Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Bakalářská práce

Vývoj CMS aplikací ve WPF s využitím .NET

Jan Palyza

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Jan Palyza

Informatika

Název práce

Vývoj CMS aplikací ve WPF s využitím .NET

Název anglicky

Development of CMS applications using WPF .NET

Cíle práce

Bakalářská práce se zabývá analýzou, návrhem a vývojem aplikace pro platformu Windows Presentation Foundation (WPF). Aplikace slouží pro správu webového obsahu (CMS) a podporu pro manažerské rozhodování v rámci získávání statistik.
Hlavními cíli práce je zaměření se na analýzu požadavků zadavatele aplikace, tvorbu modelů požadavků a grafický návrh. Dílčím cílem je rozbor a popis struktury kódu a popis metodiky návrhu samotné aplikace.

Metodika

Na základě studia odborných informačních zdrojů, technických specifikací, analýzy zadavatelských požadavků bude nejprve vypracován specifikační model požadavků. Tento model obsahuje všechny zadavatelská očekávání od aplikace. Dle kritérií budou prozkoumána možná existující řešení, vytvořeny diagramy případů užití nebo diagram tříd. Dále bude sestrojen návrh uživatelského rozhraní a specifikována volba vývojové technologie.

Při implementaci budou využity všechny poznatky analýzy a návrhů. V této fázi budou hlouběji popsány specifické části aplikace s ukázkou kódu.

Testování aplikace se zaměří na stabilitu a rychlost jádra, ale i na správnou synchronizaci aplikace s externími datovými servery, anebo na stabilitu uživatelského rozhraní (view model).

Doporučený rozsah práce

35-40 stran

Klíčová slova

WPF, XAML, redakční systém, .Net, C#, .NET Framework, vývoj, API, UML, analýza, návrh, MVVM

Doporučené zdroje informací

Andrew Troelsen; Pro C# 5.0 and the .NET 4.5 Framework ; Vyd. 6.; New York: Apress, 2012; ISBN 9781430242338

ARLOW, J. – NEUSTADT, I. *UML 2 a unifikovaný proces vývoje aplikací : objektově orientovaná analýza a návrh prakticky.* Brno: Computer Press, 2007. ISBN 978-80-251-1503-9.

Darryl Grove; Programování aplikací pro vícejádrové procesory; Vyd. 1.; Brno: Computer Press, 2011; ISBN 9788025134870

Gary McLean Hall ; Adaptive Code via C#; Vyd. 1.; Albuquerque: Microsoft Press, 2014; ISBN 9780735683204

Charles Petzold; Mistrovství ve Windows Presentation Foundation; ISBN: 9788025121412

Jon Skeet; C# in Depth; Vyd. 3.; New York: Manning Publications, 2013; ISBN 9781617291340

Michael Howard, David LeBlanc ; Bezpečný kód; Vyd. 1.; Brno: Computer Press, 2010; ISBN 9788025120507

Trey Nash; C# 2010; Vyd. 1.; Brno: Computer Press, 2010; ISBN 9788025130346

Předběžný termín obhajoby

2016/17 LS – PEF

Vedoucí práce

Ing. Jiří Brožek, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 13. 3. 2017

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 13. 3. 2017

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 14. 03. 2017

Poděkování

Rád bych touto cestou poděkoval vedoucímu práce Ing. Jiřímu Brožkovi, Ph.D. za pomoc a konzultace při jejím zpracování a Adéle Navrátilové za korekci textu.

Čestné prohlášení

Prohlašuji, že svou diplomovou práci „Vývoj CMS aplikací ve WPF s využitím .NET“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 14. 03. 2017 _____

Vývoj CMS aplikací ve WPF s využitím .NET

Souhrn

Téma práce je analýza a návrh desktopové aplikace v prostředí WPF a její následná implementace. V první části je vysvětlena teoretická část, na základě, které je dále postupováno v praktické části práce. Praktická část řeší analýzu problematiky pomocí metodik Unified Process. Práce se zabývá stanovením požadavků funkcionality aplikace, které jsou následně modelovány UML diagramy. Implementace se zaměřuje na programování aplikace s využitím .Net Frameworku a jazyku C#, dále byl použit návrhový vzor Model-View-View-Model. Přínosem práce je navržená aplikace, splňující požadavky zadavatele a zároveň dojde k usnadnění administrační práce zadavatele.

Klíčová slova

WPF, XAML, redakční systém, .Net, C#, .NET Framework, vývoj, API, UML, analýza, návrh, MVVM

Development of CMS applications using WPF .NET

Summary

The subject of this work is the analysis and design of desktop application in WPF and its subsequent implementation. The first part explains the theoretical part based on which is further processed in the practical part. The practical part solves the problems of analysis using the Unified Process. This work applies the requirements of the functionality of applications that are subsequently modelled in UML diagrams. Implementation is dedicated to programming applications using .NET Framework and the C # language, also used design pattern Model-View-View-Model. The benefit of this work is designed application, fulfilling the requirements of the customer and at the same time it will facilitate administrative work of the submitter.

Keywords

WPF, XAML, CMS, .Net, C#, .NET Framework, development, API, UML, analysis, design, MVVM

Obsah

1 Úvod

Prezentace a zviditelnění je nedílnou součástí každé společnosti, bez které se dnes již málokdo obejde. Pro tyto účely neexistuje nic efektivnějšího a dynamičtějšího než Internet. Pro udržení zdraví podniku je to nikdy nekončící proces, který nejde vést jednostranně. V počátcích marketingu na internetu sloužily webové stránky, které díky své praktičnosti fungují do dnes a neustále se rozvíjejí. Díky pochopení síly, jakou reklama na internetu zastupuje, vznikala podpora propagace reklam na sociálních sítích, například Facebook nebo Google. Obě tyto společnosti zavedly jednoduchou, ale přitom účinnou podporu firem na jejich mediu, kdy za poplatek jsou ochotní poskytnou obrovskou škálu možností, jak lépe propagovat svou firmu.

Z pohledu zákazníka tato forma prezentace přináší velmi lukrativní místo získávání, filtrování a vyhodnocování informací, pro kterou společnost se rozhodnout. Kde mohou najít nejblíže jejich místu požadovanou společnost, a tak dále. Tyto informace jsou zaneseny do databáze přímo konkrétní společnosti.

Tento důvod potřeby zviditelnit se, přinutilo malé rodinné květinářství k vytvoření webových stránek a Facebookového profilu. Jenomže správa těchto dvou prostředí je z časového hlediska a kapacit dost náročná. Díky pokročilé technologii aktualizace Facebooku je opravdu jednoduché, postačí pouze aplikace v mobilním zařízení. Hlavním problémem je špatné udržování webových stránek, díky komplikovanému nahrávání nového obsahu, nepřehledné získávání statistik z obou prostředí a jeho následné porovnávání. To vše vede ke špatnému využití těchto technologií, potřeba více času a demotivace z pohledu vlastníka.

Hlavní náplní projektu je zajistit sjednocené, uživatelsky přijatelné rozhraní. Zaměřeno bude převážně na správu obsahu a rychlý přehled statistik. Cílem je tedy zaujmout stabilnější postavení firmy na trhu a lépe zhodnotit a následně využít cenná data pro marketingové účely.

Dokument je psaný návrhovou implementační formou. V průběhu všech kapitol se seznámíme s problematikou, její analýzou aktuální situace a poté návrhem možného řešení. Dále budeme zasvěceni do implementace celé aplikace a průběhu veškeré problematiky a následné testování aplikace v provozu. V závěru si představíme možné vize budoucnosti, jak a kam by se tento software mohl ubírat a další jistý vývoj, který nás čeká a nemine.

2 Cíl práce a metodika

2.1 Cíle práce

Bakalářská práce se zabývá analýzou, návrhem a vývojem aplikace pro platformu Windows Presentation Foundation(WPF), knihovna tříd grafického rozhraní, která je součástí .NET Frameworku 3.0+. Aplikace slouží pro správu webového obsahu (CMS) a podporu pro manažerské rozhodování v rámci získávání statistik.

Hlavními cíli práce je zaměření se na analýzu požadavků zadavatele aplikace, tvorbu modelů požadavků a grafický návrh. Implementace návrhu bude obsahovat samotný vývoj aplikace a následné testování. V samotném závěru práce budeme diskutovat ohledně možnosti dalšího vývoje programu. Aplikace je zaměřená na cíle zákazníka, tzn.: intuitivní, čistý design, uživatelsky přívětivý, a hlavně praktický pro potřeby uživatele. Dílčím cílem je rozbor a popis struktury kódu a popis metodiky návrhu samotné aplikace.

2.2 Metodika

Na základě studia odborných informačních zdrojů, technických specifikací, analýzy zadavatelských požadavků bude nejprve vypracován specifikační model požadavků. Tento model obsahuje všechny zadavatelská očekávání od aplikace. Dle kritérií budou prozkoumány možné existující řešení, vytvořeny diagramy případů užití nebo diagram tříd. Dále bude sestrojen návrh uživatelského rozhraní a specifikovaná volba vývojové technologie.

Při implementaci budou využity všechny poznatky analýzy a návrhů. V této fázi budou hlouběji popsány specifické části aplikace s ukázkou kódu.

Při testování aplikace se zaměříme na stabilitu a rychlost jádra, ale i na správnou synchronizaci aplikace s externími datovými servery, anebo na stabilitu uživatelského rozhraní (view model).

3 Teoretická východiska

3.1 UML a unifikovaný proces

V knize [1] autoři hovoří o UML jako o univerzálním jazyku sloužící k vizuálnímu modelování systému. Převážně je spojován s modelováním objektově orientovaného softwaru, díky zabudovaným rozšiřovacím mechanizmům má daleko širší možnosti využití.

Jazyk UML neposkytuje žádné druhy metodiky modelování, pouze poskytuje vizuální skladbu pohledů, které je možné využít při sestavování vlastních modelů. Avšak proto se zde používá unifikovaný proces, který metodikou již je. Poskytuje náhled na zdroje, činnosti atd., které bude potřeba zahrnout ke správnému sestavení funkčního systému.

Základním předpokladem jazyka UML je skutečnost, že umožňuje modelování softwaru, stejně jako dalších systémů jako kolekce spolupracujících objektů [1]. Základní dělení struktury UML má následující součásti:

- Stavební bloky – základní prvky modelu, relace a diagramu
- Společné mechanismy – obecné způsoby dosahování specifických cílů
- Architektura – pohled na architekturu navrhovaných systému

Autoři [1] a [2] hovoří o metodikách unifikovaných procesů jako o nutnosti porozumění iterací. Iterace obsahuje všechny prvky softwarového projektu:

- Plánování
- Analýza a návrh
- Tvorba
- Integrace a testování
- Interní a externí uvedení

Dále každá iterace obsahuje pět základních pracovních postupů, určují, co a jakým způsobem má pracovat, pro dosáhnutí cíle: (pojmy jsou uvedené v časové posloupnosti)

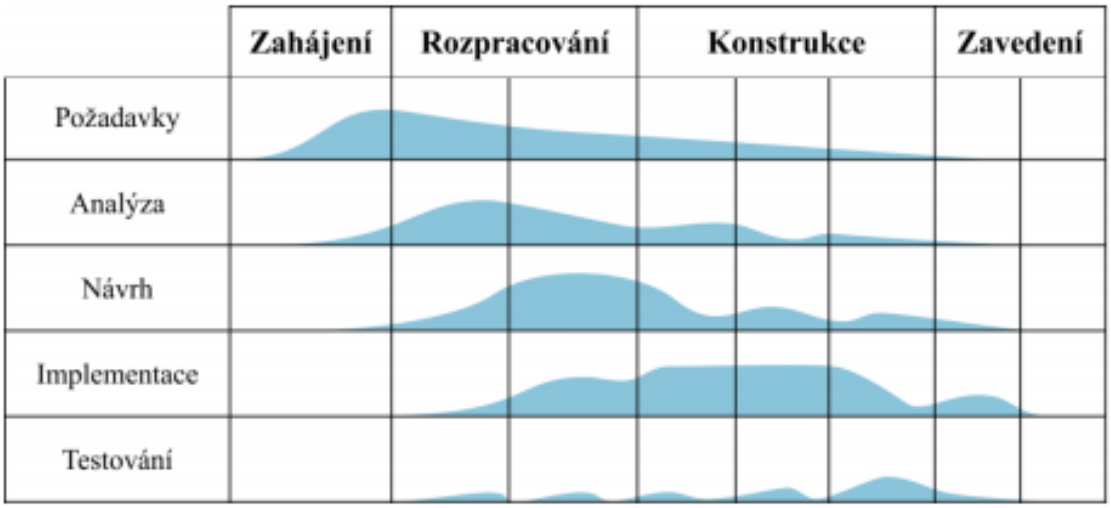
- Požadavky – kolekce požadavků na systém, co by měl dělat
- Analýza – odbornější úprava požadavků a jejich strukturování
- Návrh – realizace požadavku v architektuře systému
- Implementace – tvorba softwaru z návrhu

- Testování – ověření funkčnosti implementace podle požadavků.

Unifikovaný proces se dělí na fáze, každá fáze má určitý cíl:

- Zahájení – období plánování
- Rozpracování – období architektury
- Konstrukce – počátky provozuschopnosti
- Zavedení – nasazení produktu do uživatelského prostředí

Vztah iterace s fází a objem práce je interpretován na obrázku 3.1, kde řádky jsou iterace, sloupce jednotlivé fáze, které jsou časově rozdělené ln + 1. Výplň jednotlivých řádků představuje objem práce iterací za určité období.



Obrázek 3.1 – Fáze podle metodiky UP [1]

3.2 Analýza a návrh

Analýza a návrh zapadají do kolekce unifikovaného modelovacího jazyka (UML), o které byla řeč v úvodní kapitole.

Většina práce analýzy začíná probíhat ke konci fáze Zahájení, z důvodu pracování s poznatky z kolekce požadavků. Účelem analýzy je tvorba modelů, zachycující požadavky a charakteristickou formu požadovaného systému. Pracovní postupy analýzy a požadavků jsou často překryté, hlavně ve fázi Rozpracování. [1]

Návrh neboli pracovní postup je hlavní a zároveň poslední modelovací aktivitou fáze Rozpracování. Návrh spočívá v přesném určení analytických modelů, které budou použité ve fázi implementace. Pro tyto fáze autoři [1] doporučují vedení analýzy a návrhu pouze jedním týmem k dosažení nejpřesnějších analytických modelů.

3.2.1 Analýza současného stavu

Analýza současného stavu je založena na principu objektivním, nezaujatém pohledu na zkoumaný systém. Smyslem je získat kvalitní data, které mohou vést k vylepšení, odhalení chyb, úspor nebo inovací zkoumaného systému.

Zdrojem dat analýzy jsou konzultace s klientem, dotazníky, studie a analýzy interních dokumentů. Získaná data dále slouží pro upřesnění a sestavení analytických modelů. Velmi důležité je „mít cit“ pro efektivní třídění potřebných dat od nepotřebných, zavádějících nebo deformovaných. Chybná data by mohla zapříčinit totální kolaps celého projektu. [1] [2]

3.2.2 Model požadavku aplikace

Zpracovávání požadavků bývá prvním krokem projektování. Ještě před objektově orientovanou analýzou probíhá analýza rámcového přehledu o tom, jaký je smysl celého projektu a zajistit základní analýzy. Počáteční analýzy jsou psány terminologií srozumitelnou pro zadavatele. Analýza tímto zajistí základní typ požadavků a aktéry, které se v systému vyskytují. Požadavky jsou získávány z kontextu systému[1], který je ve snaze modelování.

Požadavek definuje jako specifikaci, co by mělo být implementováno. Rozdělují se na dva typy:

- Funkční požadavek
- Nefunkční požadavek

Funkční požadavky definují určitou funkci systému nebo jeho komponenty. Funkce je popsána jako množina vstupů, chování a výstupů. Funkční požadavky jsou například výpočty, zacházení s daty a jejich následné zpracování. Nefunkční požadavky jsou oporou funkčních požadavků. Jsou definovány jako omezující požadavky, za jakých okolností systém bude pracovat. Nefunkční požadavky ovlivňují architekturu celého systému. [1]

3.2.3 Model případů užití

Moderní forma z inženýrských požadavků, která se snaží zapojit systém s uživatelem a popsat jejich vztah v budoucím softwaru. Výsledkem je přehledný popis požadované funkcionality systému, které jsou očekávané od uživatelů tohoto systému.

[1] Kontext sytému = uživatelé systému, obchodní cíle, právní omezení, manažeři atd.

Dalším výstupem je pevná základna pro programátory, analytiky a návrháře, jelikož UP popisují jednotlivé funkční a nefunkční kroky systému a aktéra. Bohužel před tímto výstupem jsou kladeny velké nároky na správné nalezení hranic systému, vyhledání všech aktérů a identifikace případů užití i s alternativním scénářem. Postup je opakován několikanásobně, dokud nedojde k jeho ustálení.

Hranice systému je ve většině případech definováno, jako ohraničení systému tím, kdo jej využívá (aktér) a tím jaký má pro něj přínos (přínos pro aktéry). Označení aktér vyjadřuje roli uživatele nebo externího systému, který je na hranicích systému (který komunikuje se systémem). Aktéři jsou značení dle role, kterou v systému zastupují. Aktérem může být i čas, určité případy vyžadují čas pro splnění nefunkčních požadavků systému.

Modelem případů užití mohou být textové případy užití formou tabulky nebo grafické znázornění formou diagramu. [1]

Diagram případů užití graficky znázorňuje interakci (komunikaci) aktérů a případů užití, které utvářejí chování subjektu (hranice systému). Jinými slovy *diagram vypovídá o tom, co má systém umět, ale neříká, jak to bude dělat*. [3]

Subjekt lze chápat jako hranice, které oddělují konkrétní systém od zbytku světa. Správné stanovení subjektu má velký dopad na budoucí vývoj programu a jeho funkční požadavky. Aktér zastupuje roli, která komunikuje se systémem prostřednictvím jednotlivých případů užití. Aktér se dělí na dva typy: aktivní a pasivní. Aktivní aktér zahajuje komunikaci s případy užití, je zakreslován nalevo. Pasivní aktér je opak aktivního, případ užití zahajují komunikaci samy, zakresluje se napravo. Případům užití rozumíme jako specifický celek posloupností činností, včetně alternativních a chybových posloupností, které systém může vykonávat s aktéry. [3]

3.2.4 Analytické třídy

Diagram tříd dokumentuje statickou strukturu systému. Na počátku analýzy jim zachytíme konceptuální datový model systému (model pojmů). Tento počáteční pohled ilustruje entity a vztahy indikované během analýzy. [14]

Analytické třídy reprezentují velmi tenkou abstrakci problémové domény. Problémová doména je každá doména, na kterou soustředíme cíl našeho zájmu. Obvykle se jedná o zákazníka, produkty atd. Nejdůležitější povahou analytické třídy je skutečnost,

která by měla skutečný problém reálného světa. To by mělo vést k jednoznačné identifikaci daného problému a tento problém vyřešit. Hledat všechny problémy reálného světa, se kterými se systém bude potýkat záleží na šikovnosti analytika. Analytické třídy vedou k upřesnění do návrhových tříd, které při dokončení jsou napřímo implementovány do zdrojového kódu softwaru. [1]

3.3 Vývojová technologie

3.3.1 .Net Framework

.Net Framework je komplexní softwarová platforma určená pro vývoj pestré škály typů aplikací. Nabízí vývoj od základních okenních aplikací, přes dynamické webové služby až po mobilní aplikace, a to nejen pro Windows Phone / 10, ale i pro operační systém Android[2] a iOS[3]. .Net Framework je samo o sobě prostředí, které je potřebné pro běh aplikace, nabízí nejen spouštěcí rozhraní, ale poskytuje referenci na potřebné knihovny pro danou aplikaci. Platforma nabízí rozsáhlé možnosti využití programovacího jazyka. Avšak nejpopulárnější jsou tyto jazyky: C#, C++ a VB. Ve výsledku, při kompilaci zdrojových kódu je vše přeloženo do mezijazyku CLI[4], který je spustitelný na kterékoliv platformě, pokud má přístup k rozhraní .Net Framework. [4] [5]

.Net obsahuje specifické technologie, určené pro různé účely v projektu:

- ASP.NET – zaměřený vývoj webových aplikací
- Windows Communication Foundation – vývoj webových služeb a komunikační infrastruktury aplikací
- Windows Presentation Foundation – framework pro vytvoření graficky pestrého a animovaného prostředí
- Windows Workflow Foundation – vývoj aplikací spravující složité procesy
- Windows Sync Framework – usnadnění synchronizace dat
- LINQ – velmi efektivní syntaktická konstrukce usnadňující dotazování se nad různým typem dat

[2] https://www.android.com/
[3] https://developer.apple.com/
[4] Common Intermediate Language

3.3.2 Windows Presentation Foundation

Grafický subsystém WPF poskytuje nový vzhled, nové principy přizpůsobení ovládacích prvků, nové grafické funkce (včetně animace a 3D) a nové programovací rozhraní. WPF kromě toho zahrnuje zajímavý nový značkovací jazyk založený na XML, který se nazývá XAML (Extensible Application Markup Language). V některých případech lze v jazyku XAML vytvořit celé programy. Obecně se však aplikace skládají z kódu i značek. Pomocí jazyka XAML můžete definovat uživatelské rozhraní a grafické prvky aplikací (včetně grafiky a animace) a v procedurálním jazyku vytvořit kód k obsluze událostí uživatelského vstupu. [6]

.NET Framework využívá multiplatformní knihovny k implementaci Model-View-View Model (MVVM) a sdílení nastavení napříč platformami[5]. MVVM využívá technologii, která rozděluje aplikační část (model) od uživatelské (view). Komunikace probíhá prostřednictvím ViewModel. Databinding je pružný mechanizmus sloužící k synchronizaci dat a uživatelského prostředí. Tento mechanizmus je součástí XAML předdefinovaných vzorů uživatelského rozhraní. Další výhodou použití WPF je možnost stylizace GUI[6] pomocí nástroje Microsoft Blend. Tato aplikace podporuje grafické úpravy 2D nebo 3D widgetů s hardwarovou akcelerací pomocí DirectX[7] [4] [7]

Zajímavost z historie WPF:

Platforma WPF byla velmi kontroverzní a do roku 2015 se na internetu v diskusích objevovalo velké množství lidí považující ji za „mrtvou“. Jako „mrtvá“ platforma byla označovaná již v jejím ranném stádiu vývoje, převážně v beta testech, kdy neobsahovala zásadní funkce[8] oproti Windows Forms platformě. Dále neexistovala visuální tvorba, což vedlo k velmi obtížnému stavění aplikace. Neexistovala podpora pro Windows 98 a chyběla podpora vývoje knihoven třetích stran.

To se ovšem velmi rychle změnilo, Microsoft investoval velké úsilí do vývoje a Windows Forms byl posunutý na druhou kolej.

[5] Silverlight, Windows Phone a Windows 8.x a Windows 10 Store aplikace

[6] Graphical user interface

[7] sada knihoven poskytujících aplikační rozhraní pro umožnění přímého ovládání moderního hardwaru.

[8] Neobsahovala například TreeView, ListView, nebo OpenFileDialog

Dnes je WPF velmi oblíbenou platformou pro vývoj aplikací, nehledě na jeho moderní technologie, velkou základnu nadšenců vyvíjejících řadu knihoven usnadňující vývoj, obrovské množství možností grafického designu díky XAML a Blend. V dnešní době (2017) Microsoft vyvinul a několik let podporuje platformu Universal Windows Platform, která se zaměřuje na Windows Phone a Windows Store aplikace, avšak není tak populární jako WPF. [4] *zdroje*[9]

3.3.2.1 Model-View-View-Model

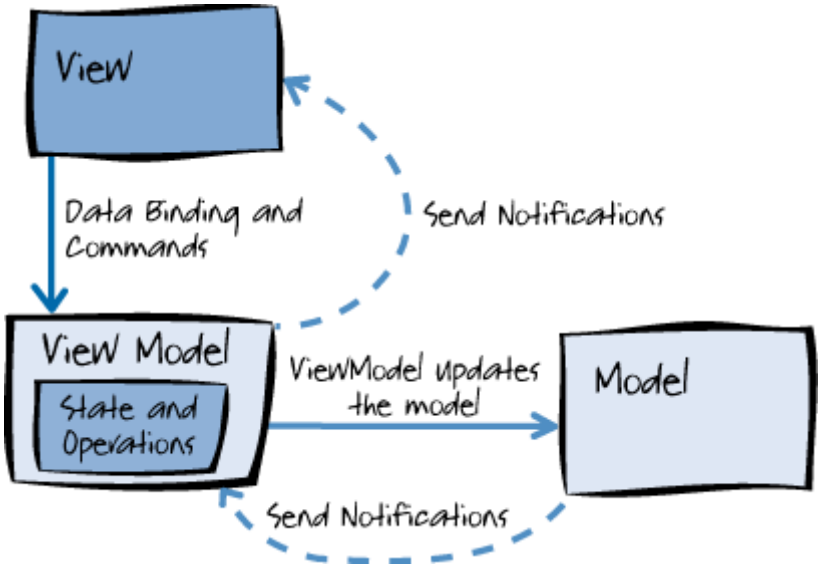
Jedná se o návrhový vzor pro WPF aplikace. Principem má být oddělení logiky aplikace od uživatelského prostředí. Výsledkem je čitelnější, přehlednější struktura, při dalším rozvoji kódu je mnohem snazší. Uživatelské rozhraní komunikuje s aplikační logikou prostřednictvím command a binding, na rozdíl od Windows forms, kde probíhala komunikace formou událostí.

Princip je založen na jednoduché myšlence, „prostředník“ třída ViewModel uchovává stav aplikace, uživatelské prostředí (View) si žádá o data, které se vypíšou uživateli. V okamžiku změny dat ve View je volán ViewModel, který si změny uloží. Princip je znázorněný na obrázku 3.2. [4] [6] [7] [8]

Z toho vyplívá následující:

- **Model** – popisuje data se kterými aplikace pracuje
- **ViewModel** – drží stav aplikace a spojuje Model s View. Ovládací prvky jsou řešeny pomocí bindingu a aby vše správně fungovalo musí být použitý *INotifyPropertyChanged*
- **View** – uživatelské rozhraní psané v jazyce XAML

[9] Zdrojů pro tento výzkum je velké množství, snadno hledatelné, například pod wpf is dead, wpf vs windows forms atd.



Obrázek 3.2 – Princip funkce MVVM[10]

3.3.2.2 Data Binding

Jedná se o technické propojení prvků a elementů na data. Data Binding je využívaný k automatizaci dosazování dat, například do tabulky pomocí Data Template, což je připravená šablona, kde jsou volána data z kolekce a díky správnému nastavení jsou dosazena na své místo. [4]

3.3.2.3 Command

ICommand je metodou ViewModel, kterou volá View při provedení akce uživatelem. Podnětem se spustí ověřování ve ViewModel, zda může provést změny v uživatelském rozhraní a zda náhodou svou akcí nepřeruší nějakou jinou. [4]

3.3.2.4 XAML a Microsoft Blend

Jazyk XAML pro vytváření grafického uživatelského rozhraní[11] (prezentační vrstvy). XAML vychází ze značkovacího jazyka XML, je velmi snadno čitelný i pro nezkušené uživatele a lze si jej upravit dle vlastní potřeby.

[10] Zdroj obrázku: https://msdn.microsoft.com/en-us/library/hh848246.aspx
[11] Cizojazyčná zkratka: GUI

Obrázek 3.3 – Ukázka aplikace Microsoft Blend[12]

Dokument XAML je složený z elementů, které mohou obsahovat další elementy. Takto strukturovaný dokument se nazývá stromová struktura. Element (tag) zapisujeme do lomených závorek. Párové elementy ukončujeme umístěním další značky za lomítko a nepárové ukončujeme lomítkem. Obsah u párových elementů můžeme vkládat mezi lomené závorky nebo přímo do atribut elementu (ukázka Zdrojový kód 1 – Ukázka XAML).

Jmenný prostor je další částí využívanou v XAML. Jedná se o označení a nasměrování na specifickou třídu v projektu, se kterou lze poté pracovat (ukázka Zdrojový kód 1 – Ukázka XAML). [7] [4] [6]

```
// Párový element s možností ukončení lomítkem díky vložení hodnoty do atribut
<Label Content="Nějaký text"/>
// Párový element ukončený elementem
<Label>Nějaký text</Label>

// Jmenný prostor
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
```

Zdrojový kód 1 – Ukázka XAML

Jak již bylo řečeno, XAML slouží pro tvorbu GUI a lze jej pomocí atribut modifikovat. Ke grafickým úpravám slouží nástroj Blend. Umožňuje snadno vytvářet

[12] Zdroj: https://en.wikipedia.org/wiki/Microsoft_Blend

grafické přechody, animace a další věci okolo GUI. Blend dokáže změny ukládat to specifických „slovníků“, které jsou poté volány jednotlivými elementy nebo mohou být nasazené pro celou stránku. Ukázka prostředí aplikace Blend je zobrazená na obrázku 3.3. [4] [6] [9]

3.3.3 Graph API a FTP

FTP je založena na frameworkové knihovně `System.Net.Socketsm, System.IO a System.Net`, která podporuje UDP[13] a TCP[14] komunikaci. Tyto knihovny jsou velmi důležité, nejen že obsahují metody a parametry potřebné k funkci, ale podporují protokol TCP, který při komunikaci mezi uzly udržuje neustálé spojení a zaručuje správné doručení všech dat. Další výhoda TCP spočívá v podpoře portů, se kterými aplikace pracuje, jedná se o tyto porty:

- 20 a 21 – jedná se o FTP[15] protokol, sloužící pro přenos dat mezi počítači (servery) pomocí počítačové sítě. Využívá dva porty, jelikož na portu 20 přenáší pouze data a na portu 21 naslouchá připojení klienta a zachycuje všechny jeho příkazy.
- 80 – jedná se o HTTP protokol určený pro výměnu hypertextových dokumentů.

Graph API je nízkoúrovňové HTTP založené na funkcích API. Zasílá klasické HTTP požadavky s přidáním potřebných autorizačních iniciálu a musí proběhnout autorizace pomocí autorizačního protokolu oAuth2.0 s Facebook servery, díky kterému aplikace dostane jedinečný klíč pro správné funkce celé knihovny. [10]

[13] User Datagram Protocol
[14] Transmission Control Protocol
[15] File Transfer Protocol

4 Praktická část

4.1 Analýza

V této kapitole si vysvětlíme hlavní záměr analýzy, získané poznatky a následně bude zhodnocená praktická část.

Analýza vychází z tvorby analytických modelů, které znázorňují chování požadavků zadavatele. Modely se specializují na to, co systém musí provést, avšak neřeší detaily způsobů, jakými to má provést. Touto otázkou se zabývá další kapitola (aktivita) Návrh.

Modelování analýzy je strategickou aktivitou, neboť během ní je snaha o modelování základního chování systému. [1]

4.1.1 Model požadavků a počáteční stav

Firma v současné době využívá dva systémy pro nahrávání obsahu. Jedná se o aplikaci Messenger[16] nebo pomocí webového prohlížeče přímo na stránkách Facebook[17]. Druhý systém se týká nahrávání obsahu prostřednictvím FTP za pomocí programu Total Commander[18]. Firma upřednostňuje první variantu zveřejňování nového obsahu a příspěvků před druhou. Tímto vznikají neaktuální „zapomenuté“ stránky, na což naráží smysl celé práce, usnadnění a automatizace práce.

Důvodem upřednostňování prvního systému před druhým je z pohledu firmy prostý. Práce firmy je focena na mobilní telefon s připojením k internetu a nainstalovanou aplikací Messenger. Tímto způsobem lze rychle zveřejnit nový obsah a mít přehled o aktivitě. V případě pořizování fotek pomocí digitálního fotoaparátu lze rovněž rychle a efektivně vystavit nový obsah. Tyto varianty, ale mají velký nedostatek. Jsou nahrávány bez třízení do kategorií a díky tomu vzniká obsáhlá a chaotická galerie, bez možností nalezení konkrétní události, například galerii svatebních vazeb.

Problém řazení dle kategorií je řešený na webových stránkách, ale nahrávání obsahu je již zdlouhavější, a proto je tento způsob již nějakou dobu neaktivní. Nahrávání probíhá vložením nového obsahu do patřičných kategorií (jsou celkem čtyři). Pro

[16] https://play.google.com/store/apps/details?id=com.facebook.orca&hl=cs
[17] https://www.facebook.com
[18] https://www.ghisler.com

dokončení musí uživatel vstoupit na specifickou stránku. Tím začne automatický proces přejmenování, vytvoření miniaturních obrázků (thumbnail), zanesení dat do databáze.

4.1.1.1 Funkční a nefunkční požadavky

Funkční požadavky jsou formulací sad podmínek a vlastností (služeb), které by měl systém provádět. Jedná se o požadavky určené zadavatelem aplikace ve znění, jemu srozumitelným. Nefunkčnímu systému lze rozumět jako souhrnu omezujících podmínek, která jsou vázána na návrhovou i logickou stánku systému.

Během konzultací se zadavatelem aplikace byla vznesena řada požadavků, které musejí být implementovány do systému. Požadavky byly dále upraveny pro dodržení metodik a snazší identifikaci zadání v dalších krocích projektu. Tímto krokem vznikly následující funkční i nefunkční požadavky:

4.1.1.1.1 Funkční požadavky

- Zobrazení seznamu fotek
- Zobrazení seznamu zpráv
- Grafický výstup statistik z obou prostředí
- Vyhledávání a řazení záznamů
- Kategorie dokumentů
- Vytváření, úprava a mazaní fotek a zpráv
- Uložení jako koncept
- Export dat

Zobrazení seznamu fotek

Zkrácený seznam se bude zobrazovat na úvodní obrazovce (Dashboard), bez možností filtrování a hledání, jedná se o rychlý přehled poslední aktivity. Úplný seznam bude k dispozici na obrazovce Media, obsahuje plně funkční filtrování, vyhledávání a řazení podle kritérií.

Zobrazení seznamu zpráv

Zkrácený seznam se bude zobrazovat na úvodní obrazovce (Dashboard), bez možností filtrování a hledání, jedná se o rychlý přehled poslední aktivity. Úplný seznam bude k dispozici na obrazovce Zprávy, obsahuje plně funkční filtrování, vyhledávání a řazení podle kritérií.

Grafický výstup statistik z obou prostředí

Statistiky budou zobrazeny podle patřičných grafů na úvodní obrazovce (Dashboard). Jedná se o nejpotřebnější údaje. Kompletní seznam bude k dispozici na obrazovce Statistika.

Vyhledávání a řazení záznamů

Widget[19], který bude umístněný u každé tabulky v aplikaci. Bude nastaven podle potřeby jednotlivé tabulky. Vykonává funkce vyhledávání obsahu podle kritérií (datum, id, název, kategorie), řazení a filtrování dle požadavku.

Kategorie dokumentů

Možnost programu přiřazovat novému nebo stávajícímu obsahu patřičnou kategorii, která se nyní vyskytuje na webovém serveru.

Vytváření, úprava a mazaní fotek a zpráv

Widget, který bude umístěný na tabulce, obsahující zprávy či obrázky a nebude se vyskytovat na úvodní stránce (Dashboard). Umožňuje uživateli upravovat, mazat a přidávat obsah. Upravovaní obsahu je vázané pouze pro konkrétní ID obsahu (uživatel bude moci upravovat v jednu chvíli pouze jeden záznam). Uživatel bude moci odstranit libovolný počet záznamů pomoci zaškrtnutí checkboxu. Limitace nebude vázána ani pro nahrávání nového obsahu, při nahrání lze označit veškerý nový obsah pod jednu kategorii nebo přiřadit různou kategorii jednotlivým položkám.

Uložení jako koncept

Uživatel disponuje možností uložení článku či dokumentů. Tyto data nebudou zveřejněny do doby potvrzení zveřejnění uložených dat.

Export dat

Uživatel si bude moci vyexportovat statistiku do formátu PDF. Export bude obsahovat zadaný rozsah období a kategorie, které si zadá sám uživatel.

[19] Widget je ovládací prvek, který je součástí aplikace. Pro webové aplikace známý jako Portlet

4.1.1.1.2 Nefunkční požadavky

- Ověřování přihlášení
- Ověření připojení
- Synchronizace dat
- Stabilita v off-line režimu

Ověřování přihlášení

Při prvním spuštění aplikace systém provede kontrolu přihlášení. Systém provádí kontrolu vždy, kdy dojde k přepnutí do jiného okna aplikace.

Ověření připojení

Systém provede kontrolu připojení k internetu a serverům při spuštění aplikace. Asynchronně ověřuje dostupnost po časovém intervale a při požadavku provedení akce s daty.

Synchronizace dat

Aplikace automaticky synchronizuje data při připojení k internetu v případě, že se jedná o první spuštění. Synchronizace je prováděna asynchronně v rozmezí desítek minut. Změna, přidání či smazání dat vede k okamžité synchronizaci se serverovými daty.

Stabilita v off-line režimu

Požadovaná vlastnost funkčnosti i v off-line režimu. Kdy publikování dat proběhne v okamžik připojení k internetu.

4.1.2 Analýza existujících řešení

Již jsou známé požadavky, které by aplikace měla splňovat. Jako první bude proveden průzkum již dostupných možností na trhu, splňující zadavatelské požadavky. Hlavním zdrojem průzkumu sloužila webová stránka Capterra[20]. Tato stránka poskytuje mohutnou databázi firemních programů s uživatelským ohodnocením.

V rámci hledání požadované aplikace se nepodařilo nalézt vyhovující produkt, který by splňoval veškeré požadavky, mezi které patří neplacení měsíčních poplatků za využívání programu. Nebo službu, která naplňuje alespoň většinu hlavních funkcí ze

[20] http://www.capterra.com/content-management-software/

zadání. Přece však byly vybrány tyto dvě aplikace: WordPress[21] a Contentful[22], vyhovující ve sjednocení nahrávání nového obsahu na obě prostředí v jeden okamžik.

4.1.2.1 WordPress

WordPress je open source software, který funguje jako plně přizpůsobitelný redakční systém a může být využitý téměř pro všechny účely. Jedná se o jeden z nejpopulárnějších a nejpoužívanějších redakčních systémů s velkou aktivní komunitou. V základní části poskytuje rychlou, přehlednou a dobře zabezpečenou aplikaci, která poskytuje možnost bezplatného web hostingu nebo jednoduché instalace na Váš webový server (tuto službu musí podporovat Váš web hostingový poskytovatel).

Jak již bylo dříve zmíněno WordPress nabízí mohutné množství doplňků, které jsou vytvářeny komunitou. Tyto dopňky jsou zdarma stažitelné a použitelné pro rozšíření funkčností služeb. V našem případě je nutné rozšířit služby doplňkem zvaný Facebook Auto Publish[23]. Díky čemuž je možné naplnit podmínku sjednoceného vystavování novinek a obrázků. [11]

4.1.2.2 Contentful

Contentful byl navržen pomocí nejmodernější praktiky COPE (Create Once, Publish Everywhere), strategie byla původně navržená společností NPR[24] ke spolehlivému řízení neustálého datového toku nového obsahu, který společnost vytváří na denní bázi a doručuje jej napříč několika publicitních míst. Program je koncipován na bázi podpory kolekcí API a lze jej využívat napříč všemi platformami. Součástí využití služeb, které jsou poskytovány zdarma (jsou omezené v rámci pravidel, vypsaných na stránkách[25]) je web hosting.

Veškerá data putující mezi aplikací (uživatelem) a serverem jsou ve formátu JSON, což znamená jednoduché použití na všech platformách a ve všech podporovaných programovacích jazycích. Společnost si klade velký důraz na zabezpečení komunikace, rychlosti a spolehlivosti své aplikace, kterou je možné využívat i na zařízeních s operačním

[21] https://wordpress.org/
[22] https://www.contentful.com/
[23] https://wordpress.org/plugins/facebook-auto-publish/
[24] http://www.npr.org/
[25] https://www.contentful.com/pricing/

systémem Android nebo iOS. Contentful poskytuje podporu pro provázání komunikace s WordPress API, tato možnost vede v rozšíření možností využití, přehlednou a moderní multiplatformní aplikací. [12] [13]

4.1.3 Model případů užití

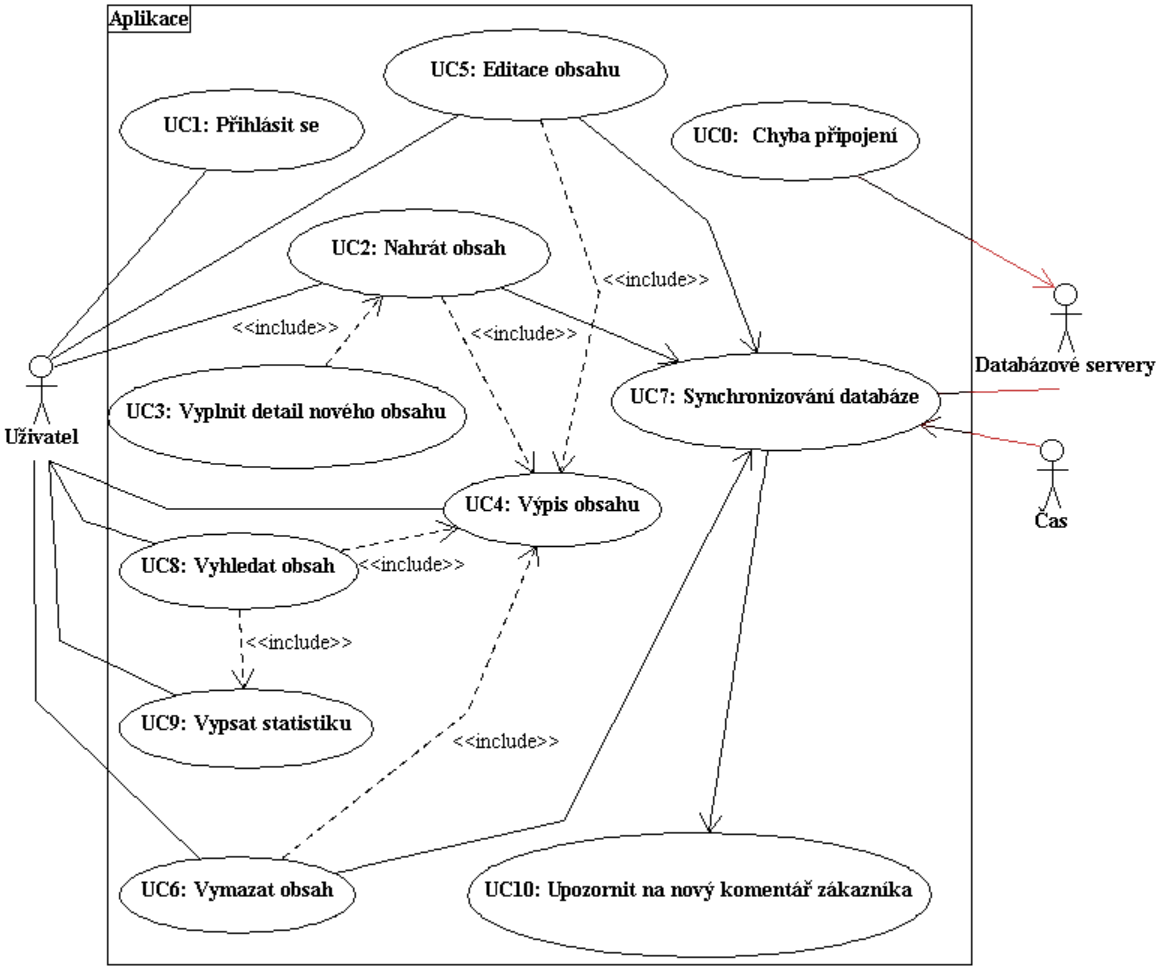
Modelování případů užití jednou z forem získávání a dokumentování požadavků a zároveň popisují interakci aktéra a systému. Také poskytují hlavní zdroj objektů a tříd. Tento zdroj patří k vstupním informacím k modelování tříd (Class diagram). Hlavní části modelování případů užití se skládají z nalezení hranic systému, vyhledání aktérů a nalezení případu užití.

4.1.3.1 Diagram případů užití

Diagram případů užití na obrázku 4.1 znázorňuje tři aktéry a jejich komunikaci s případy užití. Pro upřesnění pojmenování a jejich funkcí jsou vypsány v následujících bodech:

- **Uživatel** – uživatel systému, který jej využívá a operuje se všemi dostupnými funkcemi a má přístup ke všem datům.
- **Databázové servery** – externí systém, podněcující automatické procesy samotného systému. Probíhá zde oboustranná komunikace
- **Čas** – jedná se o nekontrolovatelného aktéra. Čas inicializuje funkce systému k vykonání aktivity.

Případy užití ohraničeny systémem Aplikace, aby vše bylo na první pohled srozumitelné. V další kapitole jsou všechny případy užití rozebrány a podrobně specifikované. Tento návrh slouží pouze pro rychlý náhled na strukturu případů.



Obrázek 4.1 - Diagram případů užití

4.1.3.2 Specifikace případu užití

Grafické zobrazení pomocí diagramu případů užití není dostatečně vhodné, aby znázorňoval jednotlivé kroky postupu k popsání případů užití. V tomto případě se využívá slovní formulace případů užití nebo také známé jako specifikace případů užití. Specifikace případů užití komplexně popisuje jednotlivé kroky postupu určité operace. V UML neexistuje žádný doporučený standard, ve většině případů se používá základní šablona znázorněná níže.

Tyto specifikace vycházejí z funkčních i nefunkčních požadavků, tudíž naplňují všechny zadané podmínky. První případ užití UC0: Chyba připojení popisuje postup chování systému při neúspěšném navázání komunikace s aktérem databázové servery. Podrobnější popis jsou k nalezení v tabulce 4.1.

Prvním krokem uživatele při spuštění aplikace je UC1: Přihlásit se. Přihlašovací formulář může probíhat na pozadí aplikace, jelikož načítá data z frameworku systému. Pro případ nenalezení dat musí uživatel zadat své heslo s možností „zapamatování si“ hesla. V tabulce 4.2 je k nalezení podrobnější popis alternativních scénářů, které systém provádí. Úspěšně přihlášený uživatel je přesměrován na úvodní obrazovku (Dashboard) a obdrží plnou kontrolu nad aplikací.

Dashboard je úvodní karta, poskytující uživateli základní přehled o událostech. Pro výpis těchto událostí systém provádí UC7: Synchronizování databáze. Podrobný popis je k nalezení v tabulce 4.10. Připravená interní databáze je použita pro okamžitý UC4: Výpis obsahu obrázků (tabulka 4.3) a zpráv do příslušných listů, dále je potřeba UC9: Vypsat statistiku (tabulka 4.10) do příslušných grafů. Celá operace je prováděna asynchronně, tudíž uživatel nemusí čekat na vypsání všech dat. Pro případ nenavázání komunikace nebo zdlouhavé synchronizaci systém nejprve načítá poslední stažená data pro rychlý výpis. Případ užití UC10: Upozornit na komentář zákazníka je forma notifikace na hlavní liště aplikace. Tyto notifikace bude možné vypnout.

V případě práce s obsahem, UC6: Vymazání obsahu, UC5: Editace obsahu, UC2: Nahrávání obsahu, slouží již zmíněné případy užití. Více informací jsou k nalezení v příslušných tabulkách případu užití. Případ užití UC2 je rozšířený o funkce dalším případem užití, a to UC3: Vyplnění detailu nového obsahu. Tento případ užití specifikuje vlastnosti nového obsahu a zatřiďuje jej do určité kategorie, se kterou systém pracuje a dále tyto informace předává databázovým serverům.

UC8: Vyhledávání obsahu pojednává o funkci vyhledávacího widgetu. Tento widget slouží pro vyhledání (do kterého patří i filtrování a třízení) dat, ať už obrázku, zprávy nebo statistik.

	UC0: Chyba připojení
Krátký popis	Systém zahlásí chybu připojení k síti, serveru či databázi a pokusí se několikrát o opětovné připojení
Vstupní podmínka	Systém nemůže navázat komunikaci s FB SDK, FTP serverem či MySQL databází
Výstupní podmínka	Systém provede patřičný úkon a zahlásí chybu uživateli; vrátí se do původního stavu před úkonem
Tok událostí	
Základní tok	
1.	Systém nedostane odpověď od vnějšího systému
2.	Systém zahlásí chybu uživateli a zpřístupní tlačítko „zastavit“
3.	Systém opakuje v intervalech pokus o znovunavázání komunikace
4.	Pokud uživatel přeruší pokus tlačítkem „zastavit“
4.1.	Systém se vrátí do stavu před odesláním
5.	Systém se po několika neúspěšných pokusech vrátí do stavu před odesláním

Tabulka 4.1 – UC0: Chyba připojení

	UC1: Přihlásit se
Krátký popis	Umožňuje uživateli přistoupit k funkcím aplikace
Vstupní podmínka	Uživatel spustí aplikaci nebo je přesměrován při odhlášení
Výstupní podmínka	Systém přesměruje uživatele na Úvodní stránku
Základní tok	
1.	Systém otevře přihlašovací formulář
2.	Uživatel vyplní jméno a heslo. Odešle formulář.
3.	Systém zvaliduje data zadané uživatelem
4.	Systém přesměruje uživatele na úvodní stránku a přidělí příslušná práva
Alternativní tok 1	
2.	Pokud uživatel zaškrtne pole „Zapamatovat uživatele“
2.1	Při opětovném spuštění systém provede automatické přihlášení
Alternativní tok 2	
3	Pokud systém zvaliduje data s chybou
3.1	Systém vypíše chybovou hlášku s upřesňující chybou pro uživatele
3.2	Uživatel opraví neplatný vstup a pokračuje na 2. bod Základního toku

Tabulka 4.2 – UC1: Přihlásit se

	UC4: Výpis obsahu
Krátký popis	Systém vypíše tabulku dat Zpráv / Soubory
Vstupní podmínka	Uživatel se nachází na stránce s výpisem obsahu
Výstupní podmínka	Systém vypíše obsah do tabulky
Tok událostí	
Základní tok	
1	Systém ověří dostupnost připojení
2	Pokud připojení nereaguje
2.1.	Systém pokračuje podle UC0
2.2	Systém načte uchovanou databázi
2.3	Systém vypíše data a zahlásí chybovou hlášku s upozorněním neproběhlé synchronizace dat se serverem
3	Systém stáhne data ze serveru
4	Systém načtené data porovná s místními daty a přepíše změny
5	Systém vypíše data do tabulky

Tabulka 4.3 – UC4: Výpis obsahu

	UC8: Vyhledávání obsahu
Krátký popis	Uživatel zadá kritéria, podle kterých požaduje seřazení či vyhledání obsahu
Vstupní podmínka	Uživatel se nachází na listu výpisu dat
Výstupní podmínka	Systém vypíše obsah podle kritérií
Tok událostí	
Základní tok	
1	Uživatel vyplní pole pro vyhledávání nebo vybere možnosti z dropdown menu u jednotlivé položky v listu
2	Systém provede třízení nebo vyhledávání podle kritérii
3	Systém vypíše hledaný obsah

Tabulka 4.4 – UC8: Vyhledávání obsahu

	UC2: Nahrávání obsahu
Krátký popis	Umožňuje uživateli nahrát soubory do systému. Nebo napsat novinku. Pokud jsou splněny všechny podmínky, systém automaticky zveřejní nahraný obsah na patřičná místa
Vstupní podmínka	Uživatel se nachází na kartě „přidat obsah“
Výstupní podmínka	Obsah byl úspěšně nahrán na server, systém přesměruje uživatele na výpis souborů
Tok událostí	
Základní tok	
1.	Pokud uživatel zvolí nahrání souboru
1.1.	Uživatel vybere soubory, které chce nahrát na server
1.2.	Uživatel vyplní obsah, Viz UC3 základní tok 1.
2.	Pokud uživatel zvolí vytvoření zprávy
2.1.	Uživatel vyplní obsah, Viz UC3 základní tok 2.
3.	Uživatel potvrdí nahrání dat tlačítkem „Nahrát“
4.	Systém úspěšně nahraje soubory na server a data do databáze. Uživatel je přesměrován na list výpisu souborů / zpráv
Alternativní tok 1	
2.	Pokud systém nenaváže spojení se serverem
2.1	Pokračuje UC0

Tabulka 4.5 – UC2: Nahrávání obsahu

	UC3: Vyplnění detailu nového obsahu
Krátký popis	Uživatel určí, jaký druh obsahu chce přidávat. Poté postupuje dle pokynu
Vstupní podmínka	Uživatel musí splnit podmínku vstupu u UC2
Výstupní podmínka	Správné odeslání obsahu
Tok událostí	
Základní tok	
1.	Pokud platí podmínka z UC2 / 1.
1.1.	Uživatel vybere soubory, které chce nahrát
1.2.	Uživatel vybere kategorii pro jednotlivé soubory
1.3.	Pokud kategorie neexistuje
1.3.1.	Založí kategorii tlačítkem „přidat kategorii“
1.4.	Pokud chce uživatel přidat zprávu k obrázkům
1.4.1.	Uživatel klikne na tlačítko „přidat zprávu“
1.4.2.	Uživatel postupuje podle Základní tok 2.
2.	Pokud platí podmínka z UC2 / 2.
2.1.	Uživatel vyplní pole „Zpráva“

Tabulka 4.6 – UC3: Vyplnění detailu nového obsahu

	UC5: Editace obsahu
Krátký popis	Uživatel upravuje již vystavený obsah nebo jej může zneplatnit
Vstupní podmínka	Uživatel musí kliknout na tlačítko „editovat“ v tabulce z UC4
Výstupní podmínka	Systém provede požadovanou úpravu v lokální i externí databázi
Tok událostí	
Základní tok	
1	Uživatel zvolí položku na výpisu dat (tabulka z UC4) tlačítkem „editovat“
2	Systém přesměruje uživatele na stránku editace dat
3	Uživatel provede úpravy (text / aktualizace souboru, změna kategorie, zneplatnit data)
4	Pokud uživatel provede aktualizaci souboru
4.1	Systém postupuje podle UC2, ale nahradí stávající soubor novým

Tabulka 4.7 – UC5: Editace obsahu

	UC6: Vymazání obsahu
Krátký popis	Systém vymaže uživatelem zvolená data
Vstupní podmínka	Uživatel musí kliknout na tlačítko „vymazat“ v tabulce z UC4
Výstupní podmínka	Systém vymaže data z lokální i serverové databáze
Tok událostí	
Základní tok	
1	Uživatel zvolí položku na výpisu dat (tabulka z UC4) tlačítkem „vymazat“
2	Systém zobrazí upozornění a tlačítko pro potvrzení rozhodnutí
3	Uživatel potvrdí, že opravdu chce vymazat zvolená data
4	Systém provede vymazání dat a synchronizuje databáze
Alternativní tok 1	
2.1	Uživatel stornuje požadavek
2.2	Systém přeruší operaci a vrací se do běžného stavu

Tabulka 4.8 – UC6: Vymazání obsahu

	UC10: Upozornit na komentář zákazníka
Krátký popis	Systém upozorní uživatele, pokud zákazník napíše zprávu
Vstupní podmínka	Systém zaznamená novou zprávu při synchronizaci UC7
Výstupní podmínka	Systém oznámí uživateli novou zprávu
Tok událostí	
Základní tok	
	Systém provádí synchronizaci UC7 a zjistí novou zprávu
	Systém zjistí počet nových zpráv
	Systém přičte hodnotu nových zpráv k aktuálnímu počtu u ukazatele nepřečtených zpráv

	Systém zobrazí uživateli aktuální počet nových nepřečtených zpráv

Tabulka 4.9 – UC10: Upozornit na komentář zákazníka

	UC7: Synchronizování databáze
Krátký popis	Systém provede synchronizaci lokální databáze s databází serveru. Jedná se o statistiky (Facebook, Google AdSense), obsahu z Facebooku a databázového serveru (zprávy, dokumenty atd.)
Vstupní podmínka	Uživatel musí být přihlášený a systém musí navázat komunikaci se serverem
Výstupní podmínka	Systém synchronizuje lokální a vzdálenou databázi a obsah
Tok událostí	
Základní tok	
1	Systém zahájí synchronizaci dat
2	Systém naváže komunikaci se serverem
3	Systém požádá o data a uloží je na lokální uložiště
4	Systém porovná původní verzi dat s aktuálními
5	Systém nenašel rozdíl, vše OK
6	Systém ukončuje komunikaci se serverem
Alternativní tok 1	
2	Pokud systém zjistí rozdíl na lokální databázi
2.1	Systém aktualizuje hlavní lokální databázi
2.2	Systém pokračuje na Základní tok, krok 6.
Alternativní tok 1	
3	Pokud systém zjistí rozdíl na serverové databázi
3.1	Systém aktualizuje serverovou databázi
3.2	Systém pokračuje na Základní tok, krok 6.

Tabulka 4.10 – UC7: Synchronizování databáze

	UC9: Vypsat statistiku
Krátký popis	Systém vypíše statistiku do patřičné podoby
Vstupní podmínka	Data z lokální databáze
Výstupní podmínka	Systém vypíše statistiku
Tok událostí	
Základní tok	
1	Systém načte data z lokální databáze
2	Systém použije data podle stanovených kritérii (např.: pouze dnešní přihlášení, aktivních uživatelů za rok, …)
3	Systém vytvoří patřičný graf
4	Systém zobrazí graf

Tabulka 4.11 – UC9: Vypsat statistiku

4.2 Návrh

Kapitola Návrh pojednává o konkrétním směru, ve kterém se bude aplikace vyvíjet.

4.2.1 Volba vývojové technologie

Specifikace požadavků zadavatele nekladla důraz na využití specifického programovacího jazyka. Pro vývoj aplikace byl zvolen vysokoúrovňový objektově orientovaný programovací jazyk C# fungující pod platformou Microsoft .NET Framework. V projektu se konkrétně pracuje s knihovnou Windows Presentation Foundation (WPF).

Pro komunikaci mezi aplikací a externími datovými servery bude zapotřebí využít internetový protokol HTTP[26] nebo skriptovací jazyk PHP[27].

4.2.2 Návrh aplikační architektury

Aplikace je navrhována jako Chytrý klient (SMART client). Jedná se o kombinaci výhod tlustého a tenkého klienta. Dokáže pracovat off-line, jelikož uchovává lokální data, která při navázání komunikace synchronizuje se serverem. Udržuje určitou logiku a pracuje se systémovými zdroji[28] a zároveň může žádat o vyřízení požadavku server.
V rámci aplikace neprobíhají výkonnostně náročné procesy, pouze jsou nutné velké datové přenosy směrem od klienta k serveru. Server posílá malé datové balíčky. V tomto případě byla zvolená dvouvrstvá architektura (klient/server).

V důsledku je nutná potřeba definice prezenční a datové vrstvy aplikace. Prezenční vrstva se stará o zobrazení místních dat, provedení aplikační logiky. Datová vrstva implementuje trvalé ukládání dat pomocí relačních databázových serverů (RDBMS) nebo jiných typů databází. Provázání komunikace mezi prezenční a datovou vrstvou slouží vrstva business logiky, která odpovídá za oboustranný přenos dat.

4.2.2.1 Prezenční vrstva

Vrstva uživatelského rozhraní zodpovědná za výkon aplikační logiky a obsluhu sběru dat nebo jejich zobrazení.

[26] Hypertext Transfer Protocol
[27] Hypertext Preprocessor
[28] Procesor, paměť, grafickou kartu atd.

Funkční požadavky formulují jasně daná pravidla pro prezenční vrstvu. Musí uživateli zobrazit rozdělená data v listě, kde bude moc být provedená úprava nebo přejití na detail jednotlivých záznamů. Uživatel by měl mít možnost zadat kritérium pro výstupní data.

Existuje zde potřeba navrhnout uživatelské prostředí, které bude schopné uživateli dovolit přidávání nového obsahu, volat vnitřní logiku a zpracovávat data.

4.2.2.1.1 Uživatelské prostředí

Uživatelské prostředí bylo navrženo s jednoduchým a čistým designem, pro snadnou orientaci. Barevné schéma bylo optimalizováno požadavkem zadavatele a webových stránek na tmavší odstíny. Pro celou aplikaci je využitý základní systémový font Segoe UI s možností rozšíření dle potřeb o některé jiné specifické fonty. Design je doplněný populárním balíčkem ikon Font Awesome[29].

Návrhová část uživatelského prostředí je tvořena formou Wireframe[30], jedná se o prvotní návrh aplikace a umístnění funkčních prvků. Návrh byl proveden aplikací ForeUI[31].

Hlavní panel aplikace

Část uživatelského prostředí, které je pro všechny obrazovky stejný. Tento panel je možné vidět na obrázku 4.2 v levé části aplikace a horní liště. Levý panel obsahuje komponenty jako je hlavní navigační menu, logo aplikace nebo uživatelské pole (Jméno uživatele, odhlášení se z aplikace, nastavení aplikace).

- Hlavní navigační menu tvoří ikona, text a pruh, který bude viditelný při aktivaci tlačítka (tento pruh není v drátovém modelu zobrazený)
- Funkce loga aplikace má dva účely, jednak slouží jako kosmetická část nebo jako přesměrování na úvodní obrazovku (Dashboard)
- Uživatelské pole slouží pro základní operace s aplikací, jako je její nastavení, zobrazení uživatelského jména nebo možnost odhlásit se z aplikace

V horní liště je možné vidět vyhledávací komponentu a notifikační ikonku sloužící pro ohlašování zpráv všeho druhu (chyby, nové komentáře od zákazníků, nemožnost

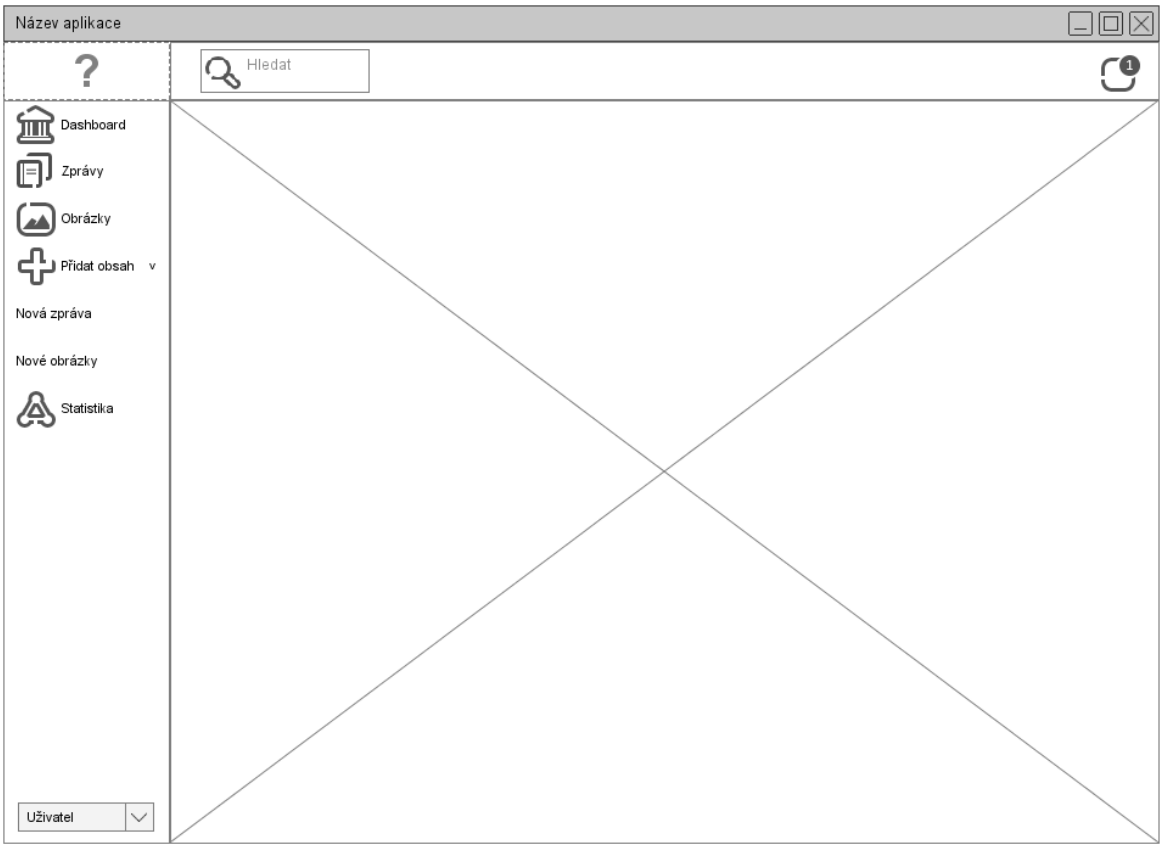
[29] Hlavní webové stránky: http://fontawesome.io/. Stránky knihovny pro WPF: https://github.com/charri/Font-Awesome-WPF

[30] Drátěný model

[31] Zdroj: http://www.foreui.com/

připojení k internetu apod.). Tato část nemusí být závazně pro všechny obrazovky stejná, horní lišta může být v některých částech doplněna o některé funkce.

Obsah z každé karty je vykreslovaný ve vyznačené (přeškrtnuté) části drátového modelu. S tím související zvýraznění jednotlivých tlačítek v hlavním menu podle toho, v jaké struktuře se uživatel momentálně nachází.



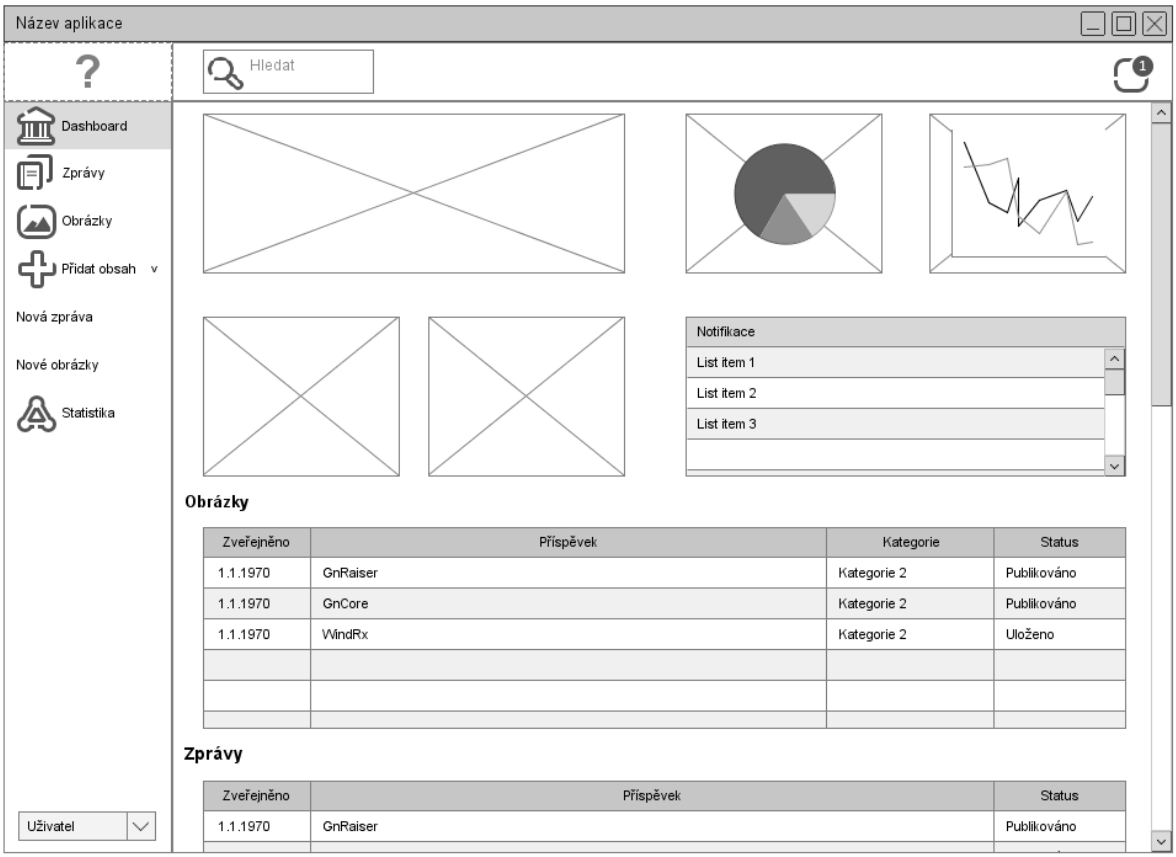
Obrázek 4.2 – Návrh GUI: Hlavní panel aplikace

Karta Dashboard

Dashboard je úvodní karta, obsahující rychlý náhled na důležitá data, tak aby uživatel měl okamžitý přehled o dění v systému. Jedná se o defaultní kartu při spuštění aplikace.

Horní část karty obsahuje komponenty sloužící pro výpis určitých statistik (například denní počet navštívení Facebookových nebo webový stránek). Může se jednat i o zobrazení několika dat v jednom grafu, které doplňuje legenda pro jednotlivá, barevně rozlišitelná data. Jsou zde použity tyto grafy: sloupcový, výsečový, prstencový nebo spojnicový. Grafy jsou rozdělené do jednotlivých komponent a podle typu grafu. Pro spojnicový graf je potřeba více místa, proto bude vypsán do komponenty 2x1 (2 bloky široký a 1 blok vysoký) nebo komponenty 2x2.

Další komponenty jsou určené pro tabulky nebo listy. První list vypisuje všechny události ve formátu: typ události a případný popisek, datum a čas. To vše do jednoho řádku, je to veškerý výpis z databáze událostí. Tyto události jsou také k vidění v horní části lišty (již bylo dříve zmíněno), ale informují pouze o „horké“ novince. Karta dále obsahuje dvě tabulky, jedna pro výpis databáze obrázků a další pro výpis zpráv. Zde se jedná pouze o informační výpis, bez možnosti filtování či třídění dat.



Obrázek 4.3 – Návrh GUI: úvodní karta

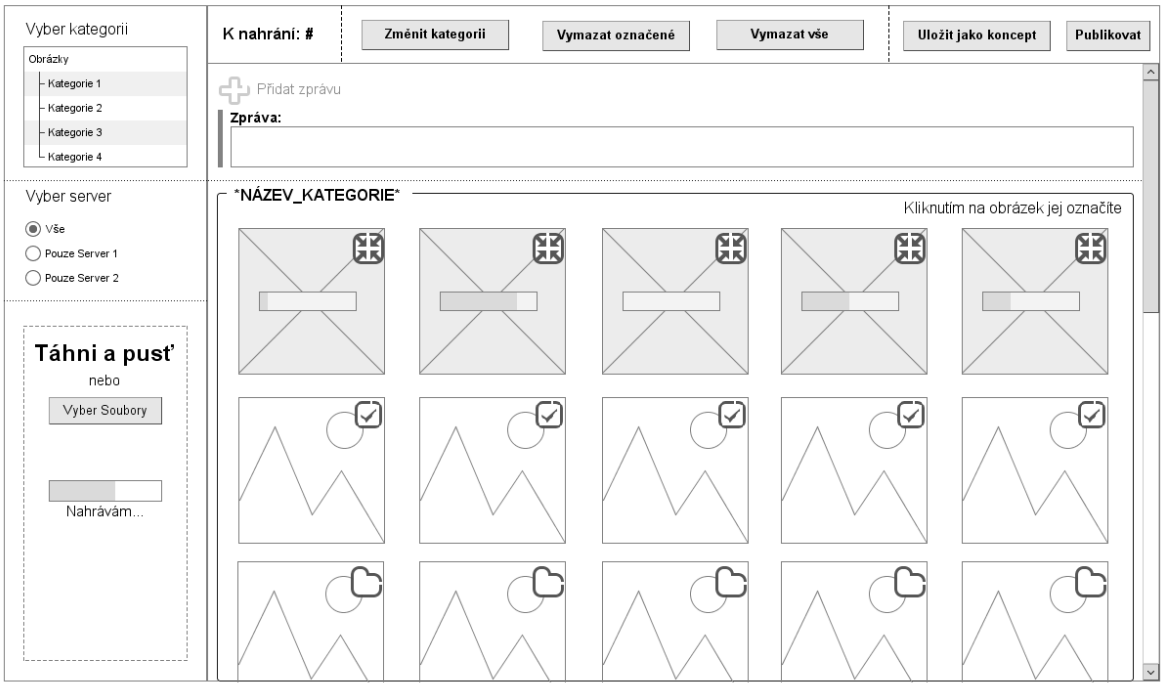
Nahrát nový obsah

Jedná se o podskupinu v Hlavním menu „Přidat obsah“, drátový model zobrazuje pouze obsah karty, nikoliv hlavní panel. Návrh zobrazuje již použité tlačítko pro přidání zprávy k nové skupině obrázků.

Uživatel musí postupovat podle určitých kroků, které budou jasně viditelné. Nejprve musí zvolit kategorii, kterou obrázky představují. Tato kategorie je vypsaná v treeview control[32] panelu pro případ budoucího rozšíření. V praxi to znamená, tato karta dokáže nahrávat a správně třídit i jiná data, než jsou pouze obrázky. Další volitelná

[32] Stromové zobrazení

možnost je publikace nového obsahu pouze na jednom prostředí, podle potřeby uživatele. Základní možnost je nastavena pro publikaci na obou dvou prostředí. Poté stačí vložit nový obsah, pro tuto část jsou dvě možnosti. První možnost je „táhni a pusť“ (drag and drop), to znamená, že uživatel označí data, která chce přemístit a tažením na panel spustí jejich načtení do aplikace. Druhá možnost spočívá v určení obsahu v klasickém systémovém okenním dialogu.



Obrázek 4.4 – Návrh GUI: karta „Nové obrázky“ (ukázka bez hlavního panelu)

V této fázi, po dokončení nahrávání je vše uložené na serverech, zatím nezveřejněno, ale uživatel má stále možnosti úprav. V případě zrušení, vypnutí aplikace se data vymažou (pokud nejsou uloženy jako koncept nebo publikovány). Nachází se zde možnost (výše zmíněné) přidání zprávy pro data pomocí tlačítka „Přidat zprávu“. Načtené obrázky jsou umístěné ve skupinovém bloku, který nese název vybrané kategorie. Kliknutím na obrázek je aktivován, to se projeví „odfajfkování“ pole v pravém horním rohu obrázku (lze pozorovat na návrhu v druhé řadě na obrázku 4.4). S označenými daty je nadále možnost pracovat:

- možnost změny kategorie – uživatel musí změnit kategorii jako v prvním kroku nahrávání a kliknout na tlačítko „změnit kategorii“. Tato akce vyvolá přesunutí obrázků do další skupiny s označením patřičné kategorie určené uživatelem

- vymazat data – kliknutím na tlačítko „vymazat označené“ uživatel provede vymazání dat z kategorie a již nadále nebudou součástí při další operaci s daty

Uživatel má možnost kompletního vymazání načtených obrázků tlačítkem „vymazat vše“. Poslední tlačítka reprezentují poslední krok. „Uložit jako koncept“ je funkce, která nahraný obsah zanese do databáze, ale neprovede jejich vystavení. Budou tedy sloužit jako koncept, který se dá kdykoliv publikovat pomocí karty Obrázky nebo Zprávy. Tlačítko „Publikovat“ odešle všechna data na servery a vystaví je pro veřejnost.

Výpis z listu – obrázky

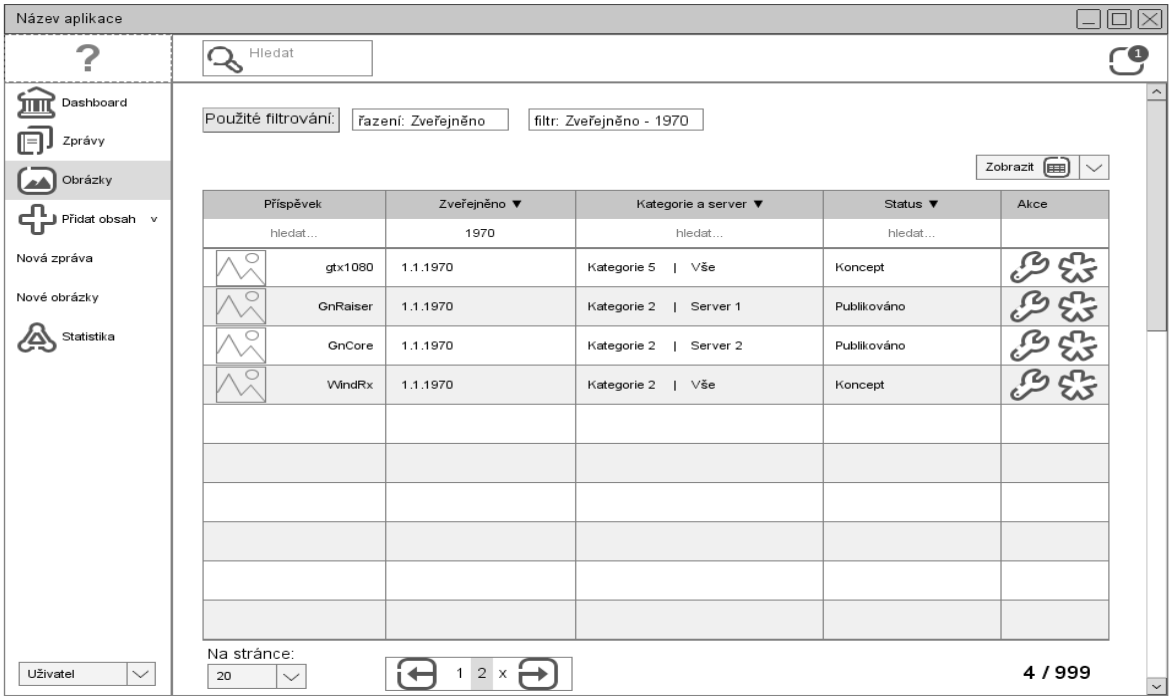
Návrh GUI pro zobrazení všech dat z databáze obrázků. Je zde přidaný widget pro filtrování a třídění dat. Použité filtry a třídění je zobrazeno ve skupině v horní části aplikace. Jednotlivé nastavení je možné vymazat pomocí kliknutí na tlačítko s patřičným filtrem či řazením.

Filtr dat funguje na rámci vyhledávání požadovaných kriterií, která se zadávají do prvního pole pod názvem sloupce. Návrh zobrazuje použití filtrování dat „Zveřejněno“ kritériem „1970“ a vzestupním řazením. Jednotlivé obrázky je možné maximalizovat kliknutím na thumbnail. Další možnosti jsou uvedené ve sloupci „Akce“: editace a odstranit. Editace umožní nahrát novou verzi obrázků, který bude následně aktualizován na všech prostředích. Podobná akce platí pro vymazání, kdy bude smazán ve všech prostředích.

Obrázek 4.5 – Návrh GUI: informace a ovládání obrázku

Uživatel má přehled o počtu všech dat v databázi v dolním pravém rohu, na které stránce se nachází anebo může určit počet položek načtených na stránce. Zobrazení tabulky je také modulovatelné, jsou zde dvě možnosti výpisu dat. První možnost je zobrazena na

návrhu (obrázek 4.6). Další možností je zobrazení dat vedle sebe, obdobně jako u nahrávání dat (obrázek 4.4), přičemž všechny informace jsou vypsány pod obrázkem. Návrh tohoto zobrazení lze vidět na obrázku 4.5.



Obrázek 4.6 – Návrh GUI: výpis všech obrázku s použitím filtru a řazení

Karta Statistika a ostatní karty

Statistika obsahuje podrobné grafické zobrazení všech statistických dat stažené z obou serverů. Podporuje rozšířené možnosti vyhledávání nebo třízení dat, než které jsou k vidění na úvodní kartě. Typ grafů zůstává zachovalý.

Ostatní nezmíněné karty jsou obdobou již zmíněných karet s obdobnými ovládacími prvky. Například karta „Nová zpráva“ je velmi jednoduchá a obsahuje pouze textové pole pro vložení textu zprávy, možnosti přiřadit k starším obrázkům nebo přesměrování na kartu „Nové obrázky“ s již aktivním textovým polem.

4.2.2.2 Datová vrstva

Vrstva jádrových systémů, které zajišťují cílové uložení informací a zajišťují další práci s nimi. Tato vrstva bývá často velice heterogenní s problematickou vzájemnou návazností některých částí (systémů). Pro snížení počtu integračních vazeb se využívá business vrstva, která tak umožňuje provázání interního systému pouze na jeden další uzel. [15]

Datová vrstva navrhované aplikace byla rozdělena do dvou částí. První část popisuje přístup k datům a jejich uložení. Druhá část pojednává o síťové komunikaci a postupu zavedení modulů do datové vrstvy. Obě části jsou vzájemně propojeny a jedna bez druhé nemohou fungovat. V této části je i zahrnutá „střední“ vrstva business logiky.

4.2.2.2.1 Ukládání dat

Aplikace funguje jako tlustý klient, co se týče ukládání dat. Jelikož je zapotřebí převádět veškeré obrázky na pole bajtů, z toho vyplívá větší zátěž na procesor pracovní stanice v případě většího množství dat. Tyto data se ukládají do textového souboru s formátem JSON. Tento soubor je pouze dočasný, při ukončení aplikace se vymaže a zároveň při spuštění vytvoří. Lokální databáze je řešená formou SQLite. JSON slouží pouze pro rychlou komunikaci a úpravu dat. V případě objemově náročné databáze je rychlejší využít JSON pouze pro jednorázové účely.

Aplikace pracuje s daty typu kategorie, „surovým“ obrázkem nebo převedeným do pole bajtů atd. Pro tento případ bylo nutné práci aplikace rozdělit na výše zmíněné řešení.

JSON udržuje pouze data, se kterými se pracovalo a poté přistupuje na komunikaci s externí databází. Tímto krokem se lze vyhnout náročnějšímu pohybu dat ze SQLite databáze. V druhém případě platí asynchronní porovnávání dat z lokální SQLite databází a externími databázemi.

V reálné situaci návrhu aplikace to znamená, že je potřeba vytvořit několik tříd, které se o tyto operace postarají. Třída `SyncData` se stará o udržení synchronizované databáze. Další například `IODatabaseHelper` zabezpečuje vytvoření a správu lokální databáze. Funkce této třídy je velmi důležitá, jelikož ostatní třídy jsou na ní závislé, bez této třídy by nebylo možné s databázi nikterak komunikovat.

4.2.2.2.2 Síťová komunikace

V teoretické části byla řeč na téma FTP a Graph API a jejich knihovny. Zmíněné knihovny jsou základem pro celou síťovou komunikaci aplikace s vnějším světem. Jelikož je zapotřebí navázat kontakt se dvěma servery, je potřeba tuto komunikaci rozdělit do dvou tříd. První třída `WebConnection` obsahuje metody udržující konektivitu a zajišťující přenos dat v rámci FTP protokolu. Jedná se o obrázky v původním formátu. Další metody zajišťují HTTP komunikaci, prostřednictvím této komunikace probíhá výměna dat

z databáze a přenáší se soubor ve formátu JSON. V této části probíhá menší úroveň zabezpečení přenosu dat. Aplikace musí vykonat přihlášení k FTP webovému serveru.

Další důležitou knihovnou je Facebook SDK podporující Framework a C#, konkrétně se jedná o knihovnu `facebook`[33]. Facebook SDK obsahuje všechny metody a parametry sloužící pro automatickou správu Facebooku anebo jeho ovládání. Primární funkce, která umožňuje celkovou komunikaci v rámci této knihovny je Graph API[34].

Třída `FacebookConnection` obstarává komunikaci a přenos dat mezi aplikací a Facebook serverem. Metody této třídy zajišťují odesílání obrázků a zpráv a žádají od serverů poskytnutí všech dat, které jsou zapracovány do databáze.

Aplikace při komunikaci využívá třídu `AsyncTask`, která zajišťuje asynchronní přenos dat. Při dokončení operace zavolá třídu `CompareAsyncTask`, jejíž úkol je porovnat nová data s lokálními. V případě odchylek zavolá podtřídu sloužící k opravě chyb na základě podmínek, podtřídy jsou provázané s třídou `IODatabaseHelper`.

[33] Zdroj: https://github.com/facebook-csharp-sdk/facebook-csharp-sdk
[34] Application Programming Interface

4.3 Implementace

Vývoj aplikace probíhá podle části analýzy a návrhu. Tato část se zaměřuje na specifické kroky a ukázku zdrojového kódu. Implementace je rozdělena na dvě podkapitoly: serverovou a aplikační. Serverová podkapitola se zaměřuje převážně na komunikaci a přenos dat mezi lokální a externí databází. Druhá aplikační část je zaměřena na aplikaci jako takovou. Obsahuje poznatky i ze serverové části, a hlavně hovoříme o jednotlivých komponentách a převedení návrhu uživatelského prostředí do skutečné podoby.

4.3.1 Serverová část

Serverová část je zaměřena na komunikaci aplikace s externími aktéry. Již dříve bylo zmíněno rozdělení a potřeba komunikovat se dvěma servery. V závěru této části se nachází shrnutí postřehů a problémů, které se neodhalily již při analýze a návrhu.

4.3.1.1 Facebook Graph API

Při komunikaci a práci s Facebook Graph API je nejprve potřeba mít založený účet s patřičným oprávněním a založit projekt, kde aplikace dostane jedinečný klíč pro

komunikaci se serverem.

Volání:

„https://graph.facebook.com/v2.8/1660511100892703/photos?fields=images&access_totok=token“

Odpověď:


```
"data": [
  {
    "images": [
      {
        "height": 960,
        "source": "https://scontent.xx.fbcdn.net/v/t1.0-
9/17155504_1845455889064889_5702328283927689185_n.jpg?oh=540627df7caefee704dd895b
57378686&oe=5964325D",
        "width": 720
      },
      {
        "height": 225,
        "source": "https://scontent.xx.fbcdn.net/v/t1.0-
0/p75x225/17155504_1845455889064889_5702328283927689185_n.jpg?oh=917d69ab092fbaf3
a14b3a2ea48c3c11&oe=59317A51",
        "width": 168
      }
    ],
    "id": "1845455889064889"
```


Zdrojový kód 2 – Ukázka Graph API požadavku a odpovědi

Při spuštění aplikace je potřebná autorizace uživatele prostřednictvím Facebooku. Zobrazí se interní webový prohlížeč, kde uživatel zadá jméno a heslo (Zdrojový kód 3). V případě, že vše proběhne v pořádku se okno automaticky zavře a aplikace nyní má potřebný klíč pro další práci (Access token). Pokud se vezme v úvahu, že aplikace provádí prvotní stahování všech fotek z alba „mobilní příspěvky“, které je zároveň kořenovým albem pro tuto aplikaci a kam bude provádět nahrávání, mazání nebo aktualizaci. Je potřeba zavolat HTTP příkazem požadavek ve tvaru vypsaného ve zdrojovém kódu 2. Server odpověděl pozitivně a zaslal výpis všech obrázků, jejich velikosti a URL pro

stažení.

```
        //Získané informace z přihlášení
        public string AppID { get; set; } // ID aplikace získané z přihlášení
        public string AccessToken { get; set; } // Access Token získaný z
přihlášení

        public FacebookLogin()
        {
                    InitializeComponent();
                    this.Loaded += (object sender, RoutedEventArgs e) =>
                    {
                        var destinationURL =
String.Format("https://www.facebook.com/dialog/oauth?client_id={0}&scope={1}&dis
play=popup&redirect_uri=http://www.facebook.com/connect/login_success.html&respo
nse_type=token",
                           AppID,
                           "email,user_birthday"
                        );
                        webBrowser.Navigate(destinationURL);
                    };
                }
        // ID aplikace získané z přihlášení
                private void webBrowser_Navigated(object sender,
System.Windows.Navigation.NavigationEventArgs e)
                {
                    //Pokud ziska acces token, pokračuj
                    var url = e.Uri.Fragment;
                    if (url.Contains("access_token") && url.Contains("#"))
                    {
                        url = (new
System.Text.RegularExpressions.Regex("#")).Replace(url, "?", 1);
                        AccessToken =
System.Web.HttpUtility.ParseQueryString(url).Get("access_token");
                        DialogResult = true;
                        this.Close();
                    }
        }
```

Zdrojový kód 3 – Webová autorizace - Facebook

HTPP příkazy aplikace ovládá veškerou komunikaci s Facebookem.

4.3.1.2 Webový server

Komunikace prostřednictvím FTP je velmi snadná, tato technologie je funguje již od nepaměti. Pomocí namespace `System.IO` a `System.Net` byly připojeny potřebné funkce. Třídou `FtpWebRequest` byla vytvořena lokální proměnná `reqFTP`, se kterou je nyní možné pracovat, určit vlastnosti nebo využít její metody. Ve zdrojovém kódu 4 je část ukázky nahrávání souborů. Nahrávání bylo určeno následovně: `reqFTP.Method = WebRequestMethods.Ftp.UploadFile`. Aplikace bude provádět nahrávání, dokud

contentLen nebude prázdný. Podmínku si neustále ověřuje ve While cyklu.

```
FileStream fileStream = fileInf.OpenRead();
try{
Stream stream = reqFTP.GetRequestStream();
       contentLen = fileStream.Read(buff, 0, buffLength);
// pokračuj dokud není konec
       while (contentLen != 0)
              {
                     // zapisuj soubory ze streamu a ukladej na server
                     stream.Write(buff, 0, contentLen);
                     contentLen = fileStream.Read(buff, 0, buffLength);
              }
              // Zavřít stream
stream.Close();
       fileStream.Close();
}
```

Zdrojový kód 4 – ukládání souborů pomocí FTP na webový server

Před nahráváním je důležité určit složku, do které má být obsah nahráván. Na serveru je řešeno nahrávání pomoci „zásobníku“, jedná se o složky, které jsou určené pouze k nahrávání nového obsahu. Složky jsou ještě rozdělené podle kategorií. Jakmile se dostane do této složky nějaký obrázek, formát souboru je ověřován serverem (jiný formát než .jpg a .png je smazán), server provede přemístění do hlavní složky, vytvoření malého obrázku (thumbnail), zápis do databáze a následné vystavení v galerii.

4.3.1.3 Souhrn implementace serverové části

Jeden z přídavků, který nebyl součástí návrhu je status ve spodní části aplikace. Pokud neproběhne správné přihlášení, bude zde vypsána chybová hláška oznamující off-line režim. Tento status je rozdělený podle serverů a vždy je vypsán, se kterým nelze navázat komunikaci.

Facebook API funguje dle pohledu autora dost obstojně, je potřeba provést úpravy komunikace, zjednodušit Model a Model-View strukturu kódu pro efektivnější akci. Ale ve výsledku vše funguje dle požadavků.

Webový server je o něco složitější a vše neproběhlo podle očekávání. Struktura aplikace na serveru je velice zastaralá a nedovoluje moc akcí. Řešení synchronizace je zde velmi obtížné a v budoucnu nejprve bude potřeba přepracovat webový server a následně aplikaci. Vše funguje dle zadání, ale aplikace „bruslí na tenkém ledě“ a mohou v budoucnu nastat případné komplikace.

4.3.2 Aplikační část

Část zaměřující se převážně na logickou strukturu aplikace, avšak rozpráví o grafickém prostředí přepracované z návrhu do reálné podoby a popisuje jednotlivé komponenty.

4.3.2.1 Navigační panel

Navigační panel zajišťuje kompletní navigační vlastnosti aplikace. Přesměrovává uživatele pomocí tlačítek v hlavním menu na různé obrazovky, poskytuje funkci „krok zpět nebo vpřed“. Proto tento panel lze nalézt ve všech obrazovkách. Proběhla změna oproti návrhu, kdy padla myšlenka zobrazovat logo aplikace, namísto se přidala již zmíněná funkce krokování. Zdrojový kód 5 – Zkrácený zápis hlavního menu v XAML představuje prvotní zápis hlavního menu bez jakéhokoliv stylování, které je řešeno statickou resourcedictionary. Existuje několik „slovníků“ s definovanými styly. Tyto slovníky jsou sjednocené a načítají se v kořenové struktuře App.xaml při spuštění programu. Za příkladem dále stojí logika, pokud `TextBlock` neobsahuje v proměnné cestu ke glyphu, bude vypsán podřazeně poslednímu zápisu s glyphem a zároveň nadřazené tlačítko ztratí schopnost být jakkoliv aktivované.

```
<ListView ItemsSource="{Binding MainMenuTemp}">
            <ListView.ItemTemplate>
                <DataTemplate>
                    <StackPanel>
                        <TextBlock Text="{Binding Glyph}"/>
                        <TextBlock Text="{Binding MenuText}"/>
                    </StackPanel>
                </DataTemplate>
        </ListView.ItemTemplate>
</ListView>
```

Zdrojový kód 5 – Zkrácený zápis hlavního menu v XAML

Zdrojový kód 6 je ukázkou principu funkce Model. Binding ve zdrojovém kódu 5 (View) požádá o data ModelView, který obsahuje předdefinované hodnoty, které jsou

pomocí metod v Modelu zařazeny do kolekce, která je následně vypsána uživateli.

```
public class MainMenuM : BindableBase
    {
        public MainMenuM() { }

        private string glyph;
        private string menuText;
        private bool isSecond;
        private DelegateCommand command;
        private Type navigationDestination;

        public string Glyph
        {
            get { return glyph; }
            set { SetProperty(ref glyph, value); }
        }
        public string MenuText
        {
            get { return menuText; }
            set { SetProperty(ref menuText, value); }
        }
        [DefaultValue(false)]
        public bool IsSecond
        {
            get { return isSecond; }
            set { SetProperty(ref isSecond, value); }
        }
        public ICommand Command
        {
            get { return command; }
            set { SetProperty(ref command, (DelegateCommand)value); }
        }
        public Type NavigationDestination
        {
            get { return navigationDestination; }
            set { SetProperty(ref navigationDestination, value); }
        }

        //Collection
        public static ObservableCollection<MainMenuM> mainMenuTemp = new
ObservableCollection<MainMenuM>();
        public ObservableCollection<MainMenuM> MainMenuTemp
        {
            get { return mainMenuTemp; }
        }
           }
```

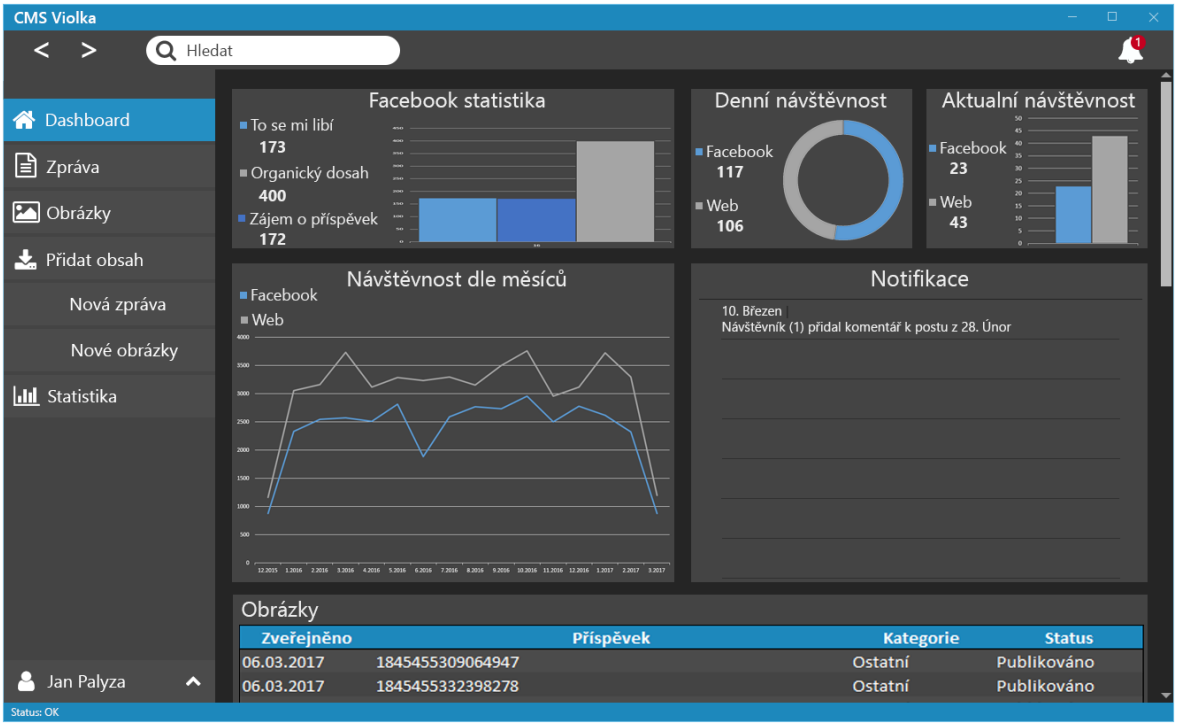
Zdrojový kód 6 – getry, setry a kolekce menu

4.3.2.2 Dashboard

Funkce Dashboard již byla několikrát vysvětlena. Pro připomínku, jedná se o úvodní stránku obrazovky, která zajišťuje rychlý přehled všech informací a obsahuje většinu funkčních widgetů, které lze v aplikaci nalézt.

Data statistik jsou dotahované z databáze `Statistika`, která obsahuje několik tabulek. Hlavní tabulkou je `traffic`, obsahuje všechna data návštěvnosti Facebooku a webu. Data jsou získávaná pomocí již zmiňované Facebook Graph API a z Google Analytics API, ten se chová obdobně jako Graph API. Notifikace jsou taktéž ukládané v této databázi, pouze mají vlastní tabulku.

Tabulka s obrázky a zprávami (zprávy jsou poslední, tudíž nejsou vidět na Obrázek 4.7 – Snímek obrazovky: Záložka Dashboard) dotahuje pouze data, bez možnosti úprav. Na jednotlivé záznamy lze přejít pomocí kliknutí na požadovanou položku. U obrázku nejsou zobrazené fyzické obrázky z důvodu šetřením místa.



Obrázek 4.7 – Snímek obrazovky: Záložka Dashboard

4.3.2.3 Zobrazit obrázky

Velmi důležitá část, jelikož dojde ke kompletnímu výpisu obsahu celé tabulky z databáze `Content`. Zdrojový kód 7 – XAML: načtení hlavního menu reprezentuje XAML zápis šablony s komponenty, které pomocí Bindingu vypíší všechna data z kolekce.

O naplnění kolekce se stará cyklus While, který do ní přidává jednotlivé hodnoty z geterů. Getry a setry jsou postupně plněné hodnotami z databáze. Ukázka zápisu Zdrojový kód 8 – Inicializace dat, vše je řešeno událostí `INotifyPropertyChanged`.

```
       <DataGrid x:Name="ImagesDataGrid"
                          Grid.Row="5"
                          Grid.Column="1"
                          Margin="5"
                          RowHeaderWidth="0"
                          AutoGenerateColumns="False"
                          ItemsSource="{Binding Path=ImagesAlbum}"
                          Style="{DynamicResource AzureDataGrid}"
SelectionChanged="AzureDataGrid_SelectionChanged">
       <DataGrid.Columns>
              <DataGridTemplateColumn Header="Příspěvek" Width="SizeToCells"
       IsReadOnly="True">
                             <DataGridTemplateColumn.CellTemplate>
                                    <DataTemplate>
                                           <Image Source="{Binding Path}" />
                                           <TextBlock Source="{Binding TextC}" />
                                    </DataTemplate>
                             </DataGridTemplateColumn.CellTemplate>
                </DataGridTemplateColumn>
                          <DataGridTextColumn Binding="{Binding Public}"
Header="Zveřejněno" />
                          <DataGridTextColumn Header="Kategorie a servery">
<DataGridTemplateColumn.CellTemplate>
                             <DataTemplate>
                                    <TextBlock Source="{Binding Database.Category}"
                                    <TextBlock Source="{Binding Database.Servers}"
/>
                             </DataTemplate>
                          </DataGridTemplateColumn.CellTemplate>
                </DataGridTemplateColumn>
                           <DataGridTextColumn Binding="{Binding Database.Status}"
Header="Status" />
                           <DataGridTextColumn Header="Akce"
<DataGridTemplateColumn.CellTemplate>
                                    <DataTemplate>
                                           <Button Content="Edit" Command="{Binding
EditCommand}" CommandParameter="{Binding ViewModelItem}" >
                <Button Content="Delete" Command="{Binding DeleteCommand}"
CommandParameter="{Binding ViewModelItem}" >
       </DataTemplate>
                             </DataGridTemplateColumn.CellTemplate>
                  </DataGridTemplateColumn>
              </DataGrid.Columns>
       </DataGrid>
```

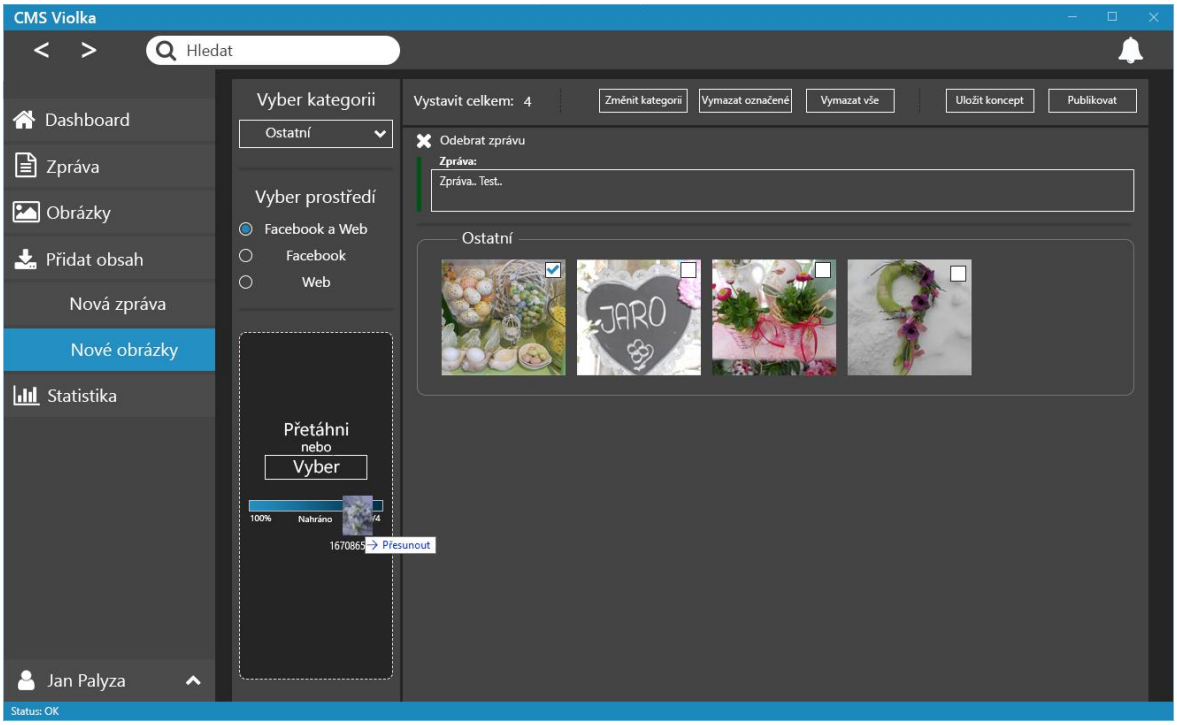
Zdrojový kód 7 – XAML: načtení hlavního menu

```
        private string _status;
        public string Status
                {
                    get { return _status; }
                    set
                    {
                        if (value == _status) return;
                        _status = value;
                        OnPropertyChanged();
                    }
                }
```

Zdrojový kód 8 – Inicializace dat

4.3.2.4 Nahrát obrázky

Obrazovka Nahrát obrázky pracuje na bázi automatického nahrání obrázku na servery se statusem nezveřejňovat. V případě vypnutí programu budou automaticky smazány. Z návrhu Obrázek 4.4 – Návrh GUI: karta „Nové obrázky“ (ukázka bez hlavního panelu) lze vypozorovat možnost zrušení nahrávání obrázků. Obrázek 4.8 – Snímek obrazovky: Nahrát obrázek je přepracování návrhu do reálné podoby. Pro zkoušku byly nahrány čtyři obrázky (zatím pouze nahrány jsou neuloženy a nepublikovány) a byla přidána zpráva ve tvaru „zpráva… test…“.



Obrázek 4.8 – Snímek obrazovky: Nahrát obrázek

Dále je možné vidět akci přidávání dalšího obrázku pomocí widgetu drag and drop. Kategorie jsou vybírány z dropdown menu, které jsou napevno definované v kódu a prostředí voleno z radio button.

4.3.2.5 Synchronizace dat

Synchronizace dat je nedílnou součástí aplikace, jelikož je neustálá potřeba udržovat aktuální data. Za tímto účelem byla potřeba vyřešit automatizaci synchronizace. Zdrojový kód 9 – asynchronní časovač je ukázka kódu, zajišťující tuto funkci. Jedná se o asynchronní operaci, která zaručuje, že se provede každý stanovený interval, což je 15 minut. Funkce přeruší v důsledku nedostupnosti připojení nebo nemožnosti navázání komunikace se serverem. Tímto jsou dvě nezávislé kontroly, první kontroluje připojení a druhá dokončení minulé synchronizace. V případě nesplnění těchto podmínek časovač nepracuje a musí být zavolán po dokončení. Třída `SyncData` obsahuje asynchronní operace, které obstarávají zpracování stažených dat ze serverů.

```
       class TimerAsync
           {
               private SyncData _syncData;
               public async Task AsyncTimerStart(CancellationToken Canceltoken =
default(CancellationToken))
               {
                   while (!Canceltoken.IsCancellationRequested)
                   {
                       _syncData.StartSyncData();
                       try
                       {
                           await Task.Delay(TimeSpan.FromMinutes(15),
Canceltoken);
                       }
                       catch (TaskCanceledException)
                       {
                           break;
                       }
                   }
               }
```

Zdrojový kód 9 – asynchronní časovač

4.3.2.6 Widget prstencový graf

Grafy v celém projektu byly náročnou překážkou. Prstencový graf je v hlavním jádru rotující progress bar. Pro vykreslování grafu byla třeba využít třídu `path`, která dokáže kreslit křivky a obrazce. `PathGeometry`, `Figures` atd. jsou shluk objektů, které při naplnění dat vytvoří finální podobu grafu (Zdrojový kód 10 – XAML Prstencového

grafu).

```
        <Path x:Name="pathRoot" Stroke="{Binding SegmentColor,
ElementName=DonutGraph}"
                StrokeThickness="{Binding StrokeThickness,
ElementName=DonutGraph}"
                HorizontalAlignment="Left" VerticalAlignment="Top">
        <Path.Data>
                <PathGeometry>
                        <PathGeometry.Figures>
                                <PathFigureCollection>
                                    <PathFigure x:Name="pathFigure">
                                        <PathFigure.Segments>
                                            <PathSegmentCollection>
                                                <ArcSegment
x:Name="arcSegment" SweepDirection="Clockwise" />
                                            </PathSegmentCollection>
                                        </PathFigure.Segments>
                                    </PathFigure>
                                </PathFigureCollection>
                        </PathGeometry.Figures>
                </PathGeometry>
```

Zdrojový kód 10 – XAML Prstencového grafu

Renderování grafu se spouští automaticky při zapnutí aplikace, jelikož se widget nachází na úvodní obrazovce. Vlastnost `Angel` zajišťuje, že se graf vždy vykreslí v kruhu a nevytvoří žádný jiný obrazec. Třída `RenderArc` provádí vykreslování grafu (Zdrojový kód 11 – Renderování prstencového grafu). Začíná v 0 bodě, který se nachází uprostřed nahoře grafu a pokračuje do bodu, který je následně vypočítán. Dále musí být nastavená hodnota pro tloušťku grafu, aby byl dokonale vykreslen. Stejným principem lze generovat výsečový graf, který bude obsahovat větší tloušťku linky. Prstencový graf obsahuje pouze dva údaje, protože bylo snazší graf vytvořit. Stačí pouze vypočítat procenta podílu a vykreslit pouze jeden údaj, druhý bude automaticky dokreslen bez dalších operací. Poté

stačí pouze dosadit přesné číselné hodnoty a názvy hodnot pomocí ViewModelu.

```
        public GraphModel()
                    {
                        InitializeComponent();
                        Angle = (Percentage * 360) / 100;
                        RenderArc();
                    }
        public void RenderArc()
                    {
                        Point startPoint = new Point(Radius, 0);
                        Point endPoint = ComputeCartesianCoordinate(Angle,
Radius);
                        endPoint.X += Radius;
                        endPoint.Y += Radius;

                        pathRoot.Width = Radius * 2 + StrokeThickness;
                        pathRoot.Height = Radius * 2 + StrokeThickness;
                        pathRoot.Margin = new Thickness(StrokeThickness,
StrokeThickness, 0, 0);

                        bool largeArc = Angle > 180.0;
                        Size outerArcSize = new Size(Radius, Radius);
                        pathFigure.StartPoint = startPoint;

                        if (startPoint.X == Math.Round(endPoint.X) && startPoint.Y
== Math.Round(endPoint.Y))
                            endPoint.X -= 0.01;

                        arcSegment.Point = endPoint;
                        arcSegment.Size = outerArcSize;
                        arcSegment.IsLargeArc = largeArc;
                    }
```

Zdrojový kód 11 – Renderování prstencového grafu

4.3.2.7 Widget filtrování dat

Widget je použit na několika místech, například na již zmíněném Zobrazit obrázky. Funkce je velmi prostá, ve View jsou vytvořené dvě možnosti. První je vyhledávání pomocí klíčových slov, které uživatel zadává do vyhledávacího formuláře. Metoda `FilterView` obdrží požadovaný text a spustí proces hledání v kolekci dat ve sloupci. Po nalezení všech dat provede aktualizaci tabulky s požadovaným obsahem.

Dalším způsobem, jak najít požadovaná data je pomocí comboboxu. Bindingem jsou načteny všechna data ze sloupce a jsou přiřazena k hlavičce sloupce. O tuto akci se stará metoda `IColumnHeaderCollector`.

`ColumnHeaderFilter` je zodpovědný za výpis požadovaných dat, která se shodují s požadovanou hodnotou. (Zdrojový kód 12 – Načtení dat ze sloupce do

comboboxu).

```
       public ColumnHeaderFileter (IMainFilterEvents mainFilterEvents, String
headerName, String propertyName, List<String> filterTexts)
               {
                   _mainVMEvents = mainVMEvents;

                   HeaderText = headerName;

                   PropertyName = propertyName;

                   Filters = new ObservableCollection<IFilterLocator> ();

                   _isHeaderPopupOpen = false;

                   foreach (var filterText in filterTexts)
                   {
                       var filter = new FilterViewModel (mainFilterEvents,
filterText);

                       Filters.Add (filter);
                   }
```

Zdrojový kód 12 – Načtení dat ze sloupce do comboboxu

5 Diskuse

Během analýzy systému a vývoje aplikace autor objevil větší i menší zádrhele, které komplikovaly samotný vývoj. V časovém intervalu by nebylo možné vše vyřešit a platilo zde riziko kompletního selhání. Proto byl stanovený budoucí plán rozvoje celé architektury webového serveru a následně i aplikace a její další rozšíření na platformu Android. Tím velmi utrpělo budování této aplikace, platilo zde vysoké nebezpečí úniku nešifrovaných dat ze strany serveru.

V obzoru několika měsíců či let existuje plán na spuštění rozvoje serverové architektury a přechod od poskytovatele webového serveru z důvodů špatné podpory API rozhraní. Cílem je mít co nejvíce totožnou architekturu s Facebook Glyph API, pro ulehčení a zmenšení počtu potřebného kódu a nutnost spravovat dvě rozdílné prostředí v jeden okamžik. Další hlavní myšlenka poukazuje na potřebu velkého datového přenosu aneb duplicitního přenosu na dvě rozdílná prostředí. Cílem je vytvořit „prostředníka“, přes kterého budou veškerá data putovat, sám vyhodnotí správnost obsahu a dále zašle ověřená, správná data bez potřeby provádět kalkulace na straně klienta. Tímto se podaří vyhnout datové náročnosti přenosu.

Stanovený cíl modernizace přináší s sebou další novinky. První novinkou by byla úprava dosavadního kódu a vytvoření takzvané sdílené knihovny. Tato knihovna obsahuje všechny komponenty struktury MVVM, styly a další. Vedlo by to k maximálně přehledné a čisté struktuře, která by se dala využít i na jiných platformách. To vede k poslednímu plánu, a to vytvořit velmi jednoduchou aplikaci na platformu Android.

Jednalo by se o „osekanou“ aplikaci psanou v C# a používající sdílenou knihovnu pro komunikaci se serverem a základní funkcionality. Pod základní funkcionalitou si lze představit nahrávání obrázku dle kategorií a prostředí nebo zobrazování databáze obrázků. Důvod, který k tomu navádí je, co nejrychlejší správa obsahu.

6 Závěr

Práce se zabývala problematikou tvorby desktopové aplikace vytvářené ve WPF pod .Net Framework. První teoretická část vysvětluje složitosti a zákonitosti tvorby objektově orientovaného návrhu jazykem UML a využití metodik Unified Process. Ve druhé teoretické části je věnován prostor pro objasnění zásad vývoje aplikací ve WPF, podrobněji popisuje funkce, které jsou využívány v průběhu řešení problematiky.

Praktická část na základě poznatků popisuje průběh od sběru poznatků, přes jejich analýzu až k vyhotovení návrhu a jeho následné implementace. Analýza probíhala na základě osobní komunikace se zadavatelem a vyhotovení seznamu požadavků, které následně byly rozpracovány do modelů případů užití. Ještě před modelováním ovšem proběhla analýza existujících řešení, zda by existovala možnost, jak vyřešit problém rychle a efektivně. Návrh byl zaměřený na aplikační logiku a grafické prostředí. Bylo velmi důležité pochopit, jakým způsobem bude aplikace stavěna z pohledu klient / server, jelikož stav architektury serveru nebyl příznivý a vývoj musel probíhat opatrně. Byly vytvořené návrhy prostředí, které vyhovovalo obou stranám a mohlo se začít implementovat. Implementace návrhu probíhala v některých krocích obtížně z nedostatků zkušeností autora a větší náročnosti kódu.

Stanovené cíle byly naplněny do posledního, jak již bylo zmíněno v implementaci v praktické části. Ovšem, ne všechny funkční či nefunkční požadavky jsou zahrnuty v implementaci. Ale například export statistických dat do formátu .PDF je triviální záležitost, kterou lze snadno a rychle docílit.

Pro autora měla práce velký význam, hlavně z hlediska praxe a porozumění komplexnosti, skrývající se za „obyčejnou“ aplikací. Díky splnění všech požadovaných cílů má velký vliv na spokojenosti zadavatele. Aplikace se chystá do zátěžového testu a v případě úspěchu rovnou do ostrého provozu. Jak již bylo řečeno v Diskusi, zadavatel má zájem o další spolupráci a rozvoj celého systému i pro další platformy.

7 Bibliografie

[1] Jim Arlow a Ila Neustadt, UML2 a unifikovaný proces vývoje aplikací, Brno: Computer Press, 2011.

[2] D. Chlapek, T. Bruckner, J. Voříšek a A. Buchalcevová, Tvorba informačních systémů, Grada Publishing a.s., 2012.

[3] D. Čápka, „ITnetwork.cz," [Online]. Available: http://www.itnetwork.cz/navrhove-vzory/uml/uml-use-case-diagram.

[4] I. Vrána a K. Richta, Zásady a postupy zavádění podnikových informačních systémů, Grada, 2004.

[5] A. Nathan, Windows Presentation Foundation unleashed, Indianapolis: Sams Publishing, 2007.

[6] T. Herceg, „úvod do .net frameworku," DotNetPortal.cz, 3 Duben 2009. [Online]. Available: http://www.dotnetportal.cz/clanek/125/Uvod-do-NET-Frameworku.

[7] C. Petzold, Mistrovství ve Windows Presentation Foundation, Brno: COMPUTER PRESS, 2008.

[8] Developer Network, „Používání knihovny přenosných tříd spolu s modelem MVVM," Microsoft, [Online]. Available: https://msdn.microsoft.com/en-gb/library/hh563947(v=vs.110).aspx.

[9] „The MVVM Pattern," Microsoft Developer Network, 10 2 2012. [Online]. Available: https://msdn.microsoft.com/en-us/library/hh848246.aspx.

[10] „Vytvoření uživatelského rozhraní pomocí nástroje Blend for Visual Studio," Microsoft Developer Network, [Online]. Available: https://msdn.microsoft.com/en-gb/library/jj171012.aspx.

[11] „Graph API Overview," Facebook for developers, [Online]. Available: https://developers.facebook.com/docs/graph-api/overview/.

[12] „WordPress," [Online]. Available: https://wordpress.org/.

[13] „Contentful," [Online]. Available: https://www.contentful.com/.

[14] Ed Holzwarth, „Little Green Software,“ 10 Prosinec 2015. [Online]. Available: https://littlegreensoftware.com/blog/strategy/why-contentful-is-our-content-management-system-of-choice.

[15] J. Tonar, „Jak uplatnit principy třívrstvé architektury v rámci web integračního projektu,“ Webová integrace, 15 10 2013. [Online]. Available: http://www.web-integration.info/cs/blog/jak-uplatnit-principy-trivrstve-architektury-v-ramci-web-integracniho-projektu/.

8 Seznamy

8.1 Seznam odborných zkratek

WPF – Windows Presentation Foundation

UML – Unified Modeling Language

CMS – Content management system

COPE – Create Once, Publish Everywhere

MVVM – Model-View-View Model

RDBMS – Relational DataBase Management System

UDP – User Datagram Protocol

TCP – Transmission Control Protocol

8.2 Seznam obrázků

8.3 Seznam tabulek

8.4 Seznam zdrojového kódu

8.5 Přílohy

Přílohy jsou dostupné na optickém médiu, které je přiložené v deskách této práce.

Příloha 1 – zdrojové kódy aplikace