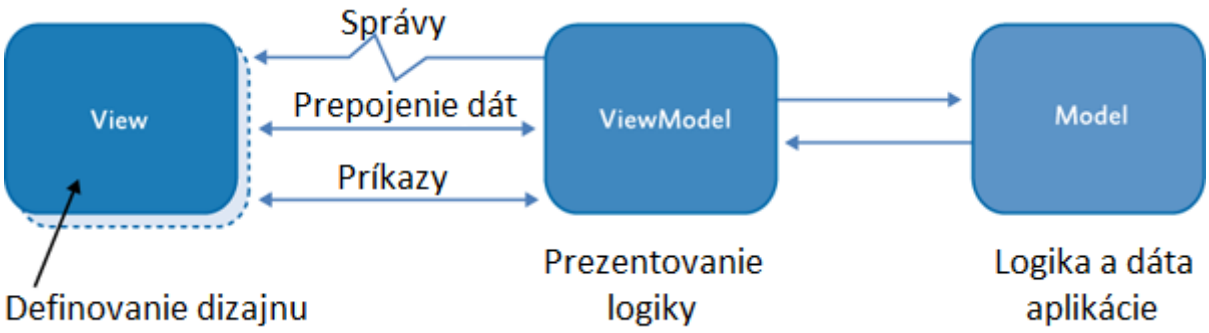
MVVM

Hlavným propagátorom MVVM je Microsoft, ktorý ho presadzoval už v aplikáciách WPF v roku 2005. Postupne sa stal hlavným základom väčšiny aplikácii, ktoré sú spojené s touto firmou. Rovnako aj najnovšia univerzálna platforma je založená na modeli MVVM, ktorý aplikáciu rozdeľuje do troch veľkých celkov.

Jednou zo skupín je časť aplikácie, ktorá reaguje na správanie sa užívateľa a komunikuje s ním je zobrazovacia vrstva nazývaná „View“. V univerzálnych aplikáciách od Microsoftu sa vrstva primárne programuje v značkovacom jazyku XAML a následne sa zobrazovacie dáta prepájajú s ďalšou vrstvou „ViewModel“. Jej hlavnou úlohou je spracovávane dát aplikácie, ich filtrovanie a zobrazenie užívateľovi. Poslednou vrstvou a zdrojom dátových informácii je vrstva „Model“.

Týmto spôsobom sme si rozdelili aplikáciu a môžeme využívať jej výhody:

- Nezávislý vývoj jednotlivých častí
- Nezávislé testovanie
- Spoločné rozhranie na vzájomnú komunikáciu
- Možnosť nahradiť niektoré časti dočasným objektom, ktorý simuluje jej chovanie počas testovania

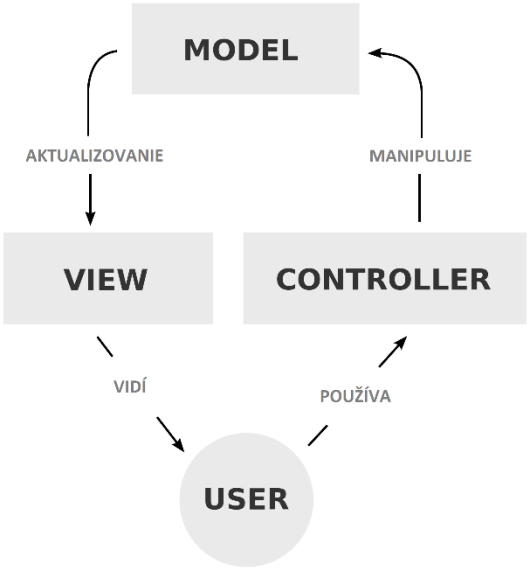


Obrázok 2.11: MVVM [10]

MVC

Obsahuje tri hlavné komponenty a rozdeľuje smer informácii k užívateľovi a od neho. Hlavnou aplikačnou logikou je vrstva „model“, ktorá priamo posiela spracované dáta k zobrazeniu. Prijaté (nové) dáta prijíma vrstva „View“. Vrstva je jednosmerná a dokáže ich len zobraziť. O interakciu užívateľa s programom sa už stará „Controller“, ktorý zbiera jednotlivé gestá a príkazy od užívateľa a posúva ich späť do vrstvy „Model“ k spracovaniu a zobrazeniu relevantnej zmeny podľa požiadavky užívateľa. Zmena je presunutá do vrstvy „View“ a navzájom sa točia dookola, avšak vždy len jedným smerom.

[10] MSDN, 2016. Dostupné online: https://i-msdn.sec.s-msft.com/dynimg/IC448690.png



Obrázok 2.12: MVC [11]

Porovnanie

Okrem dvoch postupov, bol spomenutý aj postup MVP. Je veľmi podobný s MVVM a v projekte nie je využitý, preto sa mu viac venovať nebudeme. Všimneme si MVVM a MVC, kde hlavný rozdiel je spôsob komunikácie. U MVC je komunikácia kruhová a jednosmerná. V MVVM sú tri vrstvy sériovo za sebou a komunikácia prebieha v oboch smeroch. Komunikácia je založená na posielaní správ medzi vrstvami. Druhým rozdielom je používanie. MVC sa väčšinou používa pri webových aplikáciách kde to vývojové postupy MVP a MVVM sa skôr využívajú pri vývoji aplikácii, ktoré sa do počítača inštalujú.

3 Návrh implementácie

Zameriame sa na použitie spomínanej aplikácie do veľkokapacitnej jedálne. Na základe kapitoly 2.4 ako bol opísaný a odôvodnený výber, budeme sa ďalej venovať vývoji aplikácii pod operačným systémom Windows 10 v programovacou jazyku C# s použitím frameworkov UWP a Prism. Okrem nich budeme využívať služby WCF komunikácie po sieti internet, keďže aplikácia je konštruovaná na univerzálne použitie. Vie sa prispôsobiť veľkosti rozlíšeniu a rovnako sa musí dokázať prispôsobiť miestu použitia, čiže jedálni, v ktorej sa bude používať. Na identifikáciu sa bude používať ID zariadenia a adresa servera. Obidve hodnoty sú unikátne v rámci jedného páru aplikácii. Čiže jedna aplikácia klient a druhá servis, budú mať rovnakú adresu servera aj ID. Zabezpečí to, aby sa aplikácie dokázali spárovať, čo umožní, že sa nemôže nachádzať viac aplikácii s rovnakým ID pre daný server. Aplikácia musí byť maximálne autonómna a vyžadovať čo najmenej interakcie od užívateľa, preto je potrebné, aby si daná inštalácia aplikácie pamätala všetky nastavenia. V prípade, že sa jej nepodarí

[11] Wikimedia Foundation, 2005. Dostupné online: https://upload.wikimedia.org/wikipedia/commons/thumb/a/a0/MVC-Process.svg/200px-MVC-Process.svg.png

prihlásiť, musí danú požiadavku zopakovať (môže sa stať že server prestane fungovať, po jeho obnovení sa musí aplikácia opakovane automaticky pripojiť).
Niekedy sa môže stať, že sa zmení adresa servera prípadne ID zariadenia. Je potrebné zachovať možnosť zmeny. Nie je to štandardné používanie a preto je dobré túto možnosť relevantne skryť a zabezpečiť.

3.1 Kontajnery

Výhodou použitia kontajnerov je viacero. Napríklad, ak vytvárame viac aplikácii, ktoré potrebujú rovnakú funkcionalitu. Môžeme ju zabaliť do kontajnera a vyžiadať si ju v každej z aplikácii. Prípadne sa kontajnery dajú využiť, ak máme časovo náročný kód, ktorý aplikácia nemusí potrebovať a vykonal by sa zbytočne, tak sa vykoná až pri skutočnej potrebe niektorej z funkcii daného kontajnera. V prípade, že sa žiadna z funkcii kontajnera nevyžiadala, nevykoná sa ani náročná časť kódu na jeho vytvorenie a tým urýchlime spustenie aj priebeh aplikácie. V prípade, že potrebujeme niektorú z funkcii daného kontajnera, vytvorí sa a začne sa používať.
Často využívanou vlastnosťou, je možnosť definovať správanie kontajnera. Správanie je potrebné nastaviť pri spustení aplikácie. Môžeme ho definovať, aby pri každej požiadavke o použitie sa vytvorila nová inštancia kontajnera, prípadne sa pri prvom zavolaní vytvorí raz a ďalšie volania už len dostanú ukazovateľ, na existujúci kontajner. Spolu s parametrom sa dá definovať aj spôsob vytvorenia kontajnera prípadne metóda, ktorá kontajner vytvorí a pomocou parametrov ho pri vytváraní môžeme upraviť podľa potrebného použitia.

3.1.1 Navigácia medzi oknami a ich prekreslenie

Spomínaný framework Prism má v sebe implementovanú službu „navigation service. Stará sa o navigáciu a prekresľovanie medzi oknami a je implementovaná v kontajneri, kde v celej aplikácii existuje len jedna inštancia a v prípade potreby tejto služby dostaneme ukazovateľ už existujúcej. Kontajner v sebe uchováva históriu stránok a povoľuje navigáciu medzi navštívenými stranami spolu s navigáciou na stranu definovanú menom. Správanie sa veľmi podobá správaniu internetového prehliadača, ktorý dokáže zobraziť obsah stránky podľa zadanej adresy alebo prechod na predchádzajúcu stránku.

3.1.2 Záznam o činnosti aplikácie

Veľmi dôležitá súčasť všetkých aplikácii, o ktorých prevádzku sa starajú externé firmy je záznam o činnosti. Kým aplikácia beží v bezproblémovom behu, túto službu nevyužijeme. V prípade, že sa aplikácia začne chovať neštandardne, jediným zdrojom informácii, čo sa dialo na pozadí aplikácie sú tieto záznamy, keďže väčšinou jedinú informáciu, čo od zákazníka firma dostane je: „Aplikácia prestala fungovať“, prípadne jej odvodené vety.

V takomto prípade si správca aplikácie môže pozrieť informácie z aplikácie a zistiť, v ktorom momente sa jej beh zastavil. Ako pracovala, prípadne, čo sa pokúšala spraviť a priviedlo ju do chybového stavu.

Tu sa dá pekne poukázať na výhody použitia kontajnera. Najprv záznam o činnosti zapisovalo jedno vlákno bez súbežného behu viacerých úloh a nebol problém, že v jednom momente chceli dva procesory zapisovať do rovnakého súboru. Stačilo vytvorenie jednej inštancie služby a postupne sa k nej pripájali a odpájali časti programu. V každom okamihu ho využívala len jedna časť programu. Pri používaní univerzálnych aplikácií UWP, bolo potrebné vyriešiť problém viacerých vlákien, lebo už pri spustení má samotná aplikácia vytvorených niekoľko vlákien, kde každé vlákno má inú úlohu a ak chce mať každé vlákno možnosť zapisovať do súboru, je potrebné mať pre každé vlákno osobitnú službu. Keby aplikácia bola vyvíjaná bez použitia kontajnerov, je potrebné každú časť programu prepísať tak, aby si vždy vytvorila novú inštanciu služby. Vďaka kontajnerom, stačí prepísať spôsob vytvárania kontajnera, čo je zvyčajne pár riadkov kódu v hlavnom súbore. Pôvodne sa vytvorila inštancia kontajnera pri prvom volaní a následne sa posielal ukazovateľ na už existujúcu službu. V novej aplikácii sa bude pri každom volaní vytvárať nová inštancia služby a viaceré vlákna môžu naraz zapisovať záznamy o činnosti do počítača.

Čo program zaznamenáva

Každý záznam o činnosti predstavuje riadok vo výstupnom súbore. Začiatok obsahuje aktuálny čas, kategória záznamu, časť programu, ktorá záznam vytvorila a jej obsah. Príklad záznamu môžeme vidieť na obrázku 3.1

```
4|2016-04-27T07:07:45.6479411+00:00|TRACE|3|App|ClientApp sa uspesne pripojil na server
6|2016-04-27T07:08:45.9586799+00:00|TRACE|3|CardReader|CardReader.StartRead, StartKey = 192
7|2016-04-27T07:08:46.0494343+00:00|TRACE|3|CardReader|CardReader.Read, Key = NumberPad5
8|2016-04-27T07:08:46.1023230+00:00|TRACE|3|CardReader|CardReader.EndRead, Key = Enter
9|2016-04-27T07:08:46.3013607+00:00|TRACE|3|ServeMealPageViewModel|Vydane jedlo pre "Král Karel"
```

Obrázok 3.1: Záznam o činnosti aplikácie

Kategória záznamu definuje dôležitosť. Najčastejšie sa správy delia približne do piatich skupín. Najdôležitejšie sú napríklad chyba alebo varovanie. Najmenej dôležité sú informácie o priebehu programu.

Hlavnou výhodou deliť informácie do kategórii je v prípade, že programátor chce najdôležitejšie správy ukladať na iné úložisko, napríklad server. A všetky informácie o behu programu do lokálneho počítača alebo iné rozdelenie či filtrovanie.

V prípade používania zaznamenávania o behu aplikácie je dôležité myslieť na to, aby týchto informácii nebolo príliš veľa a zároveň, aby ich bolo dostatočne na to, aby sa dal rekonštruovať beh programu. V prípade, že si otvoríme súbor so záznamami, aj neznalý človek vydedukuje čo aplikácia robí a jednotlivé kroky ako to dosahuje.

3.1.3 Dátová vrstva

Aplikácia je navrhnutá v podobe stavového automatu. V každom momente behu je definovaný stav. V tomto stave sa musia nachádzať obe spárované aplikácie, keďže tento stav je zdieľaný medzi viacerými zariadeniami a jeho hodnota je uložená na serveri. Server je zároveň samostatným fyzickým zariadením. Znamená to, že je potrebná komunikácia aj s ním. Na komunikáciu sa používa služba WCF opisovaná v kapitole 2.5. Dátová vrstva sa následne stará o všetky záležitosti potrebné ku úspešnej komunikácii prostredníctvom WCF, keďže jeho docielenie nie je jednoduchou záležitosťou. Počas prenosu je potrebné si uschovávať určité informácie, ktoré sú pre samotný program nepodstatné. Základné informácie ostávajú v tejto vrstve, napríklad meno spojenia, adresu servera, typ aplikácie. Okrem toho je potrebné riešiť aj chybové stavy, ktorých je veľmi veľa pri použití verejnej siete internet, napríklad výpadok spojenia, chyba v prenose, strata dát počas prenosu. Skrytím všetkých spomenutých častí do dátovej vrstvy vzniká pre program abstrakcia nad samotnou komunikáciou. Samotná dátová vrstva sa stará o priebeh komunikácie a v prípade, že sa nepodarí kontaktovať server, bude pokus opakovať pokým nedostane požadovanú odpoveď, alebo ho užívateľ nepreruší.

Napríklad pri potrebe zistenia stavu aplikácie vyžaduje server meno spojenia získane pri prvom nadviazaní spojenia a typ aplikácie, ktorá žiada o odpoveď (klientska alebo servisná aplikácia). Pri použití dátovej vrstvy stačí zavolať funkciu GetAppState bez parametra. Dátová vrstva si dva ďalšie potrebné parametre sama doplní. Týmto spôsobom zaobaľuje všetku funkcionalitu, ktorú ponuka WCF server.

Okrem spomínanej funkcie na zistenie stavu aplikácie ponúka funkcie na získanie nastavení a konfigurácie. Získanie a nastavenie vydávaných druhov jedál. Rozpoznanie užívateľa na základe jeho čísla karty. Vydanie objednaných jedál pre užívateľa. Získanie posledných objednávok a získanie počtu porcií jednotlivých jedál.

3.1.4 Čítačka kariet

Rozpoznávanie stravníka vo väčších jedálňach funguje na princípe bezkontaktných kariet. Karty v sebe obsahujú unikátne číslo, na rozpoznanie užívateľa. Na trhu je dostupných veľké množstvo rôznych kariet. Hlavným rozdelením je typ čítacej hlavy a ich pripojenie k počítaču alebo tabletu.

Aplikácia primárne podporuje USB čítačku. Čítačka po pripojení do počítača sa sama rozpozná a nainštaluje. Následne funguje ako virtuálna klávesnica v počítači, preto ak si otvoríme textový dokument, môžeme si všimnúť, čo všetko čítacie zariadenie posiela do počítača. Po otvorení textového editora a prečítaní priloženej karty, pribudol do textového dokumentu reťazec znakov „`11111111" ukončený klávesov ENTER. Prvý znak, ktorý je jeden zo skupiny špeciálnych znakov, presnejšie je to klávesa pod klávesov esc a nad klávesov tab sa volá „tilda" a záleží na jazykovom nastavení, ako sa klávesa zobrazí v textovom editore. Daný znak patrí klávesnici anglickej. Na

slovenskej a českej klávesnici sa na jej mieste nachádza znak „;“. Po orezaní reťazca od štartovacieho znaku „tilda“ a koncovej klávesy ENTER dostávame výsledné číslo karty.

Jeden z problémov USB čítacej hlavy je fakt, že sa javí v počítači ako virtuálna klávesnica. Znamená to, že po prečítaní priloženej karty je možné nasimulovať stlačením prislúchajúcich kláves, ktoré obsahuje daný reťazec. Túto možnosť je potrebné zablokovať a preto aplikácia obsahuje časovače, ktoré definujú za aký najdlhší čas po prijatí štartovacieho znaku musí prísť prvý znak karty. Druhý časovač, definuje maximálny čas medzi prijatím štartovacieho a koncového znaku. Prislúchajúce hodnoty sme nastavili na 0,25 sekúnd pre prvý znak a 1 sekundu na prijatie celého reťazca. Hodnoty sa javili ako najlepšie pri viacerých testoch. V prípade, že boli časovače nastavené na kratšie hodnoty, stávalo sa, že čítacia hlava nestihla svoju postupnosť znakov preniesť do počítača.

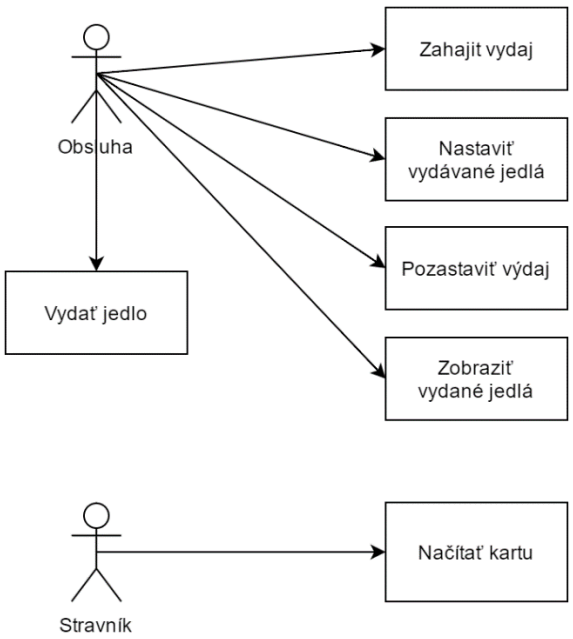
Jednotlivé problémy poukazujú na zložitosť implementovania čítačky karty, ktorá nepatrí do kódu programu ošetrovanie podmienok, ktoré musia byť splnené, aby bolo číslo karty platné, keďže s ním nesúvisí. Preto som ho oddelil a vytvoril službu, ktorá sa stará o obsluhu všetkých systémových udalostí súvisiacich so stlačením klávesy (systém nerozoznáva stlačenie klávesy na klávesnici a simulované stlačenie virtuálnej klávesy v čítačke kariet). Výhodou implementácie čítania kariet cez službu je asynchrónne správanie. Beh aplikácie je nezávislý s obsluhou čítačky kariet a preto je potrebné zabezpečiť aj asynchrónnu odozvu s programom na priloženie karty k čítačke.

Programovací jazyk C# obsahuje v implementácii asynchrónnu komunikáciu cez udalosti. Kde program sa prihlási k udalosti a zadá jej, ktorú obslužnú funkciu má zavolať pri vzniknutí danej udalosti. Následne služba, ktorá má v sebe definované jednotlivé udalosti si pri prihlásení novej obslužnej funkcie k obsluhe zapíše a pri uskutočnení udalosti zavolá všetky obslužné funkcie, ktoré sú zapísané k obsluhe.

Týmto spôsobom je oddelená obsluha čítacieho zariadenia a obsluha aplikácie, kde po priložení karty sa zavolá obslužná funkcia už s orezaným číslom karty pripraveným k spracovaniu.

3.2 Diagram použitia

Obe aplikácie sú určené cielenej skupine užívateľov. K servisnej aplikácii majú prístup užívatelia zo skupiny zamestnancov kuchárov, ktorí sú zároveň aj aktívnymi používateľmi. To znamená, že sa dá očakávať postup práce s aplikáciou, ktorí sa sami naučia, prípadne sa naučia na krátkom školení a ďalej to využívajú. K druhej aplikácii pristupujú stravníci. Stravníci sú zvyčajne rôznych vekových a vedomostných kategórii, preto musí byť aplikácia spravená dostatočne jednoducho pre deti a zároveň dostatočne odborne, aby pokryla užívateľov vysokých škôl a vyučujúcich.



Obrázok 3.2: Stavový diagram

Stavový diagram na obrázku 3.2 zobrazuje, najväčšiu časť interakcie s aplikáciou na servisnej časti. Dokáže ovládať beh oboch aplikácii, povoliť alebo pozastaviť výdaj.

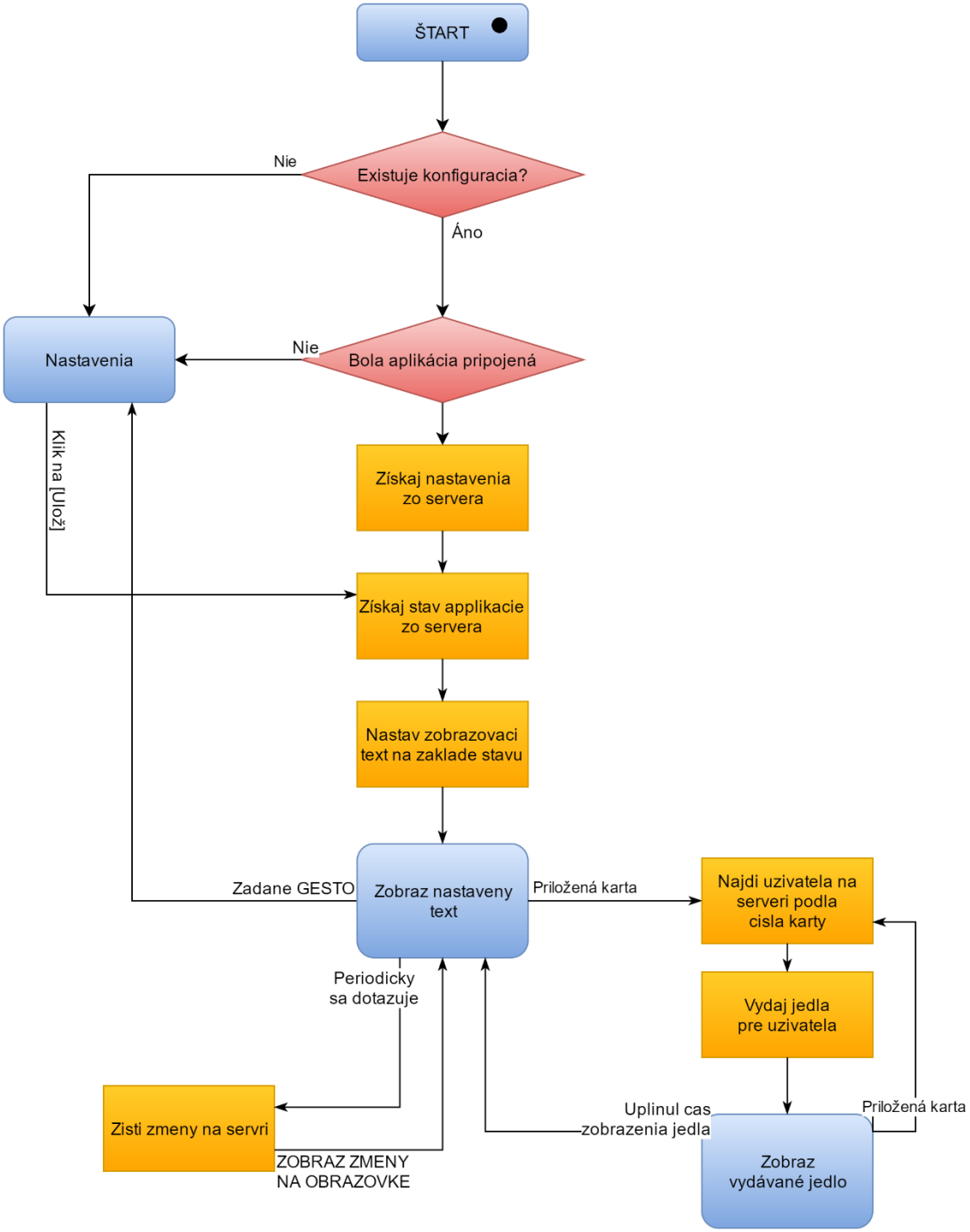
3.3 Stavový diagram

Pod týmto pojmom, v informačných technológiách, definujeme druh diagramu, ktorý opisuje správanie sa systému od jeho začiatku až po jeho vypnutie. Podľa noriem definuje všetky stavy do ktorých sa aplikácia počas svojho života dokáže dostať, spolu s akciami, ktoré vykonáva a spracováva.

Obe aplikácie majú prvú časť diagramu rovnakú. Táto časť predstavuje spustenie aplikácie, ktorá sa pokúsi nájsť uložené nastavenia z predchádzajúceho spustenia. V prípade, že aplikácia nebola spustená, tak sa jej nepodarí žiadne nastavenia načítať a priamo prejde do obrazovky nastavení. Ak aplikácia uložené dáta načíta, a zistí, že posledný stav aplikácie nebol pripojený na server, tak rovnako prejde do nastavení. V tomto prípade už užívateľovi zobrazí hodnoty, ktoré naposledy do nastavení zadal.

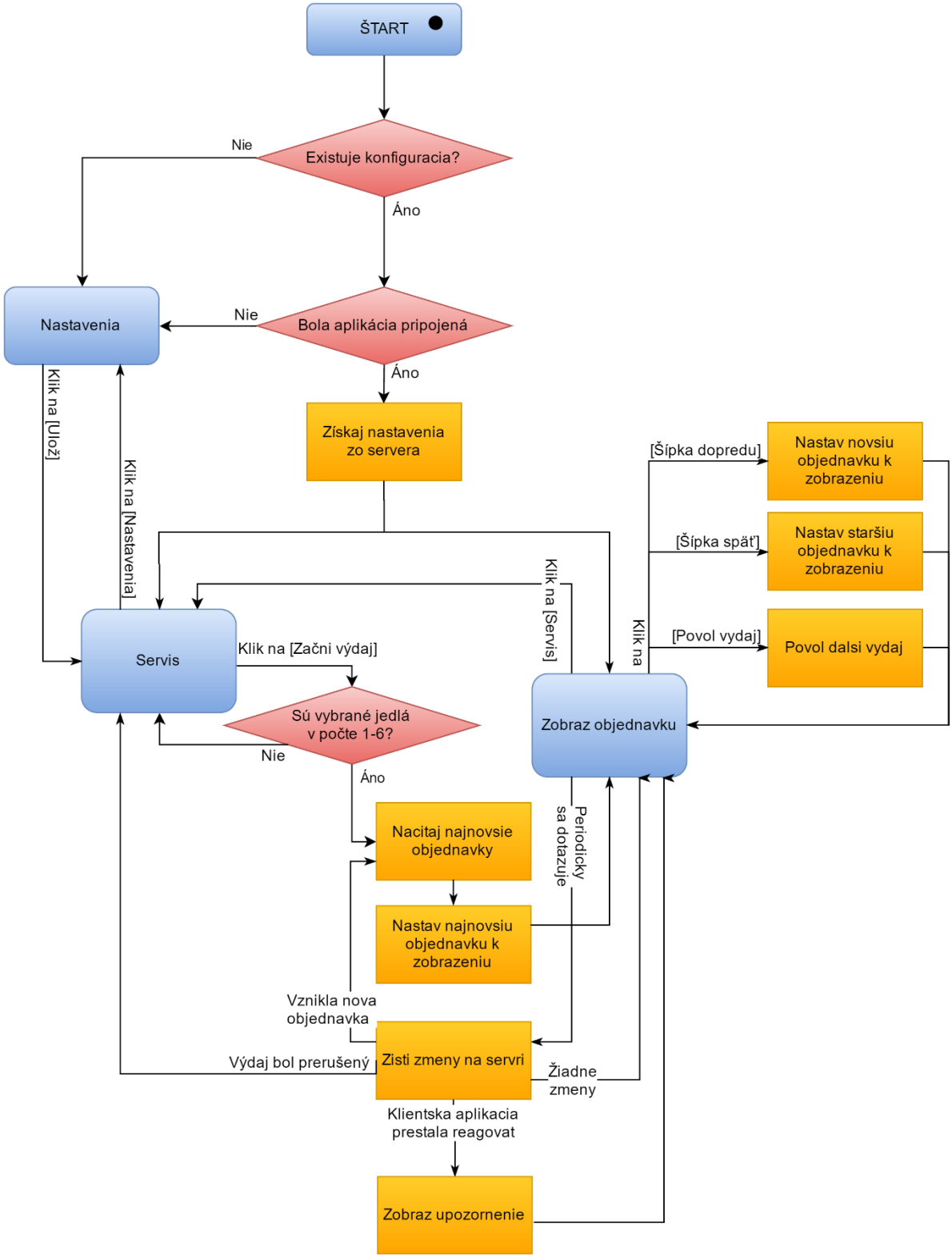
V inom prípade, ak aplikácia už spustená bola, pokúsi sa pripojiť na server. Po pripojení aplikácia zistí konfiguráciu a posledné jeho stavu. Je potrebné, aby obsluha mala čo najmenej interakcie s aplikáciou, preto nezačína aplikácia vždy pri nastavovaní aktuálne vydávaných jedál, ale zistí stav, v ktorom bola pri poslednom spustení a druhy jedál, ktoré boli naposledy vydávané a pokračuje ďalej. Je to hlavne z dôvodu, keď sa aplikácia reštartuje a po štarte sa dostala naspäť do stavu a pokračovala v prevádzke.

Klientska aplikácia



Obrázok 3.3: Stavový diagram klientskej aplikácie

Servisná aplikácia



Obrázok 3.4: Stavový diagram servisnej aplikácie

3.3.1 Servis aplikácie

Tento stav označuje pozastavený výdaj. Obsluha má možnosť nastaviť jedlá na výdaj. Pre klientsku aplikáciu znamená zobrazenie hlášky o pozastavenom výdaji. V prípade servisnej aplikácie je potrebné zobraziť obrazovku s dostupnými druhmi jedál, ktoré môže editovať a nastaviť za vydávané.

3.3.2 Výdaj

Aplikácia je pripravená k vydaniu jedla. To znamená, že klientska aplikácia má povolené priloženie a spracovanie karty. A zároveň servisná aplikácia čaká na nové jedlo.

3.3.3 Nové Jedlo

Tento stav môže nastaviť jedine klientska aplikácia kde užívateľ priložením čipovej karty vydá nové jedlo. Následne servisná aplikácia zistí, že nové jedlo bolo vydané, načíta ho a zobrazí na obrazovku, tým obsluha jedálne získa informáciu aké jedlo má vydať. Po úspešnom zobrazení, vráti stav aplikácie naspäť do stavu „výdaj“ alebo „čakaj na obsluhu“. Táto možnosť záleží na konfigurácii potvrdzovať výdaj jedla .

3.3.4 Čakaj na obsluhu

Tento stav nie je prednastavený a je potrebné ho nastaviť v nastaveniach. Jedná sa o stav, ktorý nasleduje po vydaní jedla a čaká na potvrdenie obsluhou, kedy bude pripravená vydať ďalšie jedlo. Za tento čas je potrebné upozorniť stravníka, aby počkal a zatiaľ neprikladal kartu k čítačke. Po povolení ďalšej objednávky obsluhou sa aplikácia dostane do stavu „výdaj“, kde stravník svoju kartu priložiť môže. Toto nastavenie sa využíva v jedálňach, kde výdaj jedál trvá rozdielne dlho a obsluha potrebuje určitý čas, na obsluhu ďalšej osoby.

3.3.5 Pozastavený výdaj

Týmto termínom sa dajú nazvať všetky ostatné stavy, kedy obsluha servisnej aplikácie pracuje s aplikáciou a za ten čas je potrebné pozastaviť výdaj. Týchto dôvodov na pozastavenie je viacero. V prípade, že si obsluha potrebuje pozrieť posledné výdaje, ktoré boli uskutočnené, prípadne si pozrieť kontakty na zodpovedné osoby starajúce sa o servis aplikácie alebo sa nachádza v iných miestach nastavení aplikácie.

3.3.6 Chybové stavy

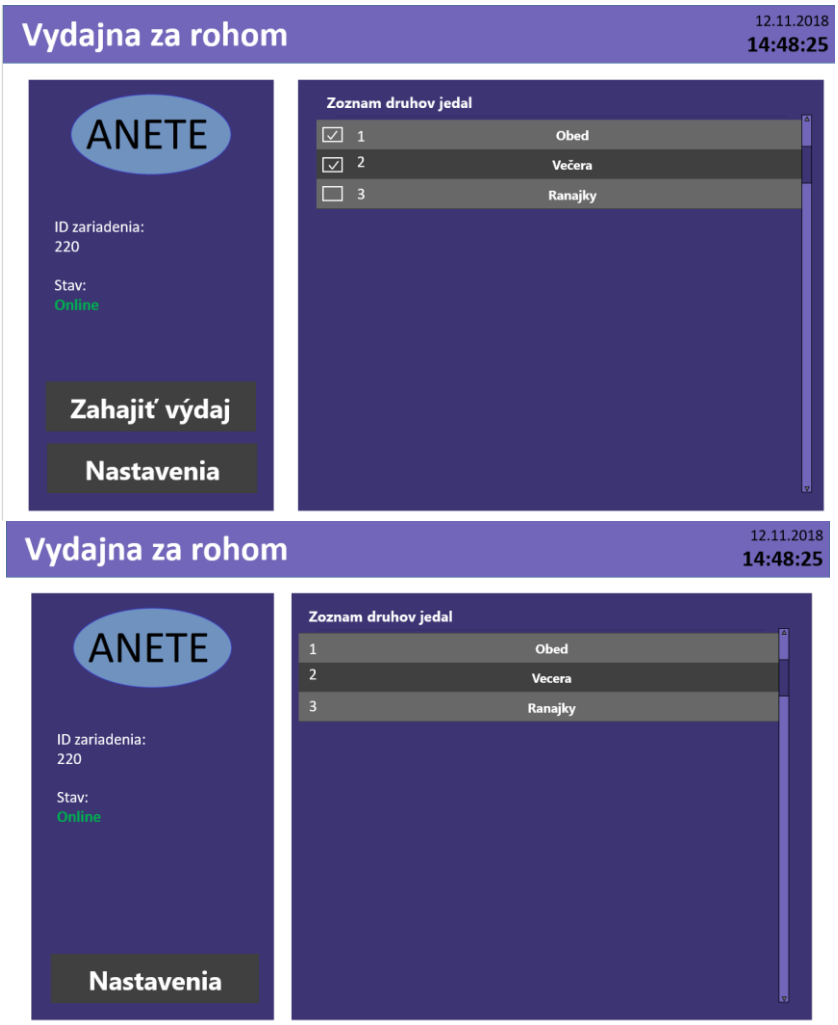
Chybový stav sa považuje neaktivita jednej z aplikácii. Za behu aplikácie sa v každom stave posielajú periodicky informácie serveru, že je aplikácia aktívna a zisťuje zmeny na ktoré treba reagovať. Podľa týchto periodických dotazov je možné zistiť, že aplikácia sa dlhšie neozvala a označiť ju ako neaktívnu. Je to potrebné hlavne v prípade, keď prebieha výdaj jedál a servisná aplikácia prestane komunikovať so serverom. Ak by sa servisná aplikácia vypla a klientska aplikácia by sa o tom nedozvedela, tak by pokračovala vo výdaji a ďalej by vydávala jedlá. No servisná aplikácia by nedokázala zobraziť obsluhe jedlo, ktoré ma vydať a tým by sa stravník k svojmu jedlu nedostal. V opačnom prípade, kedy vypadne klientska aplikácia sa nič vážne nedeje. Ďalšie jedlo nie je možné vydať, keďže klientska aplikácia, ktorá spracováva karty nedokáže povoliť ďalšie jedlo k vydaniu.

Servisná aplikácia dokáže ďalej pracovať s obsluhou, povoľuje zobraziť vydané jedlá a indikuje stav, že klientska aplikácia sa nenašla. Po opätovnom pripojení klienta sa výdaj vráti do pôvodného stavu ako bol pred chybou a prípadne očakáva priloženie karty na spracovanie ďalšieho klienta.

3.3.7 Jednodruhový a viacdruhový výdaj jedla

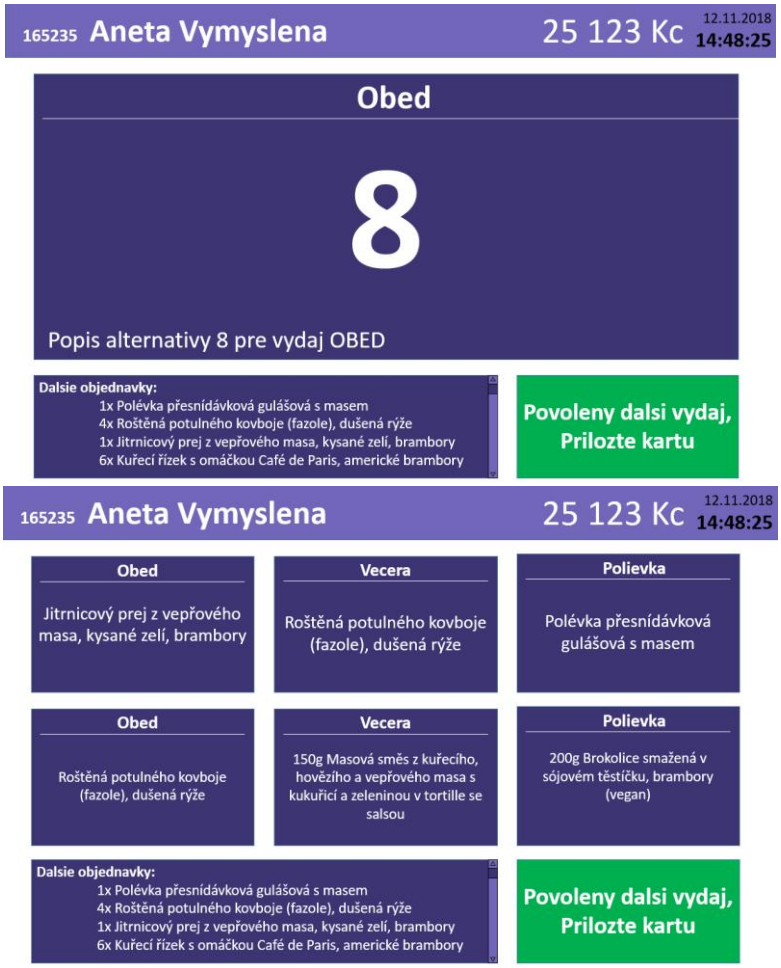
Druh znamená chod jedla, ktoré dokáže jedáleň vydať. Sú to napríklad raňajky, polievky, hlavné jedlá, dezerty, večere. Následne každý druh jedla má svoje alternatívy, medzi ktorými si môže klient objednať. V nastaveniach aplikácie je možné povoliť jednodruhový alebo viacdruhový výdaj. Záleží od druhu jedálne a funkcionality, ktorú potrebujú.

V prípade viacdruhového výdaja sa prispôsobí celý beh programu. Začína pri nastavení druhu vydávaného jedla, kde z pôvodného výberu niekoľkých jedál stačí na požadovaný druh jedla len stlačiť. Pri viacdruhovom výdaji je potrebné odškrtnúť, ktoré druhy jedál sa idú vydávať a následne potvrdiť výber a zahájiť výdaj tlačidlom „Start serving“. Porovnanie dvoch spôsobov môžeme vidieť na obrázku 3.5, kde vo viacdruhovom výdaji pribudol stĺpec s potvrdením, či sa jedlo bude vydávať a tiež pribudlo tlačidlo na zahájenie výdaja.



Obrázok 3.5: Porovnanie viacdruhového a jednodruhového nastavovania výdaja

Okrem zahájenia výdaja záleží na type výdaja aj samotný výdaj. Pri jednodruhovom výdaji stačí prispôsobiť obrazovku jednému jedlu. V prípade viacdruhového výdaja je potrebné mať zobrazovacích častí viac. Aplikácia si sama vyberá, ktorý typ obrazovky sa použije na základe počtu vydávaných jedál, ktoré získa zo servera. Samotný výdaj môžeme vidieť na obrázku 3.6, kde je rozdielny počet vydávaných jedál.



Obrázok 3.6: Porovnanie viacdruhového a jednodruhového výdaja

3.4 Definovanie vzhľadu a štýlov v XAML

Výpočtová logika aplikácie je programovaná v jazyku C#. Jazyk je primárne určený k logickým výpočtom, preto sa grafický návrh aplikácie programuje v XAML značkovacom jazyku, ktorý je lepšie použiteľný v dizajne. Samozrejme jednoduché grafické konštrukcie sa dajú vytvoriť aj priamo v logike aplikácie ale podľa postupu MVVM tam nepatria a vo väčších projektoch sa nevyužívajú. Jeden z dôvodov môže byť, že logická aj dátová vrstva aplikácie môže byť spoločná pre viac zariadení prípadne aj pre rôzne operačné systémy. Rozdiel je vo vrstve zobrazovacej. V takomto

prípade, by sa použilo iné grafické rozloženie prvkov a môžeme aplikáciu používať na väčšom množstve zariadení.

V prípade, že chceme vytvoriť vlastné štylizované objekty, ako napríklad tlačidiel textov, môžeme využiť spoločný štýl, ktorý aplikujeme na jednotlivé objekty. Tento štýl je zdieľaný medzi viacerými objektmi a získavame ucelený tvar jednotlivých častí aplikácie, spolu s možnosťou globálnej zmeny jednotlivých atribútov. Atribúty v textovom objekte môžu byť napríklad farba, veľkosť, či typ písma.

3.4.1 Prepojenie logickej a grafickej vrstvy

Obe vrstvy sú definované v iných programovacích jazykoch, no zároveň je potrebné udržať medzi nimi komunikáciu. Aby sme mohli vykresliť text z logickej vrstvy, priradíme ho do verejnej premennej a po jej nastavení vyvoláme udalosť zmeny. Udalosť odchytí grafická vrstva a prekreslí objekty, ktoré boli zmenou ovplyvnené. Takto môžeme ovládať všetky atribúty grafických prvkov cez logickú vrstvu.

3.5 Grafické animácie

K dosiahnutiu pekného a intuitívneho zobrazenia dopomáhajú tiež animácie, ktoré zdôraznia prvky obrazovky, ktorých hodnoty sa zmenili, prípadne zobrazia presun hodnôt, ktoré sa v rámci obrazovky premiestnili. V neposlednom rade aj samotná aplikácia vďaka animáciám pôsobí estetickejšie.
Štýl samotného zobrazenia a animácie sú definované v jazyku XAML. Ich spustenie môže nastať pri zmene veľkosti okna aplikácie, kedy sa poruší minimálna alebo maximálna veľkosť okna daného zobrazenia a je potrebné prejsť na nové zobrazenie, alebo na základe vyvolania udalosti ako je ukázanie myšou alebo kliknutie ukazovateľom na grafický objekt. Prípadne treťou možnosťou je vyvolanie animácie z logickej vrstvy. Vrstva oznamuje zmeny obsahu zobrazovaných textov na obrazovke cez udalosti a reaguje na udalosti generované logickou vrstvou, ktoré v sebe obsahujú obsah k zahájeniu alebo ukončeniu požadovanej animácie.

3.6 Responzívne zobrazenie

Responzívne zobrazenie, znamená schopnosť aplikácie prispôsobiť sa rozlíšeniu displeja. V prípade, že užívateľ zmení veľkosť okna, je potrebné, aby sa obsah zmenšil na čitateľnú veľkosť. Z tohto dôvodu XAML umožňuje definovať obmedzenia na veľkosť okna a následne sa okno pri prekročení veľkostných limitov usporiada alebo inak prispôsobí podľa daných parametrov. Napríklad menej dôležité dáta sa skryjú, niektoré časti okna sa čiastočne prekryjú a následne pri kliknutí alebo ukázaní na zmenšeninu sa automaticky roztiahnu a ukážu svoj plnohodnotný obsah.

3.7 Návrh grafického rozhrania

Ako sme si do teraz definovali možnosti, grafické zobrazenie prispieva k samotnému zlepšeniu vzhľadu, čo využijeme v samotnej aplikácii. Z veľkej časti úspechu alebo neúspechu aplikácii je dizajn. Je dôležité nájsť správne vyváženie zobrazovaných údajov, aby aplikácia bola prehľadná a čitateľná a zároveň pokryť požiadavky užívateľa, ktoré mal na danú aplikáciu.

Navrhujeme, aby najdôležitejšie informácie, ktoré aplikácia má vždy zobrazovať sme zvolili: meno stravníka, stav na účte, druh jedla a identifikačné číslo alternatívy, ktoré si stravník objednal. Zobrazenie je určené na aplikáciu na displej mobilného telefónu, ktorý by mohol byť použitý v prevádzkach, hlavne vďaka znižujúcim sa cenám mobilných telefónov. Rozloženie prvkov môžeme vidieť na obrázku 3.7.

Obrázok 3.7: Minimálne rozlíšenie

Pri väčšom rozlíšení displeja dostáva aplikácia možnosť zobraziť viac informácii. K potrebným údajom sme pridali informácie, ako je osobné číslo užívateľa, aktuálny čas a ďalšie objednávky na daný deň. Hlavným parametrom pre použitie tohto zobrazenia je zväčšenie šírky displeja. Jeho minimálna hodnota je nastavená na 700px šírky. Vyžaduje to použitie aspoň HD rozlíšenie (1366x768). Šírka je nevyhnutná v hornej lište, kde panel zobrazuje v riadku vedľa seba osobné číslo, meno, stav na účte a aktuálny čas.

Obrázok 3.8: Stredná veľkosť rozlíšenia

Ako tretie zobrazenie, kde zobrazovacieho miesta je dostatok, pri rozlíšeniach FullHD(1920x1080) a viac, bude aplikácia zobrazovať aj názov jedla textovou formou. V predchádzajúcich zobrazeniach bola táto informácia zakódovaná do identifikačného číslo alternatívy jedla ako je zobrazené na obrázku 3.8 číslo alternatívy osemnásť. V prípade, že si jedálen definuje príliž dlhý opis pre dané jedlo, tak bude orezané v možnostiach displeja.

Obrázok 3.9: Zobrazenie pri maximálnom rozlíšení

3.8 Jazykové mutácie aplikácie

Ako bolo spomenuté, značkovací jazyk XAML definuje štýly a usporiadanie objektov, prípadne im priradí statickú hodnotu. V prípade, že aplikácia má vyššie ciele ako uspieť len v jednej krajine, je dôležité umožniť jazykové prispôsobenie systémovému nastaveniu. Všetky textové reťazce, ktoré budú čítať užívatelia je potrebné ponechať v logickej vrstve a podľa systémového nastavenia zariadenia vybrať v správnom jazyku a zobraziť ho. Jazykové reťazce sú uložené v špeciálnych súboroch a na ich preklad sme použili aplikáciu „multilingual app toolkit“.

K fungovaniu jazykových mutácii aplikácie je potrebné definovať východiskový jazyk projektu a následne vytvoriť zoznam dvojíc identifikátora a textu, ktorý bude použitý.

Po spustení, aplikácia načíta všetky textové reťazce z východiskového jazyka a pripraví preklad identifikátorov do nového jazyka. Okrem samotného prekladu textu súčasne označuje jednotlivé preklady atribútmi ako nový, potrebuje kontrolu, preložený alebo finálny, čo pomáha pri hromadnom preklade viacerých ľudí. Takto vytvorené preklady sa uložia do projektu a budú použité v prípade, že zariadenie má v systéme nastavený prislúchajúci jazyk.

Veľkou výhodou použitia externej aplikácie a nie aplikácie priamo integrovanej vo vývojovom prostredí je fakt, že programátor naprogramuje aplikáciu v jednom jazyku a dokáže texty exportovať a nechať preložiť prekladateľovi. Následne si už programátor len načíta výstupný preklad, ktorý bude pri ďalšom preložení projektu k dispozícii.

3.9 Vytvorenie inštalačného programu

Celá aplikácia ako bola spomínaná je vyvíjaná primárne pre operačný systém Windows 10. Operačný systém má všetky aplikácie lokálne uložené v pamäti počítača, preto je potrebné aplikáciu nainštalovať. Podobne ako aj konkurenčné firmy informačných technológii majú vlastný obchod s aplikáciami, aj Microsoft ho vytvoril pod menom „Store“. Je to internetový obchod počítačových aplikácii, ktorý zabezpečuje všetky najdôležitejšie prvky obchodu a odľahčuje vývojárov od propagácie a ďalších vecí spojených s predajom ich produktov.

Ako náhle je aplikácia dokončená a pripravená k zverejneniu, je potrebné nastaviť systémové nastavenia. V nastaveniach projektu je nevyhnutné nastaviť tieto položky:

- Logo aplikácie na lište
- Logo aplikácie v štarte (rôzne veľkosti)
- Orientácia displeja (povolenie na výšku alebo šírku)
- Povolenú platformu (verzia operačného systému, ktorá ju spúšťa)

Po nastavení parametrov môžeme aplikáciu exportovať. Exportované súbory sa dajú priamo nainštalovať cez príkazový riadok, alebo po vytvorení účtu v Microsoft Store je možné ju kúpiť prostredníctvom internetového obchodu.

4 Používanie

V najbližších odstavcov si priblížime spôsob používania vytvorenej aplikácie. Začneme jej inštaláciou, ktorá sa vo Windows 10 zmenila a je možné ju stiahnuť a nainštalovať jedným kliknutím. A následne budeme pokračovať opisom, ako sa samotná aplikácia ovláda a spúšťa.

4.1 Inštalácia

Inštalácia je primárne vytvorená na inštalovanie cez aplikáciu Microsoft Store. Aplikácia sa vyhľadá podľa názvu a jedným kliknutím sa stiahne a nainštaluje automaticky do lokálneho úložiska. V prípade, že chceme aplikáciu odskúšať a nechceme ju zverejňovať v obchode Store, dá sa aplikácia nainštalovať aj príkazovým riadkom. Tento spôsob je o niečo náročnejší a neštandardný. Je potrebné nastaviť operačný systém do vývojárskeho módu, nainštalovať certifikát, ktorý je súčasťou inštalačných súborov. Na záver cez príkazový riadok „Power Shell“ spustiť inštalačný súbor, ktorý obsahuje koncovku „.appx“. Ten skontroluje všetky potrebné závislosti na beh aplikácie. V prípade úspechu nainštaluje aplikáciu a vytvorí odkaz na spustenie.

4.2 Spustenie a práca s aplikáciou

Po úspešnej inštalácii sa aplikácia nachádza v počítači a jej odkaz sa dá nájsť v štarte pod možnosťou „všetky programy“. Po prvom spustení aplikácie si aplikácia vyžiada nastavenie úvodných parametrov ako je adresa servera a identifikačné číslo zariadenia. Vďaka týmto dvom unikátnym číslam sa dokážu obe aplikácie spolu spárovať cez server a spolu sa podieľať na fungovaní výdaja v jedálni.

Okrem základnej funkcionality, kde aplikácia reaguje na požiadavky užívateľa je tiež možné ju ovládať externe prostredníctvom internetu. Aplikácia podporuje zmenu stavu zo vzdialeného miesta. Zdrojom môže byť aktuálny čas, obsluha alebo firma, ktorá sa o beh aplikácie stará. Vďaka vzdialenému ovládaniu je možné kedykoľvek získať alebo aj zmeniť stav aplikácie, čo dokáže obnoviť alebo prerušiť výdaj jedál bez toho, aby sme fyzicky aplikáciu obsluhovali.

Využití má viac, napríklad pri potrebe vzdialenej pomoci alebo časové zahájenia a ukončenia jednotlivých vydávaných jedál. Rovnako je možné vzdialene zmeniť druhy vydávaných jedál. Využiť to môžeme v zariadeniach, kde sa výdaj jedál začína v presne stanovený čas. V takom prípade aplikácia sama zaháji rôzne druhy jedál v rôzny čas, prípadne počas vydávania dokáže upraviť vydávané jedlá v závislosti na požiadavkách.

5 Testovanie aplikácie

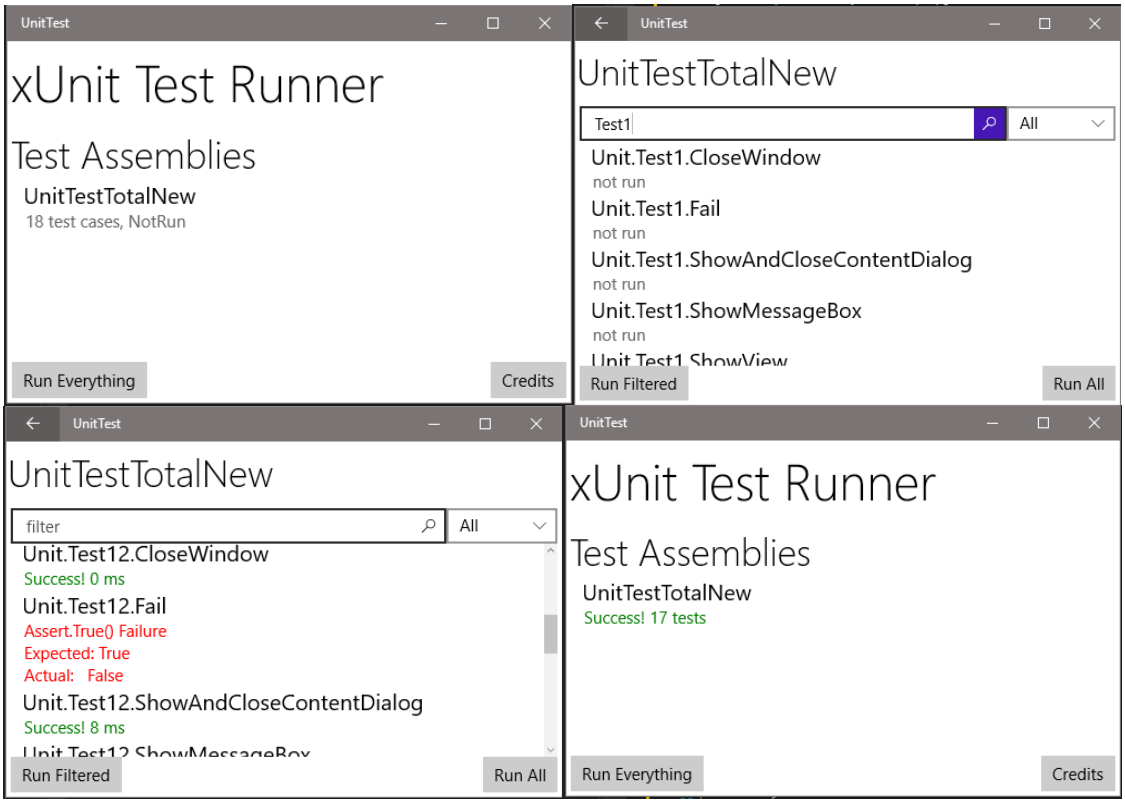
Pred akýmkoľvek zverejnením aplikácie je dôležité ju najskôr otestovať. Obzvlášť viacplatformové aplikácie je potrebné otestovať dôkladnejšie, aby boli spustiteľné na každom druhu zariadení, ktoré majú podporovať. Rovnako je dobré otestovať rôzne verzie systému, kde v prípade staršieho operačného systému môže zmena zabrániť spusteniu alebo správnemu chodu aplikácie. Keďže testovanie spôsobom spustiť a konštatovať či ide alebo nejde je veľmi nepraktické a v podstate nepotrebné. Vyžaduje si testovanie rôznych krajných stavov, do ktorých sa aplikácia môže za behu dostať. Na testovanie s najvyššími požiadavkami, sa využívajú „Unit“ a integračné testy, ktoré zabezpečia bezproblémový chod aplikácie.

5.1 Unit testy

Termín je prevzatý z angličtiny a znamená, že testovací program vytiahne jednu časť programu a dodá mu rôzne dáta, čím simuluje beh aplikácie. Následne kontroluje výstupy s referenčnými hodnotami. Vytvorené testy je možné spustiť vždy po vylepšení aplikácie a skontrolovať, či zmena nespôsobila aj zmenu potrebných funkcii.

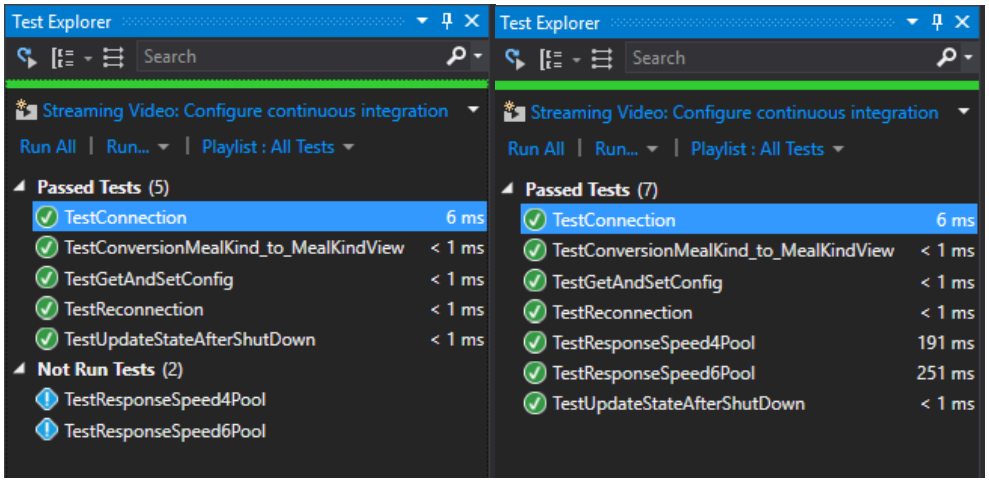
Všetky testy by sme navrhli rozdelil do dvoch skupín, čo pokrýva testovanie grafického rozhrania a interakcie užívateľa na počítač. Vrstva „View“ v modeli MVVM ako bolo vysvetlené v kapitole 2.6.

Tieto testy obsahujú rôzne problematiky spojené s vykresľovaním objektov na obrazovku a interakcie s nimi, prípadne navigáciu medzi oknami.



Obrázok 5.1: Príklad unit testov na grafické rozhranie

Druhou skupinou testov sú testy zamerané na aplikačnú logiku. Táto zväčša spracováva dáta a testuje vytvorením objektu dát, ktoré sú danej vrstve podhodené. Následne sa sleduje či táto vrstva relevantne odpovedala na dáta. Do aplikačných testov patria obe zvyšné vrstvy modelu MVVM.



Obrázok 5.2: Príklad unit testov na aplikačnú logiku

5.2 Integračné testy

Ďalšou možnosťou testovania aplikácii sú integračné testy, ktoré kontrolujú funkčnosť a komunikatívnosť medzi jednotlivými komponentmi aplikácie, prípadne medzi aplikáciou a operačným systémom. Ako bolo spomenuté v kapitole 5.1, kde sme testovali internú funkčnosť

jednotlivých komponent. Je nevyhnutné otestovať celkové komponenty medzi sebou. Integračné testy nie sú nevyhnutnou časťou celkového testovania aplikácie. V prípade, že v aplikácii je chyba na úrovni testov a nebude odhalená, nemá vplyv na celkové testovanie. Pretože chyby odhalia neskoršie testovania celkovej aplikácie. Avšak všetky skoršie odhalenia chýb vedú k rýchlejším a menej náročným úpravám ako by to bolo v záverečných fázach projektu.

Visual štúdio ponúka pri exportovaní aplikácie vlastné automatické testy na integritu. Testy kontrolujú podporu behu aplikácie pod rôznymi verziami operačného systému Windows. Testami bola skontrolovaná funkčnosť oboch aplikácii a predpokladá sa pripravenosť aplikácii k nasadeniu do skutočnej prevádzky.

6 Rozšírenia aplikácie

Ako pri každom vývoji aplikácii, vždy je čo upravovať a vylepšovať. Rovnako sa aj v tejto aplikácie nájde veľké množstvo zlepšení výsledného produktu. Funkcie, ktoré aplikácia ponúka nie sú ničím obmedzené, no zároveň majú byť jednoduché, nakoľko slúžia obsluhe. Zjednodušujú výdaj a nie je potrebné dávať do aplikácie funkcionalitu, ktorá priamo s použitím nesúvisí. Hlavne pri aplikácii, ktorá má jasný a nie moc náročný cieľ: výdaj jedál. To znamená ak aplikácia bude obsahovať zbytočné možnosti nastavení ako je zmena pozadia, typu písma a farebných variácii, dávame možnosť obsluhe k rozptyľovaniu pozornosti.

Obrázok 6.1: Raspberry Pi 2 [12]

Aplikáciu sme rozšírili iným smerom. Vybrali sme si platformu, ktorá patrí spoločnosti Microsoft a je viazaná na operačný systém Windows. Prináša výhody aj nevýhody. Nevýhodou je nutnosť použitia výhradne zariadení s potrebným operačným systémom. Systém je platený a prináša ďalšie náklady na používanie a inštaláciu v nových prevádzkach. Vyriešil to nový operačný systém Windows 10, ktorý nie je viazaný na procesory s architektúrou 32 alebo 64 bit ako to bolo doposiaľ a jeho použitie sa rozšírilo aj na mobilné zariadenia, v ktorých je typicky použitá architektúra ARM. Veľkou výhodou sú hlavne tablety s architektúrou ARM, sú ľahšie a tenšie vo svojej veľkosti. Nás však obmedzujú zariadenia s operačným systémom Windows 10. Ale okrem Windows 10 vyšiel tiež Windows 10 IoT (Internet Of Things). Tento operačný systém je odľahčenou verziou plného Windows 10. Ponúka minimálnu systémovú funkcionalitu, čím znižuje nároky na výkon zariadenia,

[12] Wikimedia Foundation, 2005. Dostupné online: https://wiki.openwrt.org/_media/media/raspberry_pi_foundation/rpi2b.jpg

ale zároveň stráca možnosť každodenného používania na počítačoch alebo notebookoch. Jeho použitie je mierené iným smerom. Je to smer aplikácii, ktoré nepotrebujú veľký výpočtový výkon a často krát fungujú celé dni bez zastavenia alebo reštartovania. Typickým použitím je monitorovanie prostredia alebo ovládanie teploty a vykurovania v domácnosti. Práve toto vyzerá ako ideálne riešenie pre našu aplikáciu. Ako som zatiaľ zistil, tento operačný systém je možné nainštalovať na zariadenie Raspberry PI 2 zobrazené na obrázku 6.1, ktoré ponúka dostatočný výkon aj zdroje k behu, čo by bolo zlepšením našej aplikácie. Obsahuje v sebe USB porty na pripojenie čítačky a tiež WiFi aj sieťovú kartu na pripojenie sa k serveru. Toto zariadenie je len procesor s rôznymi výstupmi. Potrebné je k tomu dokúpiť ešte dotykovú obrazovku. Toto riešenie by mohlo predstavovať finančne nenáročnú náhradu počítača alebo tabletu s nenáročnou montážou, ktorá v prípade tabletov bude pravdepodobne problémová.

Obrázok 6.2: Pracovná plocha operačného systému Windows 10 IoT [13]

Spomínaným rozšírením na Windows 10 IoT by sme sa dokázali oddialiť od drahších zariadení, avšak ich výkon nemusí byť dostatočný. Existuje ešte ďalšia možnosť rozšírenia, ktorú máme záujem v budúcnosti odskúšať. Firma Xamarin vyvíjala možnosť využitia vrstiev „Model“ a „View-Model“ z MVVM modelu vytvorených pre Windows, aj pre iné operačné systémy ako je Android do Google alebo IOS od Apple. Následne pre jednotlivé operačné systémy dodefinovať tretiu vrstvu „View“. Produkt bol veľmi drahým riešením. V poslednom období vyšla správa, že Xamarin bol kúpený firmou Microsoft a distribúciu ponúka zadarmo. To znamená možnosť všetkých aplikácii, ktoré budú primárne vyvíjané pre Windows, či už v jazyku C# alebo C++, spustiť aj na tabletoch či mobiloch s operačným systémom Android alebo IOS.

Hlavnou výhodou by bol operačný systém Andorid, keďže zariadenia s týmto systémom sa v súčasnej dobe pohybujú aj za štvrtinovú cenu v porovnaní so zariadeniami od Apple. Je potrebné overiť, či sú všetky využívané funkcie dostupne a to hlavne u systémových funkcii, keďže veľkým rozdielom medzi Windows a operačnými systémami založenými na Linuxe je spôsob ukladania dát na disk.

[13] Aolcdn, 2016. Dostupné online: http://o.aolcdn.com/hss/storage/midas/71465bd66f5558f952d3e15f39b31217/201931120/windows-10-iot.jpg

7 Záver

Softvérové riešenia podliehajú vývoju aplikácií, ktoré môžeme prispôsobovať a zlepšovať na nové vzniknuté podmienky. Vývoj aplikácie prebiehal postupne. Vytvorila sa základná aplikácia s minimálnou funkcionalitou a následne sa postupne pridávali ďalšie funkcie. Aplikácia bola vyvíjaná na pomerne novej platforme, ktorá aktuálne je zverejnená necelý rok. Pri začiatkoch bola v testovacom vývoji, z toho vyplývalo veľké množstvo problémov a nedostatok dokumentácie k danej problematike. Aktuálne je veľké množstvo ukážkových programov, ktoré využívajú hlavne základnú funkcionalitu. V prípade, že by sme chceli implementovať určité špecifické funkcie, často sa stretneme s problémami a zníženou podporou. Počas vývoja aplikácie sme zaznamenal tri veľké zmeny. Prvá pri začiatku, kde pre operačný systém Windows 10 vyšla veľká aktualizácia. Následne počas vývoja aplikácie vyšla veľká aktualizácia vývojového frameworku Prism, ktorá riešila mnohé problémy a nejasnosti spojené s pôvodnou podporou pre WPF aplikácie a jej prechod na novú platformu UWP. Väčšina zmien boli prosperujúce a žiadané. Rovnako s nimi prišla aj ďalšia funkcionalita, ktorú tento framework ponúkal. Ako tretia zmena počas vývoja prišla aktualizácia vývojového prostredia Visual Studio 2015. Bola veľmi žiadaná, kde sa prostredie viacej prispôsobilo na novú platformu UWP. Výsledkom bolo hlavne lepšie opravovanie syntaxe v kóde a lepšie prepojenie vrstiev MVVM modelu. Pôvodne nedokázalo vývojové prostredie rozpoznať všetky ich spojenia. Pre komunikáciu a zmenu stavu aplikácie bolo zvolené periodické dotazovanie sa na server. Interval, ako často budú tieto dotazy chodiť sa dá nastaviť. Testovaním sa zistilo, že jeden dotaz v prípade, že je server umiestnený na lokálnej sieti zaberie v priemere 12.8 milisekundy a zaberá len malé množstvo kilobajtov. Veľkosť záleží na druhu dotazu, či sa zisťuje stav aplikácie alebo požiadal server o dáta. Na základe testov je vo východiskovom stave nastavený interval dotazovania 250 milisekundy. Predpokladá sa za dostatočne rýchle vzhľadom na odozvu a zároveň je dostatočne pomalé, aby aplikácia zahlcovala lokálnu sieť a tým spôsobovala spomalenie pripojenia.

Celkovým výsledkom práce sú dve aplikácie naprogramované v programovacom jazyku C#, ktoré k behu vyžadujú aplikačný a databázový server. Ponúkajú efektívnejšie a finančne nenáročnejšie riešenie pre užívateľov vo veľkokapacitných jedálňach s terminálom na výdaj jedál.

8 Literatúra

[1] ERICH GAMMA .. *Design patterns: elements of reusable object-oriented software*. 2., rev. ed. S.l.: Pearson Educ, 2000. ISBN 978-020-1485-370.

[2] SMITH, Josh. Advanced MVVM. Josh Smith. Pittsburgh, USA, 2013. ISBN 9780985784546.

[3] GUAY PAZ, José Rolando. *Beginning ASP.NET MVC 4*. New York: Apress, 2013. ISBN 14-302-5752-0.

[4] MACDONALD, Matthew. *Pro WPF in C# 2010: Windows presentation foundation in .NET 4*. New York, N.Y.: Distributed to the book trade worldwide by Springer-Verlag, c2010. Expert's voice in .NET. ISBN 14-302-7205-8.

[5] LÖWY, Juval. *Programming WCF services: elements of reusable object-oriented software*. 3rd ed. Sebastopol, CA: O'Reilly, 2010. ISBN 978-144-9399-429.

[6] Model–view–presenter. In: Wikipedia: the free encyclopedia [online]. Wikimedia Foundation, 2008, posledná úprava 27.2.2016. Dostupné z: https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93presenter

[7] Introduction to UWP app design. In: Microsoft MSDN [online]. Microsoft, 2015, posledná úprava 14.4.2016. Dostupné z: https://msdn.microsoft.com/en-us/windows/uwp/layout/design-and-ui-intro

[8] Dependency injection. In: Wikipedia: the free encyclopedia [online]. Wikimedia Foundation, 2005, 28.4.2016. Dostupné z: https://en.wikipedia.org/wiki/Dependency_injection

[9] Prism. In: Microsoft MSDN [online]. Microsoft, 2014. Dostupné z: https://msdn.microsoft.com/en-us/library/ff648465.aspx

9 Zoznam skratiek

UWP	Universal Windows Platform – univerzálna platforma od firmy Microsoft, ktorá zabezpečuje beh aplikácii na rôznych zariadeniach, od malých mobilných telefónov po počítače a premietacie plátna.
WPF	Windows Presentation Foundation, predchodca UWP s obmedzeniami na špecifickú platformu.
WCF	Windows Communication Foundation, služba ktorá zabezpečuje komunikáciu cez sieť internet s okamžitou odozvou.
MVVM	Model View View-Model, spôsob ktorý udáva kostru ako má vývoj softvéru prebiehať.
MVC	Model View Controller, spôsob ako má vývoj softvéru prebiehať a ako sa má deliť. Väčšinou sa používa vo webových aplikáciách.
MVP	Model View Presenter. Rovnako ako MVVM definuje kostru vývoja softvéru ť. Obidva postupy sú veľmi podobné. Prvé bolo zverejnené MVP a následne na jeho základoch bol vytvorený model MVVM.

10 Zoznam anglických výrazov

Framework	(tiež „Software Framework“ v preklade rámec) je abstrakt nad vývojom aplikácie, ktorý ponúka vývojárovi svoje funkcie. Tie sú väčšinou zjednodušením zložitejších operácii za cieľom zjednodušiť vývoj a oddialiť programátora od hardvérovej závislosti.
Unit testy	(v preklade jednotkové testy) Unit je jednotka alebo časť. Unit testy sú testy smerované na jednotlivé časti aplikácie, kde sa vyberie časť aplikácie a zvyšok sa simuluje do takej miery, aby testy mohli prebehnúť.
Responzívnosť	schopnosť sa prispôsobovať prostrediu