



**Střední průmyslová škola na Proseku**  
**190 00 Praha 9, Novoborská 2**

## **OBOR**

**18-20-M/01 INFORMAČNÍ TECHNOLOGIE**  
**(ŠVP: INFORMAČNÍ TECHNOLOGIE)**

## **MATURITNÍ PROJEKT**

**TÉMA PRÁCE**

**Finance Curator**

**MÍSTO TOHOTO LISTU VLOŽTE LIST SE ZADÁNÍM**

## PROHLÁŠENÍ

Prohlašuji, že jsem svou maturitní práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW, atd.) uvedené v seznamu.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu § 60 zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne 2. 4. 2018

.....

podpis

## PODĚKOVÁNÍ

Tímto bych chtěl poděkovat vedoucímu mé práce, který mnohokrát poskytl cenné informace a připomínky, které mi pomohly vypracovat můj projekt lépe, než bych sám dokázal. Dále panu Ing. Jirímu Šilhánovi za cenné rady ohledně maturity a maturitní práce. Také bych chtěl poděkovat své rodině za jejich morální podporu a trpělivost.

**ANOTACE**

Jméno autora	Michal Moudrý
Název maturitní práce	Finance Curator
Rozsah práce	98 stran
Školní rok vyhotovení	2018
Škola	Střední průmyslová škola na Proseku
Vedoucí práce	David Malý
Využití práce	Mobilní aplikaci <i>Finance Curator</i> lze využít pro správu a kategorizaci osobních financí, včetně správy cestovních rozpočtů a účastníků.
Klíčová slova	Univerzální Windows Aplikace, správce financí, mobilní aplikace, .NET, .NET Native, Azure Mobile Apps, UWP, .NET Core
Anotace	Tento dokument obsahuje zpracované zadání praktické dlouhodobé maturitní práce na téma vývoj aplikace pro mobilní platformy, jejímž hlavním cílem je správa osobních financí a cestovních rozpočtů. Tato aplikace byla vyvíjena pro UWP a je tedy dostupná na různých typech zařízení.

**ANNOTATION**

Autor	Michal Moudrý
Title of graduation work	Finance Curator
Extend	98 pages
Academic year	2018
School	Střední průmyslová škola na Proseku
Supervisor	David Malý
Application	Mobile application <i>Finance Curator</i> can be used for managing and categorizing personal finances, including management of travel budgets and participants.
Key words	Universal Windows Application, finance manager, mobile application, .NET, .NET Native, Azure Mobile Apps, UWP, .NET Core
Annotation	This document contains finished assignment of long-term maturita work on topic of mobile application development, whose main goal is management of personal finances and travel budgets. This application was developed for UWP and so it is available on different types of devices.

## OBSAH

1.	Úvod .....	1
2.	Cíl práce.....	3
3.	Vývoj na UWP .....	4
3.1	Univerzální Windows Platforma .....	4
3.1.1	Design UWA.....	6
3.2	.NET.....	8
3.2.1	.NET Framework .....	8
3.2.2	.NET Core .....	10
3.2.3	.NET Native .....	11
3.2.4	.NET Standard.....	13
3.2.5	.NET Foundation.....	15
3.3	NuGet.....	15
3.4	Microsoft Azure .....	18
3.4.1	Azure App Service .....	18
3.5	MVVM.....	21
4.	Osobní finance .....	23
4.1	Výkaz peněžních toků.....	23
4.2	Rodinné příjmy .....	24
4.3	Rodinné výdaje .....	25
4.4	Finanční plán.....	25
4.5	Finanční rozvaha.....	26
5.	Technická dokumentace .....	28
5.1	Zdrojové soubory a složky projektu .....	28
5.2	Manifest aplikačního balíčku.....	34
5.3	Databáze.....	35
5.4	Azure App Service .....	37
5.4.1	Konfigurace služby .....	37
5.4.2	Konfigurace aplikace .....	41
5.4.3	Off-line používání a synchronizace .....	42
5.5	Použité technologie.....	43
5.5.1	NuGet balíčky .....	43

---

5.5.2	Rozšíření do IDE.....	44
5.6	Procesy na pozadí .....	45
5.7	Systémové API .....	46
5.7.1	JumpList API .....	46
5.7.2	Appointments API.....	46
5.7.3	Contact Manager API.....	47
5.8	Notifikace.....	48
5.9	Uživatelské nastavení aplikace .....	49
6.	Uživatelská dokumentace .....	50
6.1	Požadavky na zařízení .....	50
6.2	Konfigurace systému pro instalaci aplikace .....	51
6.3	Instalace aplikačního balíčku .....	53
6.4	Autentizace .....	55
6.5	Přehled financí .....	56
6.6	Vytvoření peněženky .....	59
6.7	Přidání finance do peněženky .....	60
6.8	Přehled cestovních rozpočtů .....	61
6.9	Vytvoření rozpočtu .....	62
6.10	Přidání účastníka k rozpočtu .....	63
6.11	Přehled kategorií .....	64
6.12	Vytvoření kategorie.....	64
6.13	Nastavení aplikace.....	65
6.14	Export dat aplikace.....	67
7.	Závěr.....	68
8.	Použitá literatura a zdroje .....	70
9.	Slovník pojmů .....	79
10.	Seznam použitých zkratk .....	82
11.	Seznam obrázků.....	84
12.	Seznam tabulek.....	86
13.	Přílohy .....	87



## 1. Úvod

Předmětem dlouhodobé maturitní práce je tvorba univerzální aplikace pro zařízení s operačním systémem Windows 10 jménem *Finance Curator*, která umožňuje uživateli evidovat osobní finance, společně s možným řazením do kategorií, které si může sám vytvořit. Aplikace také umožňuje správu cestovních rozpočtů s evidováním dluhů jednotlivých účastníků dané události. Dále synchronizaci daných dat mezi zařízeními, na kterých je aplikace nainstalována.

Aplikace je univerzální a je tedy schopna běžet na širokém spektru zařízení od mobilních telefonů až po zařízení s velkou úhlopříčkou, jako například Surface Hub, přičemž uživatelské rozhraní aplikace bylo vytvořeno jednotné pro všechny, a proto se uživatel nemusí starat o volbu konkrétního zařízení.

Aplikace využívá databázi pro ukládání záznamů lokálně a cloudovou službu pro synchronizaci se serverovým uložištěm. Dále také pracuje s různými systémovými API, runtime komponentami, procesy na pozadí a interaktivními či toast notifikacemi. Také zahrnuje práci grafickými prvky, jako například práce s vyskakovacími okny, animacemi, grafy nebo vlastními seznamy.

Aplikace bude plně lokalizována, v tomto případě bude aplikace podporovat češtinu a angličtinu, za pomoci hodnot uložených ve zdrojových souborech a ty se načítají za běhu aplikace. Vybrání jazyka, který se řídí základním nastavením manifestu aplikace, ale hlavně nastavením jazyka operačního systému, tedy jazyk aplikace je změněn automaticky bez zásahu ze strany uživatele ani programátora.

Díky lokální databázi a grafickým ovládacím prvkům umožňuje správu osobních financí, cestovních rozpočtů a kategorií. Finance jsou rozděleny na příjmy, výdaje, dluhy a trvalé platby. Evidence příjmů obsahuje informace o názvu příjmu, jeho hodnotě a datu přidání. Stejné atributy jsou obsaženy i v evidenci výdajů, ale v evidenci dluhů jsou přidány hodnoty o datu splacení konkrétního dluhu a v evidenci trvalých plateb jsou navíc informace o době, za kterou má být platba opět zaplacená. Přičemž každá finance může být přiřazena do kategorií.

Tyto finance jsou zobrazeny pouze v jednom seznamu, přičemž je vždy zobrazen jeden typ finance a jsou rozlišeny zobrazením rozdílných informací a barvou finance, kdy příjmy mají zelenou hodnotu a před ní je zobrazen znak plus. Výdaje společně s dluhy mají červené hodnoty a je zobrazen znak mínus. Trvalé platby mají hodnotu bílou a není zobrazen žádný znak. Finance, které jsou zobrazeny v seznamu je možno seřadit podle jejich atributů včetně specifických pro různé finance, a lze určit, jestli mají být seřazeny sestupně nebo vzestupně.

Evidence cestovních rozpočtů obsahuje informace o názvu rozpočtu, jeho celkové hodnotě a kolik je z něj zaplacen. Dále ke každému rozpočtu mohou být přiděleni účastníci, u kterých je evidováno jejich jméno, kolik dluží a kolik již splatili, přičemž je možno přidávat účastníky z uložených kontaktů na zařízení a následně jsou zobrazeny v seznamu, kde je jejich název a poměr mezi hodnotou a splacenou částkou. Dále u rozpočtu lze zobrazit jejich detaily, což zahrnuje informace o rozpočtu a seznam účastníků, kteří jsou k rozpočtu přiděleni.

Nepostradatelnou složkou aplikace je autentizace uživatele, která je řešena na straně cloudové služby, která pro danou funkci používá IDP třetí strany, v případě této aplikace se jedná o Microsoft Account.

V aplikaci je implementován systém exportu dat, které jsou v lokální databázi do textových souborů s formáty, jež jsou vhodné pro následné zobrazení v tabulkovém softwaru, jako například Microsoft Excel a další.

V tomto dokument je rovněž odborná část, jež souvisí s tématem aplikace, jedná se o dvě rešerše a těmi jsou: Vývoj na UWP, Osobní finance.

První rešerše se zabývá technologiemi, které souvisí s UWP, jedná se tedy o samotnou UWP a aplikacemi, jež jsou na ni vyvíjeny. Následně je část pro platformu .NET, kde je představena architektura a jednotlivé platformy ze kterých se skládá, ale je zde také popsáno, co je .NET Standard. Uvedeny jsou i technologie a služby kolem vývoje samotných aplikací, jako cloudová platforma Azure nebo systém pro distribuci knihoven NuGet. Hlavním záměrem této rešerše je seznámení s vývojem univerzálních aplikací pro Windows 10, což zahrnuje i technologie a služby pro podporu vývojářů při vývoji.

Rešerše na téma Osobní finance se zabývá tematickou stránkou aplikace, tedy osobními financemi, což zahrnuje témata jako výkaz peněžních toků, finanční plán, finanční rozvaha a rodinný rozpočet.

Dokument taky obsahuje technickou dokumentaci, která popisuje technickou realizaci aplikace, tedy implementaci nejdůležitějších částí, včetně omezení některých funkcí, následně líčení použitých technologií nebo postup při tvorbě architektury projektu. Účelem této části dokumentu je přednesení význačných funkcí pro jejich snazší pochopení, proto jsou vloženy ukázky kódu aplikace.

V rámci tohoto dokumentu je i uživatelská dokumentace, která líčí základní používání aplikace, včetně konfigurace a požadavků na zařízení, ale nacházejí se zde i pravidla pro určité části funkcionality, které potřebné pro jejich úplné používání. Jedná se tedy o text popisující jednotlivé funkce a operace, který je doprovázen obrázky pro jasnější pochopení daných možností aplikace.

## 2. Cíl práce

Cílem práce bylo vypracovat rešerše na témata *Vývoj na UWP* a *Osobní finance*, včetně naprogramování aplikace pro správu osobních financí, jež měla evidovat osobní finance zadaných uživatelem aplikace, jejich řazení do kategorií, následné zobrazení na časové ose v grafech, dále evidování cestovních rozpočtů a kategorií. Tato aplikace měla schopna běžet na různých typech zařízení s operačním systémem Windows 10.

Součástí splnění tohoto účelu byla implementace autentizace uživatele pomocí poskytovatele identit (IDP), společně s používáním bez připojení k počítačové síti a synchronizací dat vytvořených aplikací prostřednictvím cloudové služby.

Cílem bylo také získání znalostí z oblasti vývoje grafických aplikací v rámci platformy .NET, a to konkrétně vývoj aplikací prostřednictvím platform .NET Core a .NET Native. Složkou tohoto cíle bylo také se naučit pracovat s mnohými systémovými API, jak univerzálními, tak i API pro přidání funkcionality pro specifickou rodinu zařízení, například Jumplist API, pro přidání seznamu zkratk pro různé úlohy, dále Appointments API nebo Contact Manager API. Další složkou bylo porozumění životním cyklům aplikace, autentizace uživatele pomocí cloudové služby a přes danou službu i synchronizaci uživatelských dat, včetně konfigurace dané služby a částí, které byly pro její provoz nezbytné.

### 3. Vývoj na UWP

Kapitola *Vývoj na UWP* se zabývá tím, co je Univerzální Platforma Windows, dále také technologiemi či postupy, které lze nebo se využívají pro vývoj univerzálních aplikací, jako například platforma .NET nebo cloudová platforma Azure či systém pro distribuci balíčků NuGet a další.

#### 3.1 Univerzální Windows Platforma

O spojení operačních systémů Windows se společnost Microsoft snaží už mnoho let, kdy v roce 2011 desktopové a serverové systémy byly postaveny na kódu Windows NT, dále mobilní OS byl Windows Phone, což byl nástupce Windows CE a měl určité části společné s Windows NT, ale využíval jiný kód. Poslední byl OS pro Xbox 360, který sice v minulosti byl stejný, ale pak byl úplně přepracován. S příchodem Windows 8 byl unifikován kernel na všech platformách. Pak v červenci 2015 byl vydán Windows 10, který běžel na zařízeních jako herní konzole Xbox One, mobilní telefony, IoT, Microsoft HoloLens a další viz. Obrázek 1. Tento OS měl stejné jádro a kód na všech zařízeních, což vedlo k vytvoření UWP, také známé jako UAP. Další velkou změnou v tomto OS bylo vytvoření jednotného distribučního kanálu poskytovaného skrze Microsoft Store. (1) (2) (3)



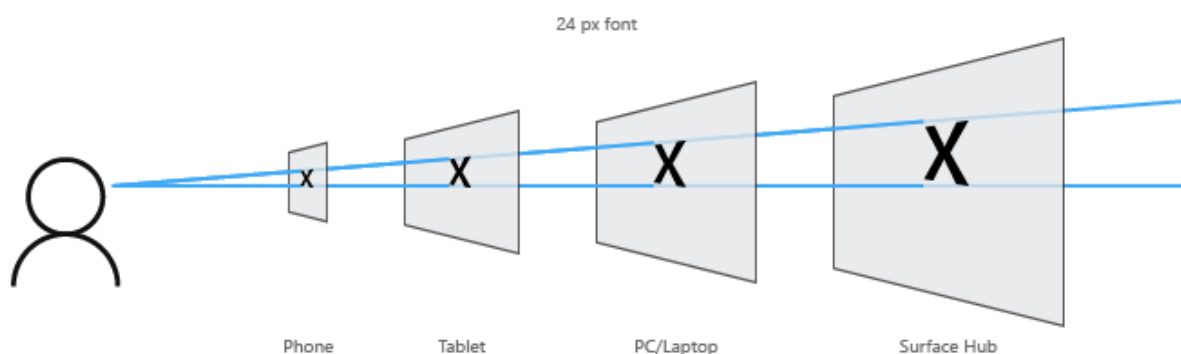
Obrázek 1 - Přehled UWP

Zdroj: (1)

Tato platforma není runtime, ale jedná se o aplikační model a API, které je zaručeně dostupné na podporovaných typech zařízení. Vývojář tedy nebude závislý na operačním systému, jako tomu bylo před tím. Místo toho se bude zaměřovat aplikační platformu, respektive na konkrétní sadu a verzi API. Ve výsledku lze tedy vytvořit jeden aplikační balíček, který lze nainstalovat na velké množství typů zařízení. (1) (3)

Aplikace vyvíjené na této platformě jsou označovány jako Universální Windows Aplikace ve zkratce UWA. Rozdíly mezi nimi a ostatními je právě ta samotná platforma, jež poskytuje jednotné API, ale také jsou distribuovány ve formátu .appx, který poskytuje důvěryhodný instalační proces a zajišťuje, že aplikace budou nasazeny a aktualizovány. Samozřejmě tyto aplikace mohou využívat i ostatní API (Win32 nebo .NET) než univerzální, třeba pro implementaci určité funkcionality na mobilním telefonu nebo na herní konzoli. Tyto rozhraní jsou zahrnuty v tzv. *Extension SDK*. Jelikož lze UWA instalovat na mnoho zařízení, tak UWP poskytuje ovládací prvky, jež se přizpůsobují zařízení a aplikace tedy podporují velké množství vstupů, jako klávesnice, dotek, herní ovladač a další. Pro přizpůsobení prvků UI se používá systém pro škálování a tzv. *Effective pixels* za účelem zlepšení čitelnosti a zjednodušení používání daných prvků. Tento systém využívá algoritmus, který pro optimalizaci uživatelského rozhraní vezme PPI zařízení společně s průměrnou vzdáleností mezi uživatelem a zařízením. Výsledkem bude například font o nějaké velikosti, jež bude stejně čitelný na telefonu s 5“ displejem a Surface Hub s 84“ displejem viz. Obrázek 2. Vývojář při tvorbě uživatelského rozhraní tedy nepracuje s fyzickými pixely ale s efektivními (epx), kdy se jedná o virtuální jednotku a používá se pro specifikování rozměrů nezávisle na PPI displeje zařízení.

(4) (5)

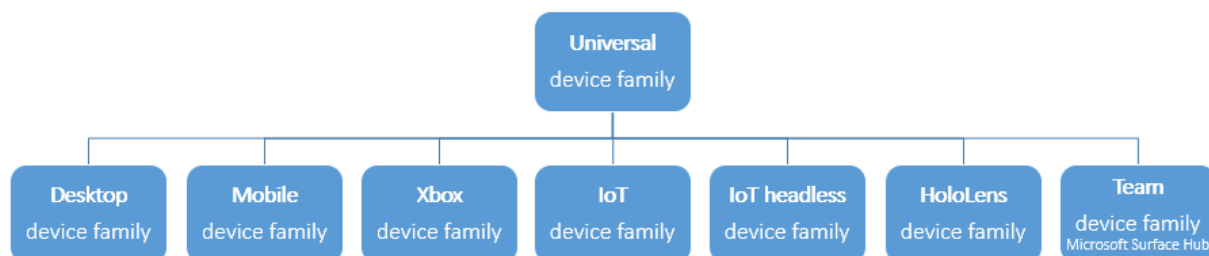


Obrázek 2 - Ukázka škálování fontu

Zdroj: (5)

Jelikož je důležité, aby aplikace běžely na různých typech zařízení, tak byl vytvořen koncept zvaný *Device families*, přičemž se jedná o označení skupin zařízení viz. Obrázek 3, respektive popis dostupných API, systémových vlastností a jejich chování na daném typu. Mezi ně patří kupříkladu *Desktop device family*, jež označuje desktop, laptop i tablet. Ve výsledku to znamená, že každá rodina přidává dodatečné rozhraní ke zděděným z *Universal device family*, která poskytuje API dostupné na všech zařízeních. Ovšem vývojář se musí rozhodnout, které bude podporovat a k tomuto rozhodnutí musí uzpůsobit výslednou aplikaci, tedy napsání tzv.

adaptivního kódu (využití API za podmínky její dostupnosti) nebo přizpůsobení UI pro jednotlivé druhy zařízení nebo vytvořit adaptivní uživatelské rozhraní k čemuž se použijí epx při určování takzvaných *breakpoints* viz. Tabulka 1. Problémem je, že univerzální API nemusí být kompletní na zařízení, což vývojář musí vzít v úvahu při rozhodování, kdy například na herní konzoli Xbox není třída *Web Account Manager*, která se nachází v ostatních rozhraní.



Obrázek 3 - Přehled rodin zařízení

Zdroj: (3)

Označení třídy	Šířka v epx	Úhlopříčka	Typické zařízení
Malé	<640px	4“ – 6“	Telefon
Střední	od 641px po 1007px	7“ – 12“	Tablet
Veliké	>1007px	13“ a větší	Desktop, laptop, Surface Hub

Tabulka 1 - Přehled tříd velikostí

Zdroj: (6)

### 3.1.1 Design UWA

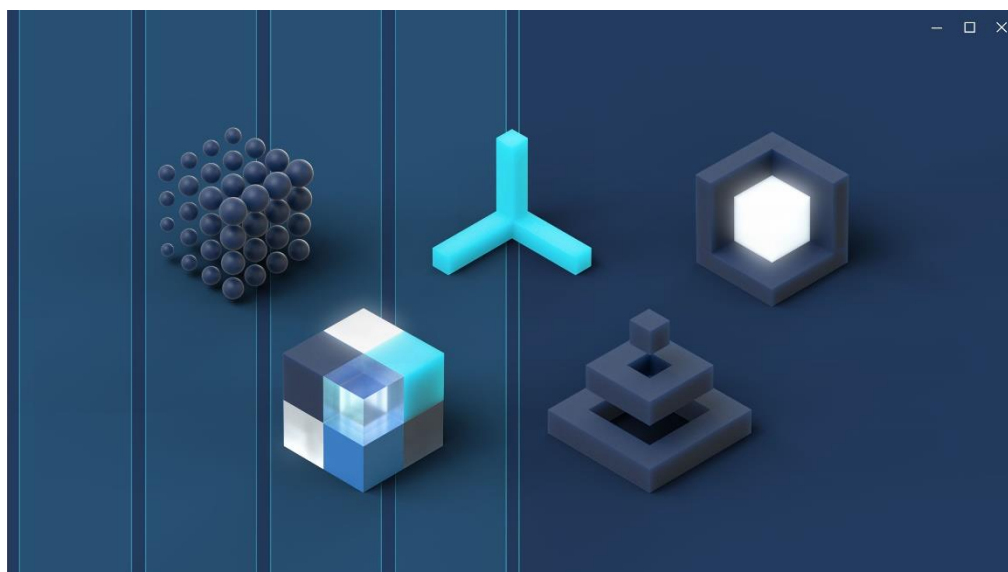
Společnost Microsoft už od poloviny 90. let minulého století vyvíjí grafický styl známý jako *Metro*, kdy se objevil v produktech jako Zune, Windows Media Center a později byl využit v samotném operačním systému, ale nejvíce se začal prosazovat od vydání Windows Phone 7. Stejně jako samotný design tak i jeho název se časem změnil a v dnešní době se nazývá *Microsoft Design Language* ve zkratce MDL. (7) (8) (9)

Do principů grafického stylu *Metro* se řadí zmenšení počtu zbytečných prvků, snaha zaměřit se pouze na hlavní úkony a rychlost aplikací. Další je redukce prvků, které nejsou obsah, a tedy obsah je UI, také má tento design dodat rozměr a hloubku uživatelskému rozhraní. Velkou součástí tohoto stylu je typografie, kdy cílem je poskytnout informace přímo a čistě, proto byla vytvořena skupina fontů *Segoe*. (8) (9)

Tento styl přinesl mnoho změn, jako určení navigačního systému, který byl stylem *Hub*, tedy zobrazení několika kategorií obsahu vedle sebe. Dále to bylo přidání gest zvláště pro zařízení s dotykovým displejem. Největší změnou byly dlaždice jako prvky systémového startovního menu viz. Obrázek 5. (8)

**Obrázek 5 - Ukázka grafického stylu Metro****Zdroj: (109)**

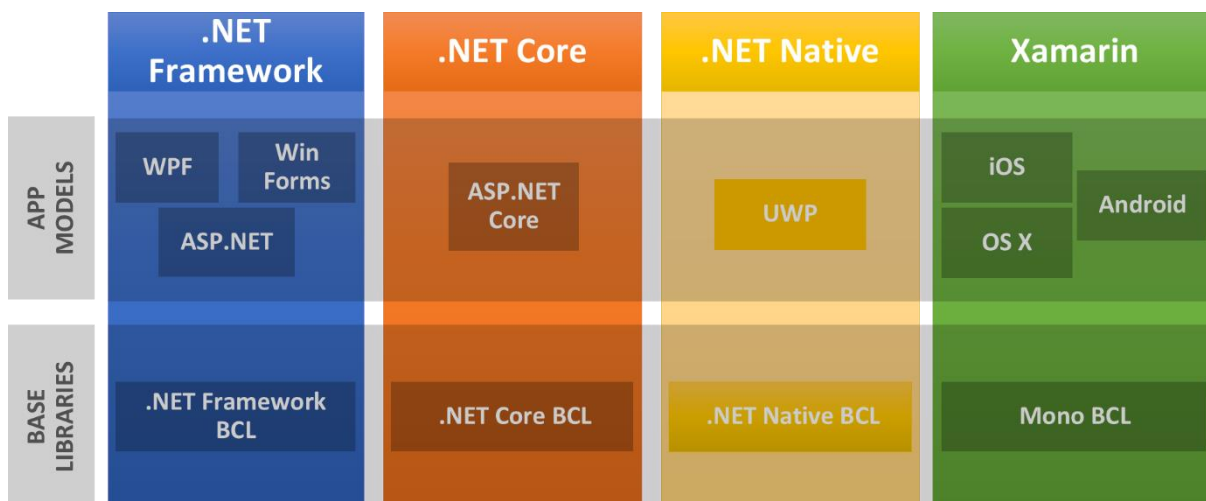
V roce 2015 vyšel MDL ve verzi 2.0, který prakticky nezměnil principy stylu, ale bylo upraveno mnoho ovládacích prvků a částí systému, jako například notificační centrum. Tato verze byla později rozšířena o tzv. *Fluent Design System* viz. Obrázek 4. Toto rozšíření doplňuje grafický styl o efekt odhalení, který má sloužit pro získání pozornosti na informaci, dále prvky pro přidání hloubky, jako tzv. *Acrylic* materiál a *Parallax* efekt. Posledním doplňkem jsou *Connected animations*, které pomáhají udržet pozornost uživatele vytvoření plynulého přechodu. (10) (11)

**Obrázek 4 - Ukázka aplikace implementující "Fluent Design System"****Zdroj: (11)**

## 3.2 .NET

Tato kapitola pojednává o platformě .NET, která byla vydaná společností Microsoft, respektive o architektuře dané platformy, jednotlivých platformách pro vývoj různých aplikací a .NET Standard, ale také o .NET Foundation, která stojí za rozvojem této platformy. Samotná platforma je zdarma, funguje napříč několika platformami a podporuje mnoho programovacích jazyků a nástrojů pro vývoj aplikací, jako například C#, F# nebo VB. Dokonce je uvolněna pod open source licencí, kdy se na vývoji podílí přibližně 25 000 vývojářů a 1 700 společností, které jsou součástí .NET Foundation a daný ekosystém se skládá zhruba z 515 repositářů a 55 členských projektů. (12) (13)

V dnešní době existují čtyři platformy v rámci .NET, tedy .NET Framework, .NET Core, .NET Native a Xamarin, přičemž každá platforma obsahuje jiné aplikační modely (např. WPF) viz. Obrázek 6. (14)



Obrázek 6 - Přehled platformy .NET

Zdroj: (14)

### 3.2.1 .NET Framework

Jedná se o platformu, která je částečně open-source a slouží pro vývoj aplikací, které jsou dostupné pouze na systémech s OS Windows. Účelem tohoto prostředí je správa aplikací, které se na danou platformu zaměřují. Skládá se z Common Language Runtime (CLR), jež slouží pro správu systémových služeb a paměti cílového zařízení, který spravuje běžící aplikace. Dále je obsažena rozsáhlá knihovna tříd, které jsou testované a lze je použít pro snazší vývoj hlavních částí aplikací. (15)

Mezi hlavní funkce této platformy patří správa paměti, kdy programátoři nejsou odpovědní za alokaci a uvolňování paměti nebo za správu životních cyklů objektů v paměti, jako to může být u jiných jazyků, ale CLR tuto funkcionalitu poskytuje na straně aplikace. Další



funkcí je již zmíněná knihovna se třídami, které slouží pro běžné programovací operace, například I/O operace, reprezentaci základních datových typů a výjimek nebo zapouzdření datových typů. Také významnou funkcí je interoperabilita programovacích jazyků, kdy kompilátor jazyku vydá Common Intermediate Language (CIL), který je pak zkompilován CLR, což umožňuje využívat knihovny, které jsou napsané v jiném jazyce, než je samotná aplikace. (15) (16)

Platforma .NET Framework z pohledu programátora poskytuje prostředí pro vývoj aplikací libovolným jazykem, díky jejich nezávislosti a interoperabilitě. Velkou výhodou je malá šance, že daná platforma nebude obsažena na uživatelském zařízení, protože je nainstalována společně s OS a od Windows 8 je dokonce komponentou samotného operačního systému. Co se týče požadavků na vývoj aplikací nebo komponent musí mít programátor nainstalovanou verzi platformy, kterou chce zaměřit, dále také musí splňovat hardwarové a instalační požadavky viz. Tabulka 2. A samozřejmě programátorem preferované vývojové prostředí.

HW parametr	Požadavek
Pracovní frekvence procesoru	1 GHz
Velikost operační paměť	512 MB
Místo na disku	4,5 GB

**Tabulka 2 - Minimální HW požadavky od verze 4.5**

**Zdroj: (17)**

Z pohledu uživatele cílového zařízení, tak ani nemusí vědět, že .NET Framework má nainstalovaný a jelikož je od určitých verzí jako komponenta OS, tak se nemusí starat o jeho instalaci, pouze v případě instalace aplikace, která by vyžadovala specifickou verzi, která se na daném zařízení nenachází. (15)

Samozřejmě se platforma vyvíjí společně nástupem nových technologií pro přidání podpory pro dané technologie, proto k normálním vydáním platformy jsou vydávány takzvané Out-of-Band (OOB) vydání pro zlepšení vývoje a přidání nových funkcí. Výhodou je vydávání častých aktualizací a rychlejší odpověď na zpětnou vazbu od zákazníků. Ovšem uživatelé nemusí aktualizovat .NET Framework na jejich zařízení, ale OOB sestavení jsou součástí aplikačního balíčku. (18)

### 3.2.2 .NET Core

.NET Core je stejně jako .NET Framework platforma pro vývoj aplikací, ale s tím rozdílem, že podporuje více operačních systémů a nejen Windows, viz. Tabulka 3, ovšem podporuje méně aplikačních modelů. A může být použita na normálních zařízeních, v cloudu nebo na embedded/IoT zařízeních, přičemž nemá žádné konkrétní požadavky na samotné nasazení. Části této platformy mohou být také využity jinými platformami v rámci .NET, kdy například aplikace, které budou využívat .NET Native budou využívat CoreCLR pro ladění aplikace z důvodu jednoduchosti kompilace a množství nástrojů pro ladění. (19) (20) (21)

Co se týče výhod této platformy, tak jednou z nich je flexibilní nasazení, může tedy být součástí aplikace nebo nainstalovaná side-by-side, což umožňuje testování více verzí pro vybírání kandidátní verze. Dále jsou výhodou nástroje příkazové řádky, které umožňují vykonání všech produktových scénářů skrze příkazovou řádku. Samotná platforma je open-source a je uvolněna pod MIT a Apache 2 licencí a dokumentace je pod CC-BY licencí. (19) (20)

OS	Verze	Architektura
Windows Client	7 SP1+, 8.1	x64, x86
Windows 10 Client	Build 1607+	x64, x86
Windows Server	2008 R2 SP1+	x64, x86
Fedora	26, 27	x64
Debian	8.7+, 9	x64
Mac OS X	10.12+	x64

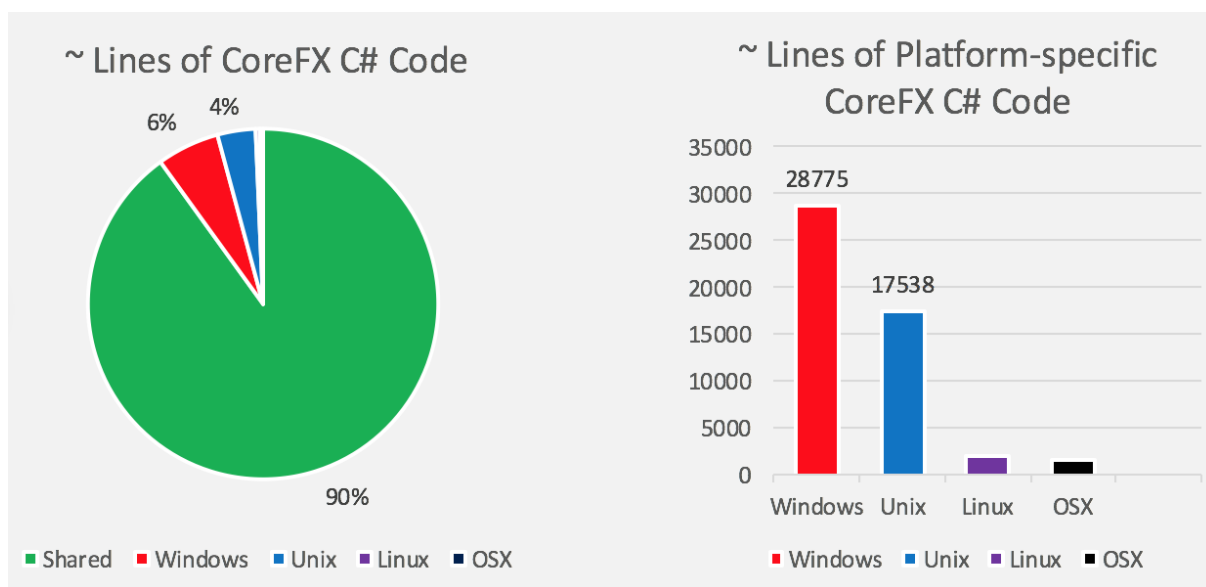
**Tabulka 3 - Podporované operační systémy v rámci .NET Core 2.0**

**Zdroj: (22)**

.NET Core se sestává z .NET Runtime, také známý jako CoreCLR, který poskytuje systém typů, správu paměti, nativní interoperabilita, načítání sestavení a další základní služby. Pak jsou framework knihovny, které poskytují základní datové typy a nástroje. Následně je sada SDK nástrojů, jako například .NET Command-line nástroje pro postavení, správu a obecně provádění operací s .NET projekty. Poslední částí je aplikační host, který slouží pro spouštění .NET Core aplikací, což zahrnuje vybrání runtime a hostování runtime, pak také poskytuje pravidla pro načítání sestavení. (19) (20) (23)

Tato platforma zahrnuje pouze jeden aplikační model a tím jsou konzolové aplikace, ale nad .NET Core byly postaveny další aplikační modely pro rozšíření její funkcionality mezi tyto modely patří například ASP.NET Core, UWP nebo také Xamarin.Forms, pokud zaměřuje UWP. (20)

Co se týče architektury, tak největším problémem je poskytování funkcí, které jsou specifické pro určité platformy, což je docíleno kódem pro dané platformy, přičemž obecné funkce jsou zajištěné pomocí sdíleného kódu, viz. Obrázek 7. Samotná architektura byla postavena s myšlenkou adaptability na nové operační systémy, nadstavby a nové sady nástrojů pro kompilátory. Dále je platforma rozdělena na několik částí pro snazší přizpůsobení daných částí na nové systémy. (20)



Obrázek 7 - Statistiky kódu v CoreFX

Zdroj: (20)

### 3.2.3 .NET Native

.NET Native, dříve známý jako *Project N*, je AOT kompilační technologie pro vytváření UWP aplikací pro operační systém Windows. Jedná se tedy o sadu nástrojů, která automaticky zkompile verzi aplikace pro vydání a zaměřuje .NET Framework a Windows 10, což znamená zkompileování IL binárních souborů na nativní binární soubory. Výsledkem dané kompilace jsou spustitelné soubory pro jednotlivé platformy, například x86 nebo ARM. (21) (24) (25) (26)

Ovšem myšlenka kompilace na nativní kód už byla realizována u .NET Framework, kdy byla vytvořena technologie NGEN, jež slouží pro kompilování sestavení na nativní kód, respektive *Native images* a vkládá je do tzv. *native image cache* na cílovém zařízení. Přestože je NGEN v několika ohledech podobný s .NET Native, tak se liší výrazně v mnoha aspektech, jako například absence nativního kódu pro nějakou metodu, a tedy tato technologie se vrátí k JIT kompilaci kódu, ale na druhou stranu .NET Native nemění kompilaci kódu. Mezi další rozdíly patří nespolehlivost NGEN v případě změn v závislostech sestavení. (27)

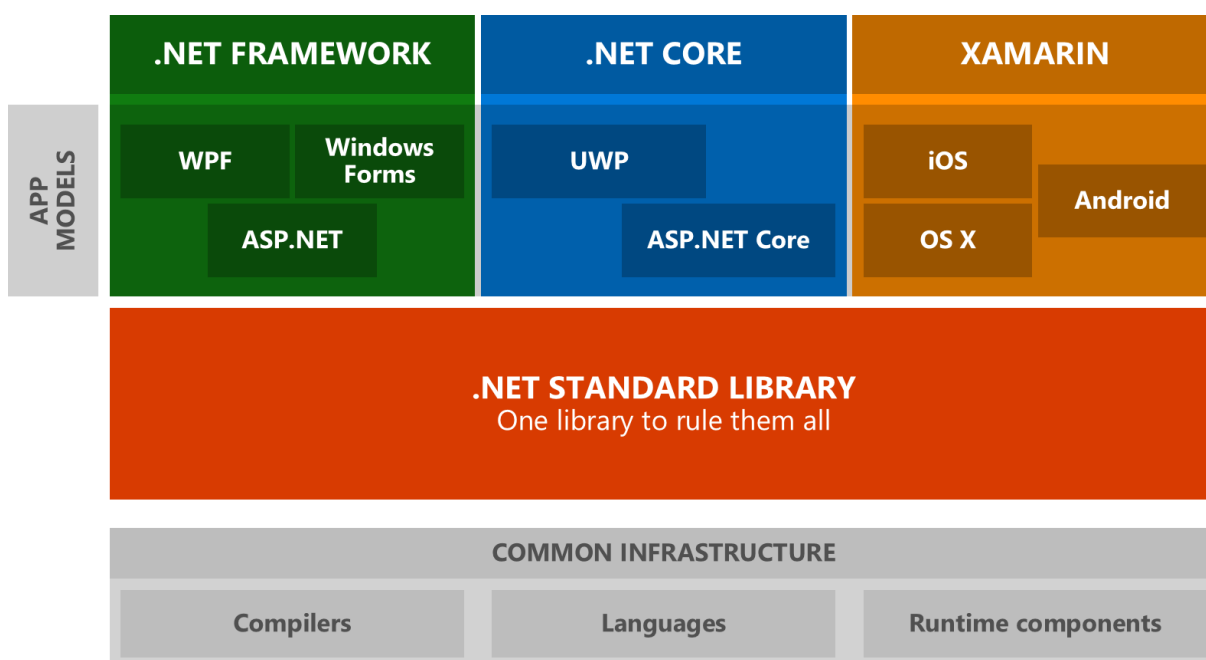
Samozřejmě .NET Native poskytuje mnoho výhod, jak pro aplikace či programátory, tak i pro samotné uživatele. Mezi ně patří například rychlejší čas spuštění aplikace, protože aplikace vykonávají již zkompilovaný kód a nemusejí tedy generovat strojový kód a ani načítat JIT kompilátor výsledkem je přibližně o 60 % zlepšení výkonu u cold startup a o 40 % zlepšení u warm startup. Také je na zařízení přítomna sdílená knihovna (SharedLibrary.dll), do které jsou kompilovány AOT základní části všech aplikací (např. třída *String*) a je sdílená mezi všemi aplikacemi na daném zařízení. Aplikace tedy neobsahuje daný kód, ale vykonává volání na danou knihovnu a samotná aplikace je kompilována nativně z čehož vyplývá, že zabírá méně operační paměti a využívá optimalizující kompilátor, jenž má za výsledek výkonnostní výhody. Takže programátor má při vývoji všechny výhody jazyka C# nebo VB a nástrojů s nimi spojenými, přičemž daný kód má výkon jako C++. Další výhodou je možnost přenášení pouze jednoho spustitelného souboru mezi různými zařízeními bez závislosti na jiných technologiích, to je docíleno nezávislostí na .NET Runtime a tím, že kompilátor vygeneruje jediný soubor, jež zahrnuje jak samotnou aplikaci, tak i CoreRT a její závislosti. (21) (25) (26) (28)

Pro zkompilování aplikace pomocí .NET Native kompilace, a ne pomocí CoreCLR, je potřeba ve IDE Visual Studio přepnout *Debug* nastavení na *Release*. Protože v *Debug* módu IL je zkompilován právě pomocí CoreCLR, který je uvnitř aplikace. Toto nastavení se využívá pro ladění kódu, kdy nasazení a kompilace trvá kratší dobu s lepšími nástroji pro ladění aplikace. Samozřejmě je taky nutné otestovat verzi pro jejíž kompilaci byl využit .NET Native kompilátor, protože by bylo vhodné zjistit, zda se nenaskytly problémy s ním spojené, například problémy se třídami v rámci jmenného prostoru *System.Reflection*. (21)

Důležitým aspektem .NET Native je tvorba aplikačních balíčků. Ve výsledku existují dva balíčky, jeden pro distribuční systém Microsoft Store s koncovkou .appxupload a druhý pro instalaci aplikace bokem. Store balíček obsahuje pouze IL, a ne nativní binární soubory, jako tomu je u .appx balíčků. Toto přineslo velký dopad na vývojáře, kdy oni odesílají pouze IL na Microsoft Store, jež hostuje na serveru kompilátor, který ho následně zkompiluje na všechny cílené platformy. Následující změnou je automatická iterace balíčku, dostupné jsou jenom první tři čísla v rámci verze a čtvrtý je rezervovaný pro případ znovu kompilování na serveru. (21) (26)

### 3.2.4 .NET Standard

Jak jsem již uvedl, tak v rámci .NET existuje mnoho platforem, které podporují různé operační systémy a vznikl požadavek pro zabránění rozdělení daných platforem, což znamenalo je sjednotit. Řešením tohoto problému je .NET Standard, kdy se tedy jedná o specifikaci, jež určuje, jaké API musí .NET platformy obsahovat viz. Obrázek 8, což zahrnuje ne jenom jaké API, ale také jejich strukturu, kdy některé rozhraní nemuseli být všude stejné. Není to tedy pro aplikace, ale pro knihovny pro dané aplikace. Dalším důvodem vzniku této specifikace bylo vytvoření nástrojů napříč platformami, kdy se jednalo o zjednodušení ve smyslu zaměření společných částí. (29) (30)



Obrázek 8 - Výsledek implementování .NET Standard v .NET platformě

Zdroj: (30)

Přestože se jedná o specifikaci pro knihovny, tak i vývojáři aplikací získají určité výhody, jako například využití nějaké knihovny na mobilním zařízení, jež bude stejná jako na desktop zařízení, samozřejmě také záleží, na jakou platformu je aplikace vyvíjena. (30)

Verze .NET Standard	1.0	1.1	1.2	1.3	1.4	1.5	1.6	2.0
.NET Core	-	-	-	-	-	-	1.0	2.0
.NET Framework	-	4.5	4.5.1	4.6	4.6.1	4.6.1	4.6.1	4.6.1
Xamarin.Android	-	-	-	-	-	-	7.0	
.NET Native	-	-	-	-	10.0			

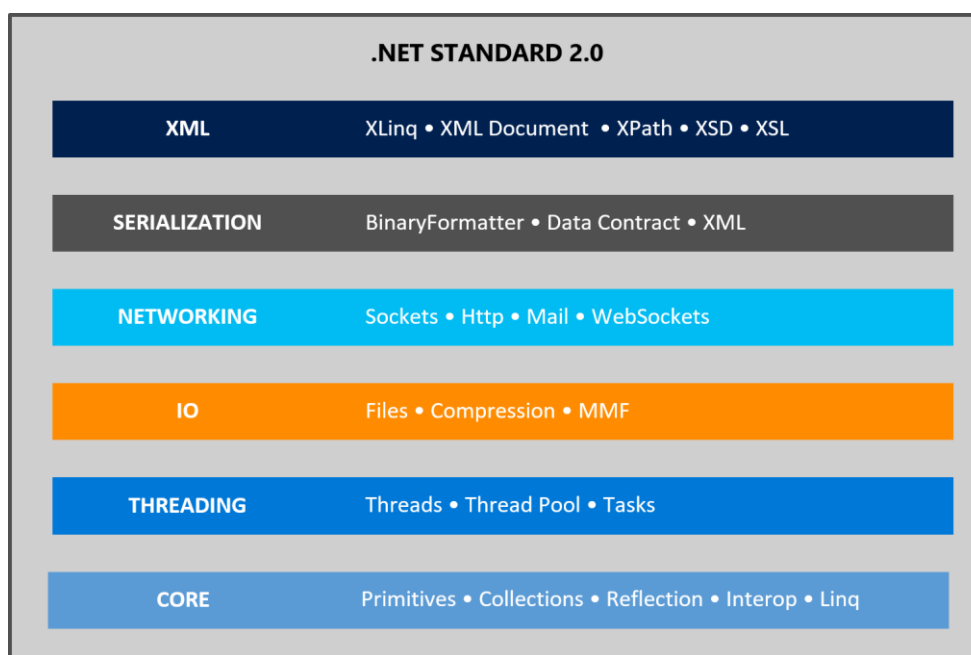
Tabulka 4 - Přehled kompatibility vybraných platforem s verzí .NET Standard

Zdroj: (14)

Verzování .NET Standard funguje na principu toho, že každá nová verze musí obsahovat API z verze předchozí pro zachování kompatibility při přechodu na další verzi, kdy například v rámci projektu zaměřující .NET Standard 2.0 je možné se odkazovat na balíčky zaměřující verzi 1.6. (29)

Velkým milníkem .NET Standardu bylo vydání verze 2.0, jejíž největší změnou bylo přidání kompatibility .NET Framework ve verzi 4.6.1, což je významné z důvodu toho že daná verze je hojně využívána a obsahuje více API než například .NET Core. Dalšími změnami je více než zdvojnásobení rozhraní, které jsou ve specifikaci zahrnuta, následně umožnění odkazovat na knihovny, které neimplementují specifikované rozhraní a Portable Class Libraries (PCL), jako například odkazování na existující .NET Framework kód bez nutnosti rekompile. S touto verzí je přibližně 70% NuGet balíčků kompatibilních a je zde okolo 20 tisíc více API než ve verzi 1.6. (14) (29) (30)

Pro .NET Standard je též důležité, co v dané specifikaci je a kdo o tom rozhodne. Pro určení, co v něm bude je potřeba klasifikovat již existující API jak v .NET Framework, tak i v Xamarin, přičemž jsou rozděleny do skupin potřebných a volitelných rozhraní. V případě potřebných se jedná o API, které jsou nezbytná na všech platformách a volitelné jsou API, které jsou většinou specifické na nějakou platformu, nebo se jedná o část zastaralé technologie a nejsou tedy součástí specifikace a jsou pak distribuovány jako NuGet balíčky. Což logicky zahrnuje odebrat části potřebných rozhraní, které využívali zastaralé API. (30)



Obrázek 9 - Shrnutí API v .NET Standard 2.0

Zdroj: (30)

### 3.2.5 .NET Foundation

Jedná se o nezávislou organizaci, založenou společností Microsoft, pro otevřený rozvoj platformy .NET a technologií kolem dané platformy. Účelem je rozšířit a vylepšit .NET ekosystém za pomoci komunity a členů takzvané *Technical Steering Group*, do níž patří například společnost Samsung, Red Hat nebo Google. Mezi další účely .NET Foundation patří podporování různého softwaru pro .NET vývojáře, ale také poskytování administraci projektů, které jsou spojeny s organizací. Rovněž poskytují konzultace a rady ohledně vedení otevřených komunitních projektů nebo též marketing pro projekty, kdy se .NET Foundation společně s ostatními partnery snaží pro ně získat pozornost.

Jak již bylo zmíněno, tak součástí .NET Foundation je *Technical Steering Group*, jež se sestává z velkých společností. Cílem této skupiny je pomoci .NET Foundation s kontrolou a koordinací základních projektů v rámci nadace. Mají tedy velký vliv na rozhodnutí ohledně podpory různých platforem a na budoucí naplánované funkce, ale také má vliv na nové projekty nebo na změny v programovacích jazycích.

### 3.3 NuGet

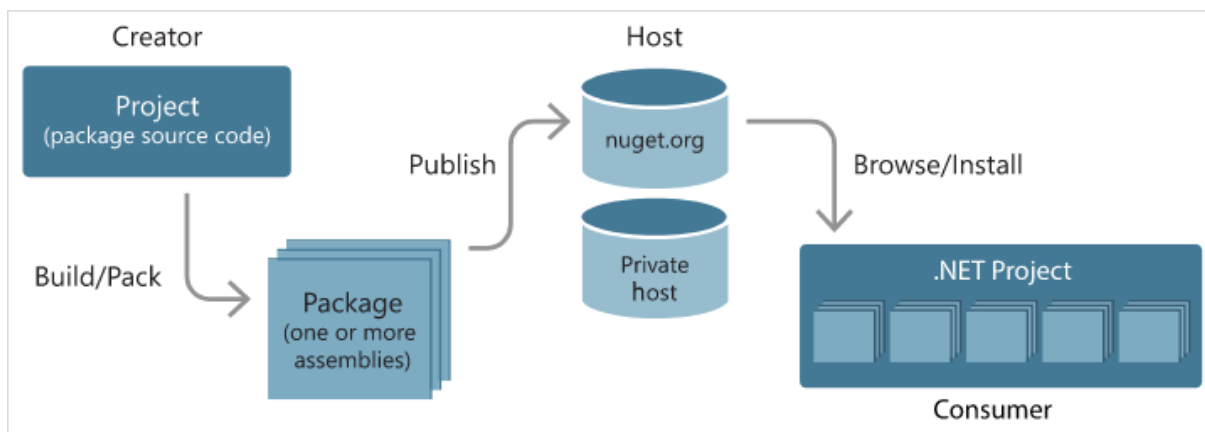
Při vývoji softwaru jsou důležité knihovny, které pomáhají při tvorbě programů poskytováním kódu, jež řeší nějaké konkrétní úlohy. A pro jejich tvorbu, sdílení nebo získávání je potřeba platforma pro dané funkce. Nejčastěji je kód šířen jako tzv. balíčky společně s ostatními důležitými věcmi pro projekt. V případě .NET platformy se jedná o systém NuGet. Tento systém vymezuje, jak balíčky jsou vytvářeny, hostovány, distribuovány a pro jednotlivé části poskytuje vhodné nástroje třeba dotnet CLI, Package Manager Console a další, přičemž jejich dostupnost záleží na platformě, kterou vývojář využívá viz. Tabulka 5. (31)

Nástroj	Platforma
nuget.exe CLI	Všechny
dotnet CLI	Všechny
Package Manager Console	VS na Windows
Package Manager UI	VS na Windows
Manage NuGet UI	MS na Mac

Tabulka 5 - Nástroje systému NuGet a jejich dostupnost

Zdroj: (31)

Proces tvorby balíčku až po jeho šíření se sestává z vytvoření knihovny tříd, následně její zabalení a publikování na veřejného hosta, což může být například nuget.org nebo na privátního hosta, a nakonec je konzumování balíčků na straně vývojáře, respektive jeho instalace do projektu viz. Obrázek 10. (31)



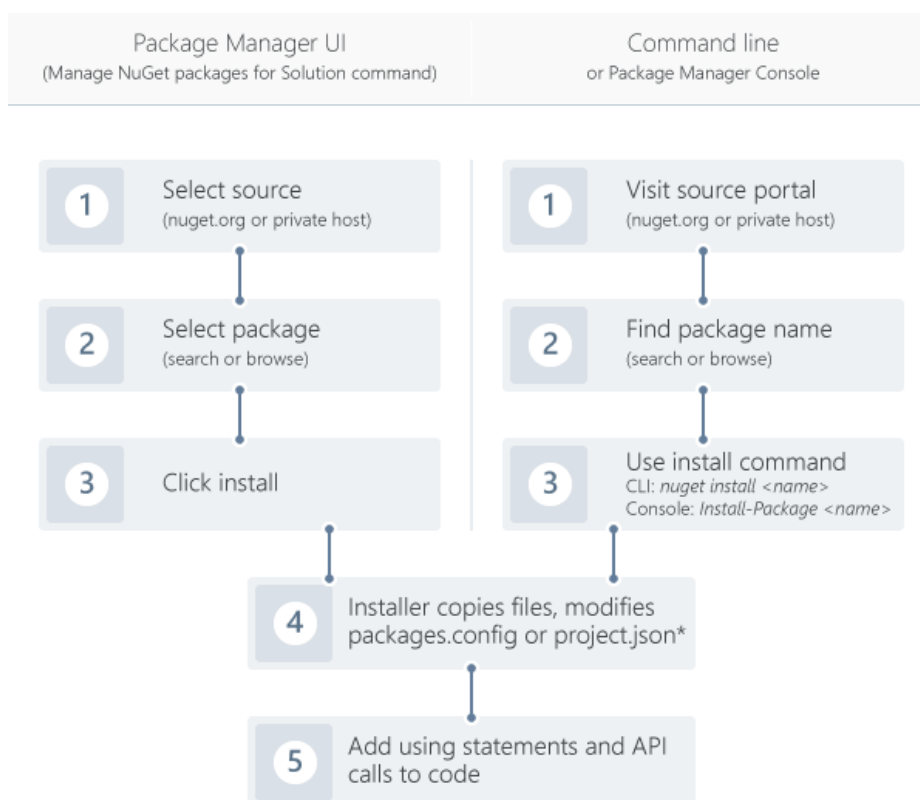
Obrázek 10 - Proces tvorby, publikování a distribuce NuGet balíčků

Zdroj: (31)

Stejně jako aplikace tak balíčky mohou využívat další balíčky, které se označují jako závislosti, ale zákazník se stará pouze o ním instalovanou knihovnu, tedy tzv. *Top-level dependencies* a veškeré *Down-level dependencies* řeší NuGet. Například ve stromu závislostí může být více stejných balíčků, ale ve výsledku je zvolen pouze jeden, jež je nejvhodnější pro potřeby zákazníka, kdy se o toto zvolení stará také NuGet. (31)



Samotné balíčky lze získat mnoha způsoby, kdy se může jednat o instalaci přes příkazový řádek (CLI) nebo přes uživatelské rozhraní. V případě uživatelského rozhraní se jedná o Package Manager UI, jež umožňuje vyhledávat či instalovat balíčky s jejich závislostmi a přidá na něj odkazy do souborů v projektu. Na druhou stranu CLI, nemusí provádět změny v souborech projektu, což je případ `nuget.exe` CLI, ale ostatní jako `dotnet.exe` nebo Package Manager Console dané změny provádí. Celý proces se sestává z jakékoliv formy vybrání a instalace balíčku, následně instalační systém zkopíruje soubory a případně upraví projektové soubory viz. Obrázek 11. (32) (33)



**Obrázek 11 - Proces přijímání NuGet balíčků**

**Zdroj: (33)**

Již zmínění privátní hosté umožňují hostování balíčků pouze určitým lidem, což může zahrnovat například organizace, kdy může být vhodné omezit určité knihovny třetích stran. Zdroje takových balíčků je třeba složka na síťovém uložišti nebo mohou být dostupné přes lokální http server a také přes galerii NuGet. Existuje mnoho produktů, jež podporuje jejich správu a mezi tyto produkty patří VSTS, NuGet Server, Nexus a další. (34)

Kompatibilita balíčků spočívá v obsahu, kdy musí obsahovat sestavení alespoň pro jednu verzi .NET Framework, která je kompatibilní s cílenou verzí projektu. V případě většího pokrytí může vývojář knihovny cílit verzi již zmíněného .NET Standard. (31)

### 3.4 Microsoft Azure

Jedná se o cloudovou výpočetní platformu od společnosti Microsoft poskytující komplexní sadu služeb, jež umožňuje sestavovat, nasazovat a spravovat jednoduché i složité aplikace. (35)

Mezi hlavní znaky Azure patří zvýšení produktivity vývojářů, kdy lze aplikace vyvíjet v mnoho programovacích jazycích jako například C#, Node.js a další za použití nástrojů, které jsou dostupné na několika operačních systémech. Dále obsahuje více než 100 služeb, jež se řadí do kategorií jako *Compute*, jež zahrnuje App Service, VM (Windows i Linux) nebo Container Instances. Samozřejmě MS Azure poskytuje služby okolo úložiště tedy škálovatelné cloudové úložiště, zálohování serveru, diskové úložiště, File Storage nebo Blob Storage. Ovšem jsou zde i služby okolo počítačových sítí, kdy lze hostovat DNS doménu, provozovat CDN systémy či VPN nebo Application Gateway. (36) (37) (35) (38)

Dalším znakem je hybridnost této platformy, kdy lze sestavovat a nasazovat řešení na veřejném nebo privátním cloudu. Co se týče tzv. *On-premises* prostředí tak pro zrychlení inovací cloud computing lze využít rozšíření Azure Stack, které umožňuje poskytovat služby Azure z vlastního datacentra při balancování flexibility a možnosti ovládání. (39) (40)

Posledním znakem je důvěryhodnost, tedy splnění mezinárodních a oborových standardů pro dodržování předpisů organizací, jako CSA/CCM, ITAR nebo ISO/IEC. Důvěryhodnost platformy je také zajištěna audity třetích stran s cílem ověření, zda byly provedeny bezpečnostní opatření vyžadovaná standardy ISO 27001 nebo HIPAA či G-Cloud. Dále to také zahrnuje například monitorování stavu bezpečnosti v celém prostředí napříč platformou. (41)

#### 3.4.1 Azure App Service

Tato PaaS služba v rámci platformy Azure poskytuje schopnosti pro aplikace, webové stránky. Mezi ně patří monitorování a upozorňování pro aplikace za pomoci služby AppInsights, již zabudované automatické škálování, CD, vyvažování zátěže a monitorování výkonu, také nové možnosti hostování, kdy byly přidány funkce jako Web Jobs, zajištění stálého běhu backend a další. (42) (43)

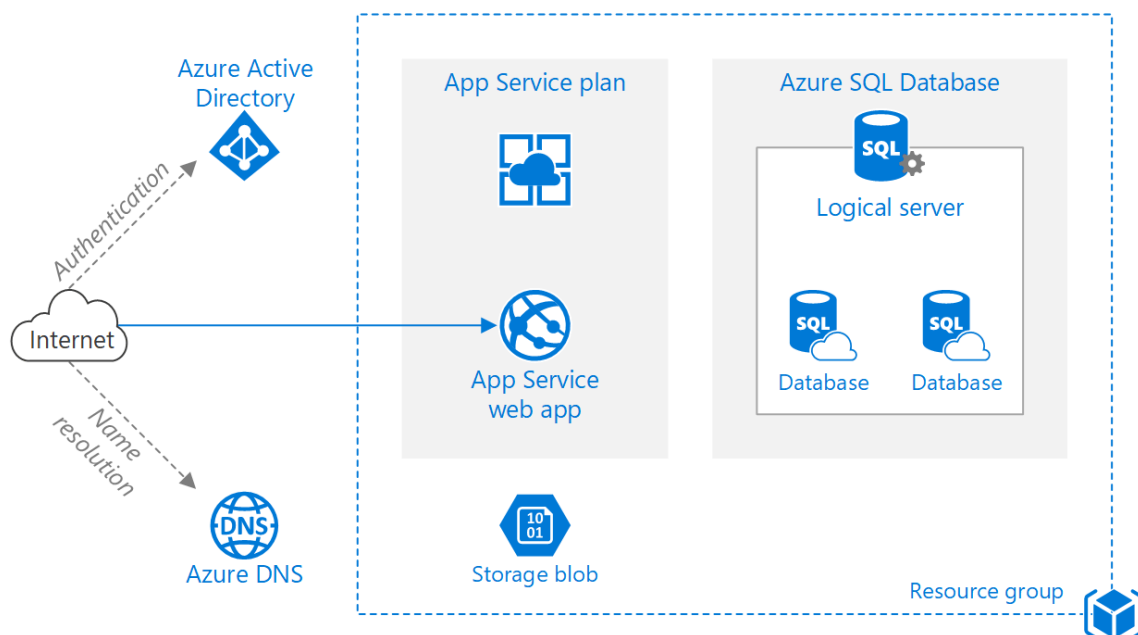
Funkce Mobile Apps této služby poskytuje platformu pro vývoj aplikací viz. Obrázek 12, která umožňuje autorizaci a ověření uživatelů společně s podporou poskytovatelů identit, přičemž pro každého je přidána OAuth 2.0 služba. Následně je funkce off-line práce s daty a jejich synchronizace, kdy aplikace pracuje s lokální databází a provedené změny lze synchronizovat se serverovým backend včetně řešení konfliktů. Další jsou push notifikace a ty je možno přes Azure Notification Hub poslat všem uživatelům. Nezbytnou součástí Mobile Apps je Client SDK, které je open-source a je jak pro vývoj nativních aplikací, tak i pro cross-platform. (43)



**Obrázek 12 - Mobile Apps – platforma pro vývoj aplikací**

**Zdroj: (43)**

Výsledná architektura aplikací, třeba webových viz. Obrázek 13 se skládá z Azure DNS, což je hostitelská služba DNS domén a slouží pro jejich překlad. Pak je IDP, který ukládá informace o uživatelských účtech a ověřuje uživatele, přičemž v základu je 5 podporovaných poskytovatelů tedy AAD, Microsoft Account, Google, Facebook, Twitter. Následně je skupina prostředků, což je logický kontejner prostředků Azure. V tomto kontejneru se nachází App Service Plan, který poskytuje VM pro hostování aplikace a nachází se v něm aplikace App Service, přičemž všechny aplikace běží na stejných instancích. Další částí kontejneru je Azure SQL Database, která se skládá z logického serveru, který hostuje jednu nebo více databází. Poslední částí je Storage blob pro ukládání diagnostických logů. (44) (45)



Obrázek 13 - Ukázka architektury webové aplikace

Zdroj: (45)

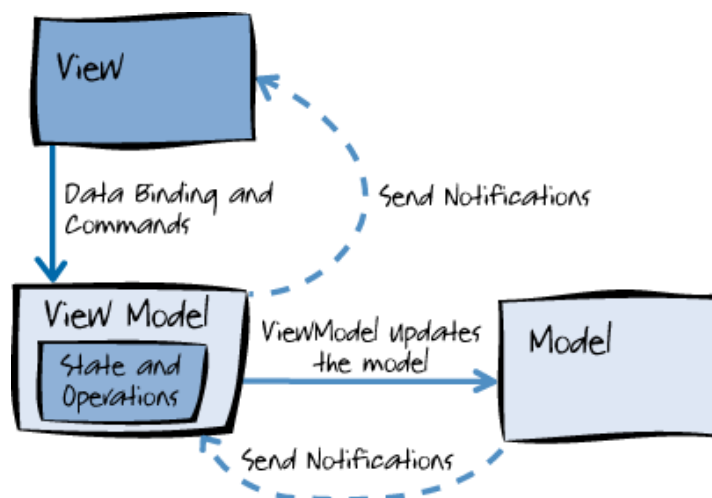
App Service Plan, ve kterém může běžet více než jedna aplikace, definuje zdroje pro jejich běh, tedy oblast datacentra, cenovou úroveň, počet a velikost instancí VM. Přičemž cenová úroveň určuje cenu plánu, ale také funkce App Service, kdy její změnou dochází ke škálování aplikace. Existují 4 kategorie úrovní, první *Shared compute* obsahuje dvě úrovně *Free* a *Shared* aplikace, jež běží na stejných VM jako ostatní a zdroje nelze je škálovat. Druhá kategorie *Dedicated compute* zahrnuje úrovně *Basic*, *Standard*, *Premium* a *PremiumV2* aplikace už mají dedikované VM a zdroje využívají jen mezi sebou. Předposlední je kategorie *Isolated*, kdy v tomto případě aplikace mají dedikované VM na dedikovaných virtuálních sítích. Poslední kategorií je *Consumption* pro tzv. *Function apps*, přičemž funkce se škálují dynamicky dle zátěže. (46)

Důležitým aspektem aplikací v Azure App Service je jejich horizontální a vertikální škálovatelnost, jež proběhne změnou cenové třídy App Service Plan a není potřeba změny žádného kódu či znovu nasazení aplikací. Případě horizontálního škálování se jedná o získání více jader procesoru, větší diskový prostor, dedikované VM, vlastní domény a certifikáty nebo více procesorových minut denně u sdílených procesorů či možnost automatického škálování a další. Horizontální škálování se týká počtu instancí VM, kdy lze mít až 20 instancí a v cenové kategorii *Isolated* je jich možné mít až 100. (47)

### 3.5 MVVM

Projekt *Finance Curator* byl vytvořen s pomocí architekturního vzoru MVVM, kdy soubory pro jednotlivé části byly rozděleny do složek s názvem pro danou část například třídy *Income* je ve složce *Models* atd. Vzor byl použit za účelem rozdělení aplikační logiky a UI, a tedy aplikace je snazší pro správu, testování a další vývoj. (48)

Samotný vzor se skládá ze 3 částí, tedy *Model-View-ViewModel* viz. Obrázek 14 a lze jej využít na všech XAML platformách, přičemž se jeho pravidla, respektive vztahy mezi jednotlivými vrstvami mohou lišit dle použité technologie.



Obrázek 14 - Ukázka částí vzoru MVVM

Zdroj: (48)

- **View**
  - Část *View* určuje vzhled a layout stránek či ovládacích prvků, případně obsahuje obslužné rutiny pro práci s UI. Obvykle se jedná o stránky aplikace jedná se například o *SettingsPage* nebo *AboutPage* viz. Kód 1, ale může se také jednat o vyskakovací okna nebo uživatelské ovládací prvky. (48)

```

1. <Page
2.     x:Class="FinanceCurator.Views.Pages.AboutPage"
3.     xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
4.     xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
5.     xmlns:local="using:FinanceCurator.Views.Pages"
6.     xmlns:uiModels="using:FinanceCurator.Models.UserInterface"
7.     xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
8.     xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
9.     mc:Ignorable="d"
10.    NavigationCacheMode="Enabled">
  
```

Kód 1 - Ukázka stránky *AboutPage*

Zdroj: vlastní

- **ViewModel**

- Tato část slouží jako vrstva mezi *View* a *Model* částmi, tedy poskytuje data z *Model* vrstvě *View*, jedná například o třídu *SortOptionViewModel* viz. Kód 2 nebo *ExpenseViewModel*, jež poskytuje data z databáze s výdaji. Tyto data jsou poskytovány formou kolekci, kdy lze využít *ObservableCollection*, která implementuje upozorňování na změny v dané kolekci. (48)

```
1. internal class SortOptionViewModel
2. {
3.     private static SortOptionViewModel _instance;
4.     private List<SortOption> _sortAttributes;
5.     private SortOption _sortOption;
6.     private List<SortOption> _sortOptions;
7.
8.     protected SortOptionViewModel()
9.     {
10.         _sortAttributes = new List<SortOption>();
11.     }
```

Kód 2 - Ukázka třídy *SortOptionViewModel*

Zdroj: vlastní

- **Model**

- Vrstva *Model* zahrnuje datový model s business nebo validační logikou, může se tedy jednat o DTO nebo POCO objekt. Příkladem třídy spadající do této části je třída *Income* nebo *SortOption* viz. Kód 3. (48)

```
1. internal class SortOption
2. {
3.     public string EntityAttribute { get; set; }
4.
5.     public string Option { get; set; }
6. }
```

Kód 3 - Ukázka třídy *SortOption*

Zdroj: vlastní

## 4. Osobní finance

Kapitola Osobní finance se zabývá tím, co je výkaz peněžních toků, dále rodinnými financemi, tedy příjmy a výdaji. Také jsou zde zahrnuty podkapitoly o finanční rozvaze nebo plánu.

### 4.1 Výkaz peněžních toků

Výkaz o peněžních tocích neboli výkaz o *cashflow* dává informace o určité položce rozvahy, tedy předmětem jsou informace o přírůstcích a úbytcích peněžních prostředků a peněžních ekvivalentů podle jednotlivých skupin činností podniku nebo osobního cashflow viz. Obrázek 15, přičemž v případě podniku existují 3 a v případě osob se jedná o analogii finančního výkazu peněžních toků. První skupinou činností podniku je provozní, tedy jde o základní činnost podniku. Druhou skupinou je investiční činnost, jež se zabývá pořízení nebo vyřízení dlouhodobého majetku. Poslední je finanční činnost, která souvisí se změnami velikosti nebo složení položek kapitálu. (49) (50) (51)

Příjmy		Výdaje	
Hrubá mzda otce	40 000 Kč	Daň z příjmu, SZP	13 650 Kč
Hrubá mzda matky	25 000 Kč	Domácnost, energie	12 000 Kč
Příjem z pronájmu	5000 Kč	Kapesné dětem	6 000 Kč
Ubytovací stipendium VŠ	700 Kč	Strava	10 000 Kč
....	??? Kč	Splátka hypotéky	9 600 Kč
		Doprava	7000 Kč
		Oblečení	2000 Kč
<b>Příjmy celkem</b>	<b>70 700 Kč</b>	<b>Výdaje celkem</b>	<b>60 250 Kč</b>
		<b>Čistý příjem</b>	<b>10 450 Kč</b>

Obrázek 15 - Ukázka osobního cashflow

Zdroj: (51)

Tento výkaz lze využít k zobrazení cashflow podniku za minulá období nebo ke strategickém a krátkodobém či dlouhodobém řízení podniku a lze jej sestavit přímou, kdy se zvolí skupiny příjmů a výdajů nebo nepřímou metodou, kdy je výsledek upraven o nepeněžní transakce, neuhrazené výdaje a zisky z minulých účetních období nebo o příjmy a výdaje spojené s finanční a investiční činností. (50)

## 4.2 Rodinné příjmy

Rodinné příjmy a osobní příjmy určují schopnost zvyšování čistého jmění, pokud je dobře investováno do ziskových aktiv, jedná se tedy o všechny finanční zdroje za určité období, které lze rozdělit do několika kategorií. (51) (52)

Rozdělení příjmů podle jejich původce:

- **Příjmy aktivní:** původcem je člověk, který se na nich podílí, například plat, mzda, odměny nebo bonusy atd.
- **Příjmy pasivní:** jsou nezávislé na aktivitě člověka, respektive nevyžadují soustavnou činnost, například výnosy z cenných papírů, pronájem bytu
- **Příjmy portfoliové:** jsou příjmy, jež pocházejí z investic či finančního majetku, například autorská práva, dividendy nebo dluhopisové kupony.

Rozdělení příjmů podle jejich pravidelnosti:

- **Pravidelné:** jsou příjmy, jejichž výše se tak často nemění a lze na ně spoléhat, například plat, pronájem bytu.
- **Nepravidelné:** jsou příjmy, na nichž nelze spoléhat a jejich výše kolísá, mezi ně patří kupříkladu výnosy z cenných papírů, odměny, úroky z vkladů atd.
- **Jednorázové:** jsou nahodilé a většinou se jedná o příjmy z prodeje majetku.

Zdroje: (51), (52)



### 4.3 Rodinné výdaje

Rodinné výdaje jsou další složkou, jež ovlivňuje čistý příjem, přičemž není značným faktorem kumulace rodinného bohatství. Jedná se tedy o utrácení financí za statky či služby pro zvýšení životní úrovně. Výdaje lze rozdělit podle jejich ovlivnitelnosti a nutnosti. (51) (52)

Rozdělení výdajů podle ovlivnitelnosti:

- **Fixní:** jsou výdaje, k jejichž změně osoba nemá pravomoci a nelze se jim vyhnout, jedná se například o paušální poplatky, leasingové splátky nebo nájemné.
- **Kontrolovatelné:** jsou výdaje, nad nimiž člověk může rozhodovat.

Rozdělení podle nutnosti:

- **Nezbytné:** výdaje, které jsou nezbytné, jako jídlo, léky či bydlení.
- **Zbytné:** jsou výdaje, které nejsou pro člověka nijak potřebné a mezi ně se řadí výdaje za zábavu, koníčky atd.
- **Investiční:** jsou výdaje, například nákup cenných papírů, po kterých člověk očekává následný příjem.

Zdroje: (51), (52), (53)

### 4.4 Finanční plán

Jedná se o plán, jež hodnotí současnou finanční situaci, stanovuje finanční cíle, posuzuje omezující okolnosti a ve výsledku stanovuje způsob dosažení daných cílů. Tento plán může být osobní, rodinný nebo podnikový. V případě podnikového se může nazývat podnikatelská rozvaha a určuje potřebu financí na nákup nezbytného vybavení při zahájení podnikání, ale také ověřuje schopnost pokrytí nákladů a dosažení zisku. Jde tedy o specifikaci a odhad počátečních výdajů. Co se týče osob nebo rodin tak cílem tohoto plánu je navýšení čistého jmění, tedy rozdíl mezi aktivy (veřejný majetek) a pasivy (dluhy, úvěry a půjčky). (51) (54) (55)

Důležitým hlediskem finančního plánu je určení priorit. Pro určení lze využít období SWOT analýzy, kdy jde o proces porovnání silných a slabých aspektů přání i případných příležitostí či hrozeb. V případě podniku je důležité, aby při sestavování plánu byl v shodě s ostatními částmi plánu, musí je tedy podporovat a být jimi podporovaný. (51) (55)

Proces sestavení finančního plánu se sestává z pěti částí viz. Obrázek 16. První je vyhodnocení finančních zdrojů, což zahrnuje zkoumání výdajů a zvyklostí. Následně je definice cílů, tato část je o zvážení a konzultování daných cílů. Po jejich zvolení je volba vhodných produktů. Po všech těchto částech následuje fáze realizace finančního plánu, která obsahuje zařizování, ověřování a podepisování smluv umožňující danou realizaci. Poté je důležité monitorovat průběh a sledovat změny tržních podmínek, také je vhodné aktualizovat daný plán, pro zabránění vzniku nedostatku financí. (51)



Obrázek 16 - Proces sestavení finančního plánu

Zdroj: (51)

## 4.5 Finanční rozvaha

Jedná se o výkaz do určitého data (tzv. rozvahový den), jež poskytuje souhrn majetku podniku, tedy o jeho aktivech a zdrojích jeho krytí, což jsou pasiva podniku. Cílem tohoto výkazu je zjištění druhů majetků a jejich složení nebo poměr vlastního a cizího kapitálu. Rozvaha je jedním ze základních výkazů účetní závěrky, přičemž účetní standardy určují obsah, rozsah či formu rozvahy a podnikatel by ji měl sestavovat v ideálním případě několikrát ročně. Tento výkaz také musí splňovat princip bilanční rovnice (součet aktiv je roven součtu pasiv). (56) (57)

V případě sestavení plánované rozvahy je zapotřebí mít vypracovaný finanční plán a dopočítanou předpokládanou úroveň zásob, krátkodobých pohledávek a závazků, které lze vyvodit z finančních cílů. Následně je potřeba mít vypočítaný obrat jednotlivých částí provozního kapitálu za předchozí období. (56)

Co se týče osobní rozvahy, tak se jedná o analogii finanční rozvahy, kdy se vypracovává pro osobní účely viz. Obrázek 17 a neřídí se zákonnými náležitostmi, například přesné pořadí individuálních položek nebo rozřídění do kategorií. (51)

Aktiva		Pasiva	
Hotovost	7 000 Kč	Kreditní karty	15 000 Kč
Bankovní účty	98 536 Kč	Půjčka od rodičů	350 000 Kč
Byt (dům)	1 690 000 Kč	Spotřebitelská půjčka	46 000 Kč
Automobil	350 000 Kč	Hypoteční úvěr	1 300 000 Kč
Akciový fond	112 000 Kč	...	??? Kč
Penzijní připojištění	34 500 Kč	...	??? Kč
Vybavení domácnosti	160 000 Kč	Čisté jmění	750 000 Kč
<b>Aktiva celkem</b>	<b>2 836 000 Kč</b>	<b>Pasiva celkem</b>	<b>2 836 000 Kč</b>

Obrázek 17 - Ukázka osobní rozvahy

Zdroj: (51)

## 5. Technická dokumentace

Tento díl dokumentu se zabývá technickým zpracováním projektu, jako například přehled a popis zdrojových souborů a složek projektu nebo využití systémových API práce s procesy na pozadí či implantace synchronizace dat aplikace atd. Dále jsou uvedeny části, které s implementací synchronizace dat souvisí, což zahrnuje například konfiguraci cloudové služby a aplikace.

### 5.1 Zdrojové soubory a složky projektu

Pro vytvoření projektu bylo využito IDE Visual Studio Community 2017 od společnosti Microsoft, které je dostupné na OS Windows a macOS. Pro zpracování zdrojových souborů projektu je nutné dodržet požadavky, které se skládají z nároků na VS 2015 nebo 2017 a na vývoj UWA. Dále třeba mít nainstalovaný balíček Microsoft.Services.Store.Engagement ve verzi uvedené v kapitole Použité technologie, protože je třeba mít na něj správný odkaz v rámci hlavního projektu (stačí odebrat a přidat odkaz). (58)

OS	Windows 10
IDE	VS 2015 a novější
SDK	SDK pro Anniversary Update (sestavení 14393) a novější

**Tabulka 6 - Minimální požadavky pro zpracování zdrojových souborů projektu**

**Zdroj: (59)**

Pracovní frekvence procesoru	1,6 GHz
Velikost operační paměti	1 – 1,5 GB
Velikost interní paměti	600 MB – 10 GB
Verze DirectX	9
Rozlišení monitoru	1024x768

**Tabulka 7 - Minimální požadavky na VS 2015**

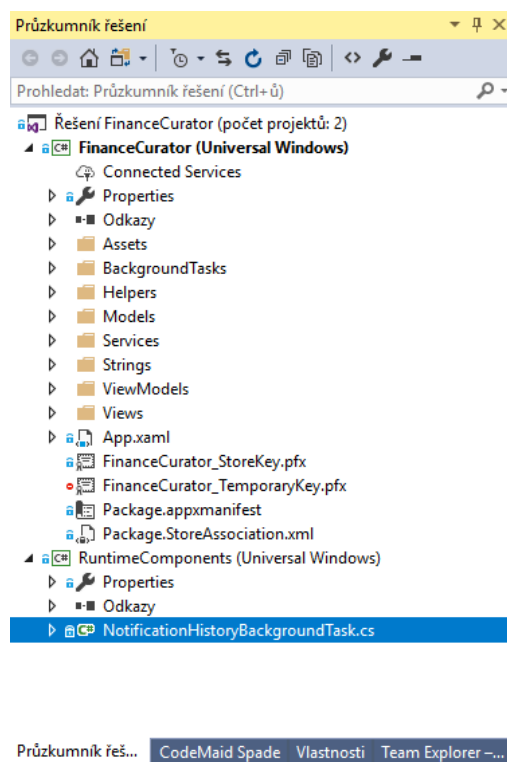
**Zdroj: (60)**

Pracovní frekvence procesoru	1,8 GHz
Velikost operační paměti	2 – 2,5 GB
Velikost interní paměti	20 – 130 GB
Rozlišení monitoru	1280x720

**Tabulka 8 - Minimální požadavky na VS 2017**

**Zdroj: (61)**

Po otevření souboru s řešením projektu (soubor s příponou .sln) ve VS 2015 nebo 2017 se v průzkumníku řešení objeví dva projekty, první s názvem *FinanceCurator* je samotná aplikace a druhý s označením *RuntimeComponents*, což je runtime komponenta. Každý tento projekt obsahuje zdrojové soubory pro aplikaci, v případě druhého projektu se jedná o soubor s procesem na pozadí (*NotificationHistoryBackgroundTask.cs*), který řeší historii notifikací. Ovšem první projekt je rozčleněn do několika složek viz. Obrázek 18, které jsou pojmenovány podle jednotlivých částí architekturního vzorce MVVM s výjimkou složek *Assets*, *Helpers* a *BackgroundTasks*. Mimo těchto složek je zde i manifest aplikační balíčky, XML soubor, který vznikl propojením s Microsoft Store a také vstupní bod pro aplikaci (*App.xaml* a *App.xaml.cs*).



Obrázek 18 - Přehled projektů a složek v rámci řešení

Zdroj: vlastní

Složky v rámci projektu *FinanceCurator* a jejich obsah:

- **Views**
  - Tato složka obsahuje tři podsložky s názvy *Dialogs*, *Pages* a *UserControls*, které jsou pro jednotlivé stránky, ovládací prvky a dialogová okna.
  - Do této složky patří například soubory: *DeleteDialog.xaml* s *DeleteDialog.xaml.cs* nebo *SettingsPage.xaml*

```
1. public sealed partial class DeleteDialog : ContentDialog
2. {
3.     /// <summary>
4.     /// Constructor for confirming deleting.
5.     /// </summary>
6.     public DeleteDialog()
7.     {
8.         InitializeComponent();
9.         PrimaryButtonText = ResourceLoaderHelper.GetResourceLoader().GetString("Confirm");
10.        SecondaryButtonText = ResourceLoaderHelper.GetResourceLoader().GetString("Cancel");
11.        Title = ResourceLoaderHelper.GetResourceLoader().GetString("DeleteDialogTitle");
12.    }
```

Kód 4 - Ukázka třídy *DeleteDialog*

Zdroj: vlastní

- **ViewModels**

- V dané složce se nachází dvě podsložky, první je *DatabaseViewModels* a druhou je *UserInterface*. Tyto složky obsahují soubory s třídami, které spadají do modelu MVVM a v tomto případě se jedná o *ViewModel* třídy. Mezi tyto soubory patří *CategoryViewModel.cs*, *DebtViewModel.cs* nebo *SortOptionViewModel.cs* a další.

```
1. public List<License> GetAppLicenses()
2. {
3.     _licenses = new List<License>()
4.     {
5.         new License()
6.         {
7.             Name = "Microsoft.NETCore.UniversalWindowsPlatform",
8.             Description = $"Copyright (c) 2015 .NET Foundation. All rights reserved.\nLicensed under the MIT License. See:",
9.             LicenceURL = "https://github.com/dotnet/core-setup/blob/master/LICENSE.TXT"
10.        }
11.    }
```

Kód 5 - Ukázka třídy *LicenseViewModel*

Zdroj: vlastní

- **Models**

- Je složka, jež obsahuje podsložky se soubory pro data aplikace, jako například enumerační třídy ve složce *Enums* nebo soubor s třídou *License.cs* reprezentující zobrazenou licenci. Ovšem nejdůležitější částí je složka *DatabaseModels*, která obsahuje třídy, podle nichž se vytvářelo lokální uložení, a reprezentují tedy jednotlivé tabulky. V dané třídě se také nachází podsložka *JoinTables*, jež obsahuje soubory s třídami se stejným účelem jako předchozí, ale v tomto případě se jedná o vazební tabulky.

```
1. public class EventAttendee
2. {
3.     [JsonProperty(PropertyName = "id")]
4.     public string Id { get; set; }
5.
6.     [JsonProperty(PropertyName = "name")]
7.     public string Name { get; set; }
8.
9.     [JsonProperty(PropertyName = "owed")]
10.    public double Owed { get; set; }
11.
12.    [JsonProperty(PropertyName = "paid")]
13.    public double Paid { get; set; }
```

**Kód 6 - Ukázka třídy *EventAttendee*****Zdroj: vlastní**

- **Services**

- Tento adresář obsahuje čtyři soubory, které obsahují kód pro obecnou funkcionalitu v architektuře MVVM, což například zahrnuje posílání notifikací nebo zobrazování dialogových oken atd.
- Obsažené soubory: *DialogService.cs*, *ExportService.cs*, *NotificationService.cs* a *AzureService.cs*.

```
1. private void SaveUserData(string sid, string token, string userName)
2. {
3.     PasswordVaultHelper.Instance().AddPasswordCredential($"{MobileServiceAuthenticationP
4.     rovider.MicrosoftAccount}", sid, token);
5.     SettingsHelper.Instance().SaveAsLocal("UserName", userName);
6.     SettingsHelper.Instance().SaveAsLocal("LastLogin", $"{DateTime.Now}");
7. }
```

**Kód 7 - Ukázka třídy *AzureService*****Zdroj: vlastní**

- **Helpers**

- V této složce se nalézají 7 souborů *ResourceLoaderHelper.cs* a *StorageFolderHelper.cs*, které obsahují kód pro jednoduchou funkcionalitu v rámci aplikace, jako například ukládání instance třídy *StorageFolder*.

```
• public static ResourceLoader GetResourceLoader()
• {
•     return ResourceLoader.GetForViewIndependentUse("Resources");
• }
```

**Kód 8 - Ukázka třídy *ResourceLoaderHelper*****Zdroj: vlastní**

- **BackgroundTasks**

- V rámci této složky jsou čtyři soubory s kódy pro procesy na pozadí, například pro posílání notifikací nebo pro synchronizaci dat.
- Obsažené soubory: *NotificationBackgroundTask.cs*, *SyncBackgroundTask.cs* a další.

```
1.  /// <summary>
2.  /// Background task for background data synchronization.
3.  /// </summary>
4.  internal class SyncBackgroundTask : IBackgroundTask
5.  {
6.      public async void Run(IBackgroundTaskInstance taskInstance)
7.      {
8.          bool backgroundSyncSetting = (bool)ApplicationData.Current.LocalSettings.Values["backgroundSyncSetting"];
9.          if (backgroundSyncSetting)
10.         {
11.             await AzureService.Instance().SyncAsync();
12.         }
13.     }
14. }
```

**Kód 9 - Ukázka třídy *SyncBackgroundTask***

**Zdroj: vlastní**

- **Strings**

- Tato složka má dvě podsložky s názvy *en-US* a *cs-CZ*, přičemž každá z nich obsahuje jeden soubor s příponou *.resw*, jež obsahují textové řetězce v angličtině nebo češtině a jsou zobrazovány v aplikaci, podle nastaveného jazyka operačního systému.

- **Assets**

- Obsahuje veškeré obrázky, které aplikace využívá. Jedná se tedy o obrázky pro malé, střední, široké a velké dlaždice společně s jejich škálovanými verzemi, ale jsou zde i pro seznam aplikací v rámci OS, přičemž jsou rozřazeny do složek jako *Medium* podle jejich účelu. V této složce se nacházejí i obrázky, které nejsou rozřazené a ty jsou pro normální využití v rámci aplikace většinou za jejího běhu.

Soubory projektu *RuntimeComponents*:

- V tomto projektu se kromě systémových složek a souborů nachází jeden soubor s názvem *NotificationHistoryBackgroundTask.cs*, který obsahuje třídu pro proces na pozadí, který reaguje na změny v historii notifikací aplikace.



```
1.    /// <summary>
2.    /// Background task triggered by changes in app notification history.
3.    /// </summary>
4.    public sealed class NotificationHistoryBackgroundTask : IBackgroundTask
5.    {
6.        private ResourceLoader ResourceLoaderHelper
7.        {
8.            get
9.            {
10.                return ResourceLoader.GetForViewIndependentUse("Resources");
11.            }
12.        }
13.
14.        public void Run(IBackgroundTaskInstance taskInstance)
15.        {
16.            ToastNotificationHistoryChangedTriggerDetail details = (ToastNotificationHis
                toryChangedTriggerDetail)taskInstance.TriggerDetails;
```

**Kód 10 - Ukázka třídy *NotificationHistoryBackgroundTask***

**Zdroj: vlastní**

## 5.2 Manifest aplikačního balíčku

Pro nasazení, zobrazení či aktualizování aplikace potřebuje operační systém XML dokument zvaný manifest aplikačního balíčku. Tento manifest má v průběhu vývoje dvě varianty, první je dočasná a není generována VS, přičemž obsahuje vývojářem vytvořenou konfiguraci aplikace, kdy se využívá během vývoje aplikace. Soubor první varianty se jmenuje *Package.appxmanifest*. Druhá obdoba je souboru vygenerovaný VS a je sestaven z informací obsažených *Package.appxmanifest* a jedná se tedy o finální verzi, která je součástí samotného aplikačního balíčku, přičemž je digitálně podepsaná a nelze pak daný dokument upravovat bez porušení podpisu balíčku. (62) (63)

Tento manifest obsahuje informace o vlastnostech aplikace, jejím vydavateli, ale také o jejím vstupním bodu, závislostech na různých API, seznam podporovaných jazyků nebo registraci komponent Windows Runtime či seznam možností aplikace viz. Kód 11. Informace o vlastnostech aplikace zahrnují údaje o jménu aplikace a balíčku včetně loga, verze nebo podporovaných rotací aplikace či podporované architektury procesorů. V případě informací o vydavateli se jedná o jeho zobrazené jméno a jeho *common name* (CN). Dále možnosti aplikace jsou pro určení přístupu aplikace k určitým zdrojům, API nebo komponentám zařízení mezi ně patří například přístup ke knihovně obrázků uživatele nebo kontaktům a událostem v kalendáři. (62) (64)

```
1.      <uap:InitialRotationPreference>
2.      <uap:Rotation Preference="portrait" />
3.      <uap:Rotation Preference="landscape" />
4.      <uap:Rotation Preference="portraitFlipped" />
5.      <uap:Rotation Preference="landscapeFlipped" />
6.      </uap:InitialRotationPreference>
7.      </uap:VisualElements>
8.      <Extensions>
9.      <uap:Extension Category="windows.protocol">
10.         <uap:Protocol Name="financemanager">
11.             <uap:DisplayName>Finance Curator</uap:DisplayName>
12.         </uap:Protocol>
13.     </uap:Extension>
14.     <Extension Category="windows.backgroundTasks" EntryPoint="RuntimeComponents.Notifi
ficationHistoryBackgroundTask">
15.         <BackgroundTasks>
16.             <Task Type="pushNotification" />
17.         </BackgroundTasks>
18.     </Extension>
```

Kód 11 - Ukázka manifestu aplikačního balíčku

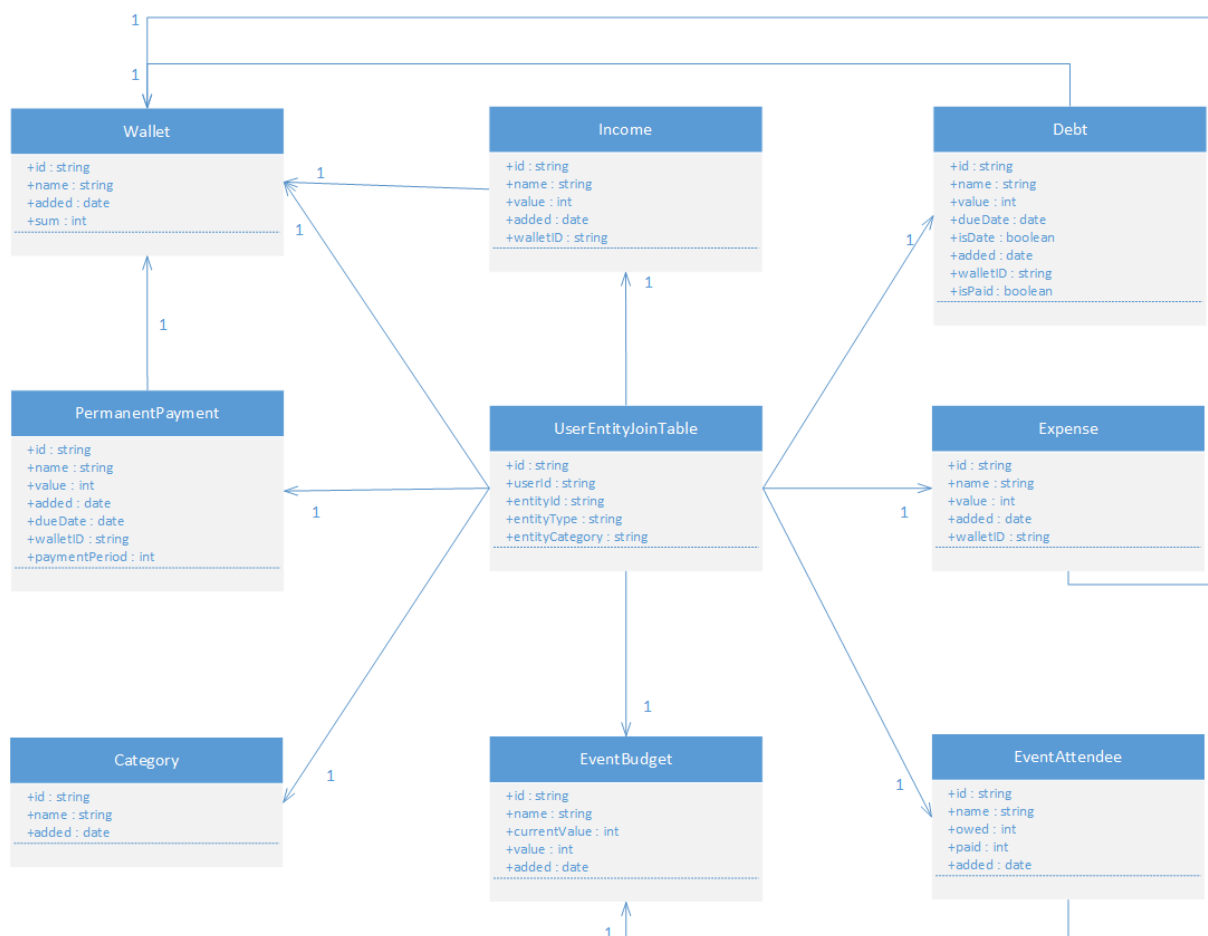
Zdroj: vlastní

### 5.3 Databáze

V rámci této kapitoly je popsáno lokální uložení aplikace, což zahrnuje popis knihovny SQLite a diagram struktury lokální databáze, následně je popsáno serverové uložení, společně s diagramem tabulek.

Aplikace pro lokální ukládání dat využívá uložení založené na SQLite, tedy softwarová knihovna, jež poskytuje celý systém řízení báze dat. Výhodou jsou minimální požadavky na zdroje zařízení, administraci a konfiguraci databáze nebo transakčnost, kdy veškeré transakce jsou atomické, consistentní, izolované a odolné (ACID). Dalším kladem je absence serveru, kdy SQLite nepotřebuje pro svůj běh server a databáze je tedy součástí aplikace z čehož plyne, že není třeba konfigurovat žádný serverový proces. (65) (66) (67)

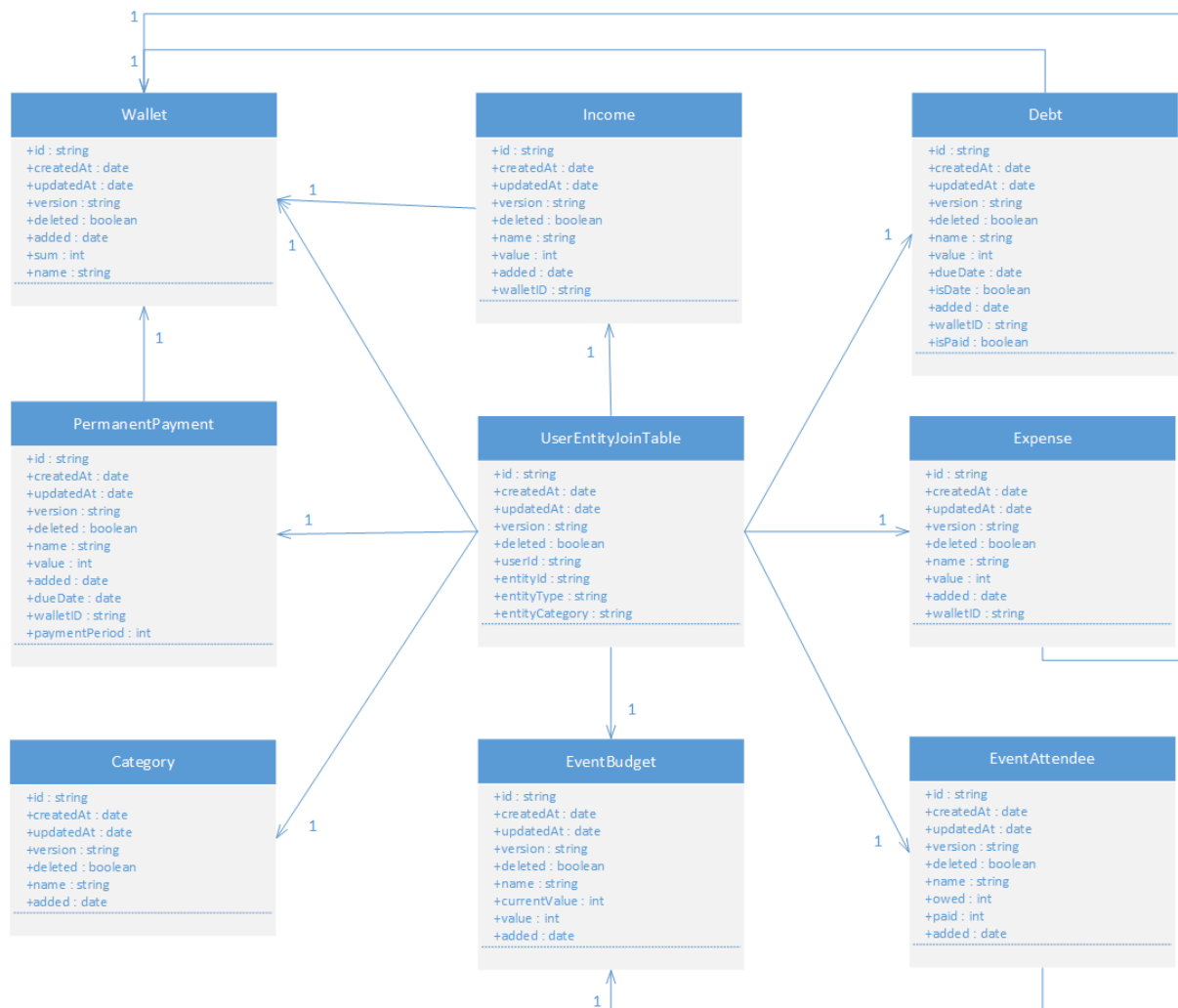
Přičemž lokální databáze se skládá celkem z 9 tabulek viz. Obrázek 19 a 3 další pro systémové účely, jako tabulka se systémovými vlastnostmi nebo s chybami. Ovšem pro aplikaci je důležitá tabulka *UserEntityJoinTable*, která obsahuje záznamy pro spojení uživatele SID s jeho daty a jejich rozřazení do kategorií. Další tabulky jsou pro záznamy financí, kategorií nebo cestovních rozpočtů a účastníků.



Obrázek 19 - Diagram tabulek v lokální databázi

Zdroj: vlastní

Databáze na SQL serveru obsahuje také 9 tabulek viz. Obrázek 20, ovšem jsou doplněny o sloupce pro hodnoty, které jsou využívány službou Azure App Service pro řešení konfliktů záznamů při synchronizaci dat nebo sloupec pro tzv. *soft delete* nebo například sloupec *version* nebo *createdAt* a další. Dalším rozdílem je, že v této databázi nejsou 3 systémové tabulky, jako tomu bylo u lokální.



Obrázek 20 - Diagram tabulek v databázi na serveru

Zdroj: vlastní

Třídy, podle kterých se vytvořily tabulky jsou v podsložce *DatabaseModels* ve složce *Models* a slouží pro objektové zpracování dat z relační databáze. Pro práci s těmito daty slouží třídy v rámci složky *DatabaseViewModels* ve složce *ViewModels*.

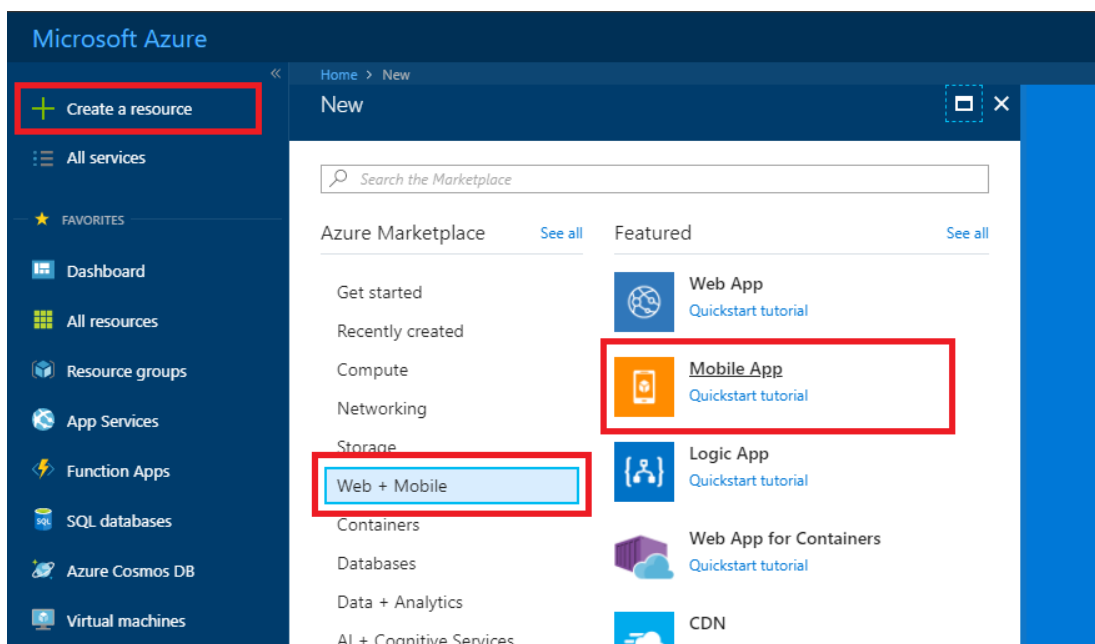
## 5.4 Azure App Service

V této kapitole je popsána konfigurace služby Azure App Service a konfigurace aplikace, aby se serverovým backend. Dále se zde nachází popis principu off-line využívání aplikace a synchronizace uživatelských dat.

### 5.4.1 Konfigurace služby

Tato část kapitoly Azure Mobile Apps se věnuje vytvoření a nastavení vlastní Azure App Service, plánu služby App Service a SQL Server s SQL databází na cloudové platformě Microsoft Azure. Ovšem lze také využít předem nakonfigurovanou službu, jež má URL adresu *https://financecurator.azurewebsites.net* a samotná aplikace *Finance Curator* je nastavena, aby jí využívala.

Pro nastavení služby je zapotřebí mít *Microsoft Azure Subscription* a poté se lze přihlásit do webového portálu Azure, kdy se v levém horním rohu stránky nachází tlačítko s popisem *Přidat zdroj* a po jeho stisknutí se zobrazí seznam kategorií služeb Azure posléze lze vyhledat nebo najít v kategorii *Web+Mobile* službu *Mobile app* viz. Obrázek 21. (68)

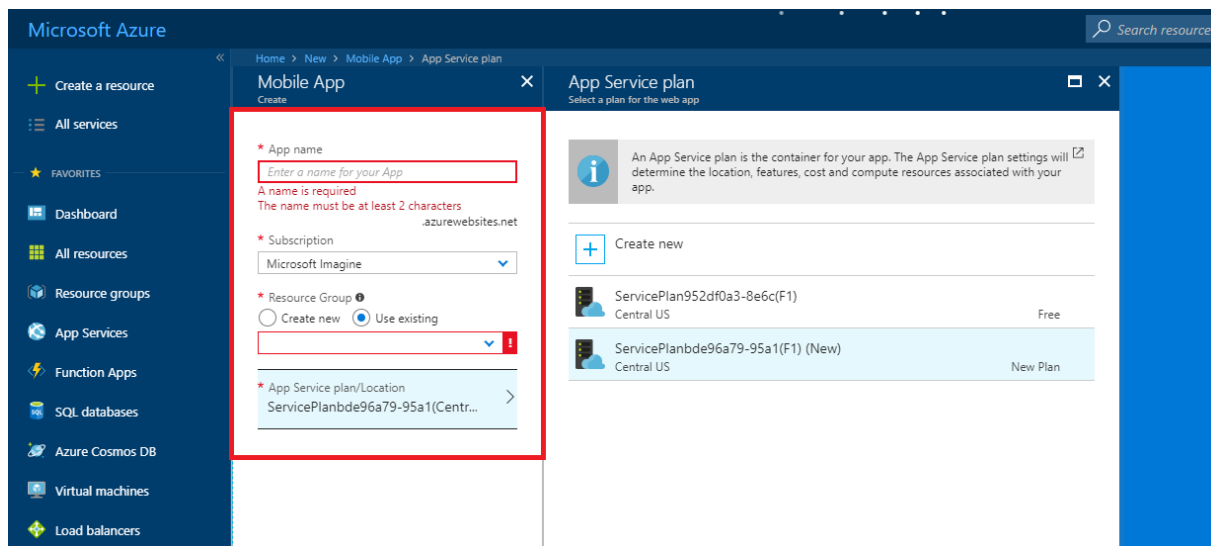


Obrázek 21 - Vytvoření zdroje Mobile Apps v portálu Azure

**Zdroj: vlastní**

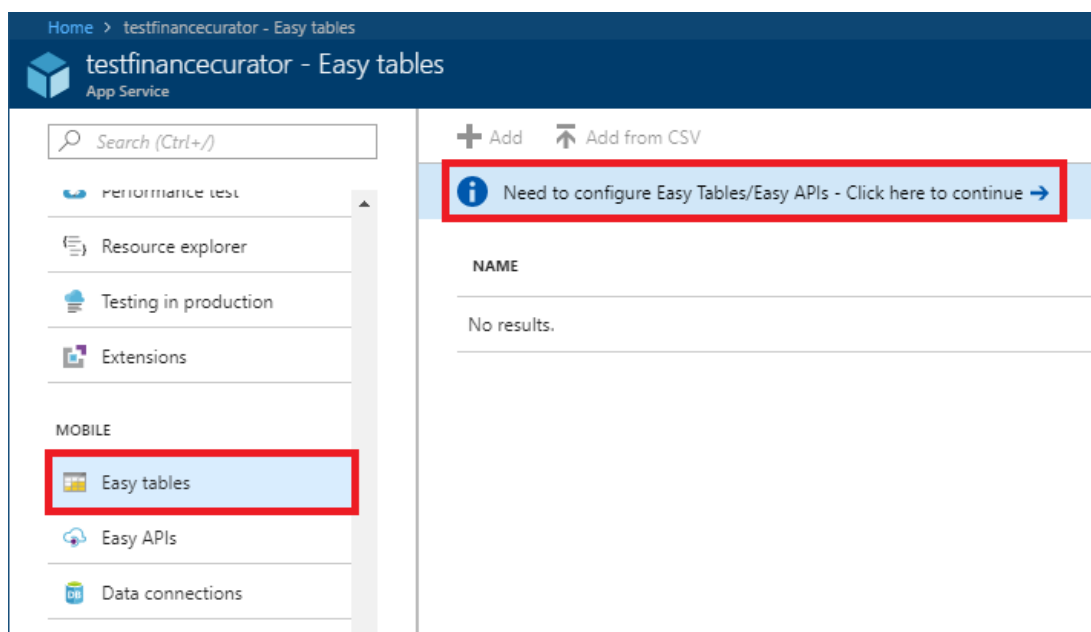
Po kliknutí na danou službu se zobrazí tzv. blade viz. Obrázek 22, jež obsahuje vstupní pole pro název aplikace a zároveň název subdomény. Následně je zde i ovládací prvek pro vybrání předplatného Azure, pak pro vybrání skupiny zdrojů a pro zvolení servisního plánu, přičemž je potřeba tyto vstupní pole vyplnit a teprve poté se služba nasadí a spustí. (68)

Samozřejmě pro aplikaci je důležité využít databázi, a tedy v sekci *Jednoduché tabulky* se lze připojit k databázi po kliknutí na odkaz pro konfiguraci viz. Obrázek 23. Pro tyto tabulky je potřeba vytvořit databázi přes datové připojení, kde se určí SQL databáze a připojovací řetězec viz. Obrázek 24 samozřejmě je nutné vytvořit SQL server viz. Obrázek 25. (68)



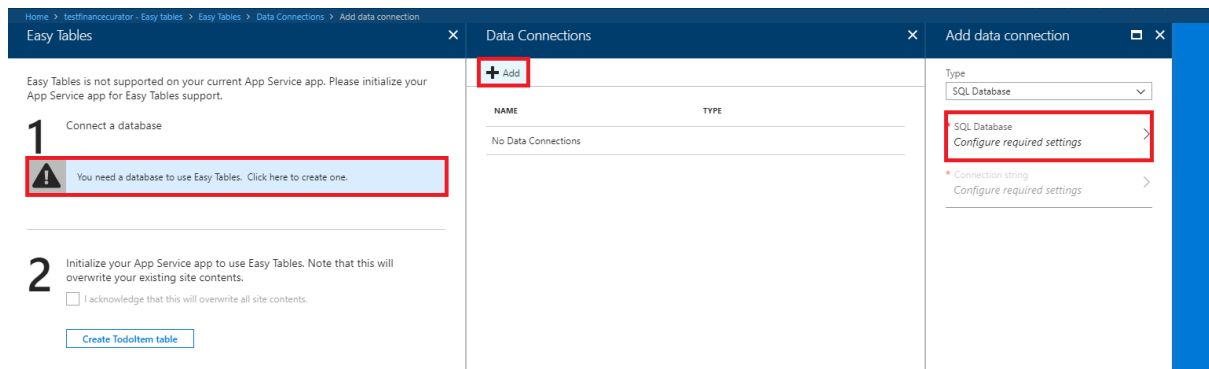
Obrázek 22 - Blade pro vytvoření služby Mobile App

Zdroj: vlastní



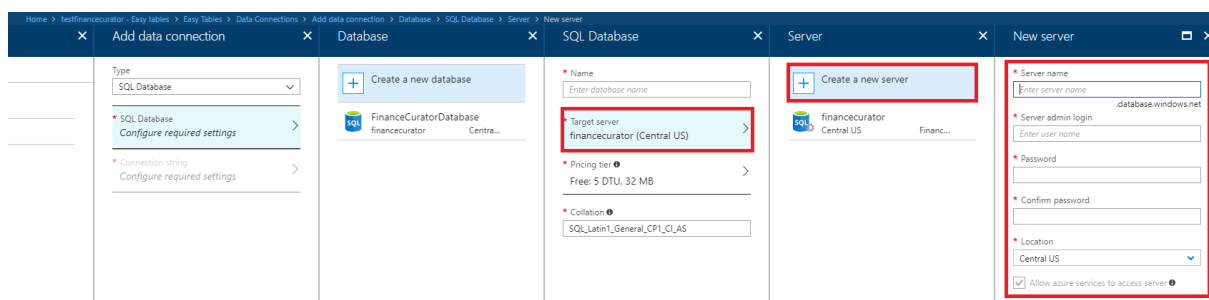
Obrázek 23 - Odkazy na tabulky a jejich konfiguraci

Zdroj: vlastní



Obrázek 24 - Nastavení datového připojení

Zdroj: vlastní



Obrázek 25 - Vytvoření SQL serveru

Zdroj: vlastní

Poté co je datové připojení vytvořeno, tak je třeba vytvořit tabulky podle schématu viz. Obrázek 20, přičemž tabulky se musí jmenovat stejně jako třídy v aplikaci.

Další částí konfigurace služby je nastavení autentizace uživatele, které se nachází v sekci *Autentizace / Autorizace* na stránce s cloudovou službou, ale před začátkem samotné konfigurace je třeba na stránce *My applications* v Microsoft Account Developer Center (je třeba vlastnit vývojářskou licenci) mít vytvořenou aplikaci, přičemž po zobrazení informací o aplikaci by tam měla být sekce s názvem *Platforms*, kde je vstupní pole pro URL adresy pro přesměrování viz. Obrázek 26 a tam má být hodnota *[URL adresa služby]/.auth/login/microsoftaccount/callback*. (69)

Na stránce služby v Azure Portal s nastavením autentizace je tlačítko pro její zapnutí viz. Obrázek 27 a po jeho stisknutí se zobrazí ovládací prvek pro nastavení akce, pokud žádosti na službu nebudou ověřené a zde je potřeba vybrat přihlášení přes Microsoft Account. Pod tímto prvkem se nachází seznam s poskytovateli identit, přičemž je třeba nastavit autentizaci přes Microsoft Account, kdy po kliknutí na danou položku v seznamu se zobrazí dvě vstupní pole pro *ClientId* a pro *ClientSecret* a do prvního patří hodnota *Application Id*, jež je uvedena v Microsoft Account Developer Center viz. Obrázek 26 a do druhého *Application Secret* a také je tato hodnota uvedena ve vývojářském centru. Pod těmito vstupy se nachází seznam s oprávněními aplikace a stačí vybrat první tři položky daného seznamu. Dále na předchozí

stránce s konfigurací autentizace se nachází další vstupní pole pro povolené URL adresy pro přesměrování a do tohoto pole patří hodnota *financemanager://easyauth.callback*.

Name

Finance Curator

Application Id

Application Secrets

Generate New Password

Version 0 Current

Platforms

Add Platform

Web

Delete

☒ Allow Implicit Flow

☐ Restrict token issuing to this app

Limits the issuing of JSON Web Tokens (JWT) for your domain to exclusively this application.

Target Domain

This is the domain that other apps will use when they request a JWT for your app on Windows (such as www.contoso.com).

Target Domain

Redirect URLs ⓘ Add URL

https://financemanager.azurewebsites.net/.auth/login/microsoftaccount/callback

Obrázek 26 - Aplikace v Microsoft Account Dev Center

Zdroj: vlastní

FinanceCurator - Authentication / Authorization

App Service

Search (Ctrl+F)

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

DEPLOYMENT

Quickstart

Deployment credentials

Deployment slots

Deployment options

Continuous Delivery (Preview)

SETTINGS

Application settings

Authentication / Authorization

Managed service identity

Backups

Custom domains

SSL certificates

Networking

Scale up (App Service plan)

Scale out (App Service plan)

Save Discard

Authentication / Authorization

To enable Authentication / Authorization please ensure all your custom domains have corresponding SSL bindings .net version is configured to "4.5" and manage pipeline mode is set to "Integrated"

App Service Authentication

Off On

Action to take when request is not authenticated

Log in with Microsoft Account

Authentication Providers

Azure Active Directory

Not Configured

Facebook

Not Configured

Google

Not Configured

Twitter

Not Configured

Microsoft

Configured

Advanced Settings

Token Store

Off On

ALLOWED EXTERNAL REDIRECT URLS

financemanager://easyauth.callback

Obrázek 27 - Stránka s konfigurací autentizace uživatele

Zdroj: vlastní



## 5.4.2 Konfigurace aplikace

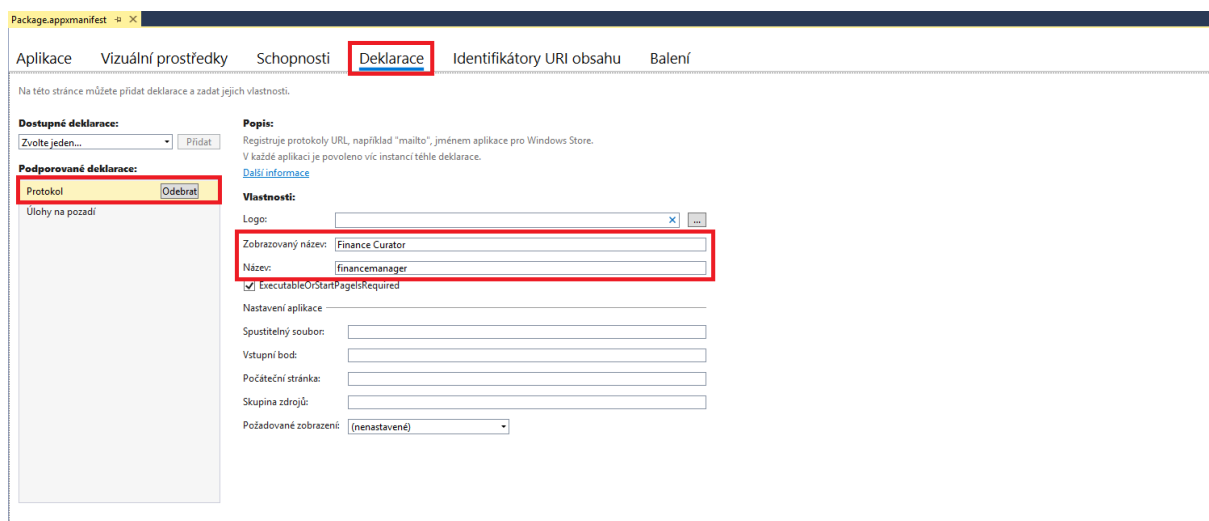
Po správném nastavení cloudové služby je třeba upravit aplikaci tak, aby danou službu využívala. Aplikace *Finance Curator* má ve své třídě *App.xaml.cs* atribut *ServiceClient* viz. Kód 12, který využívá třídu *MobileServiceClient* jako referenční datový typ, přičemž konstruktor této třídy je parametr *mobileAppUri*, což má být URL adresa služby, jako například *https://testfinancecurator.azurewebsites.net*.

```
1. sealed partial class App : Application
2. {
3.     public static string CurrencySymbol;
4.     public static MobileServiceClient ServiceClient = new MobileServiceClient("https://financecurator.azurewebsites.net");
5.
6.     public App()
7.     {
8.         InitializeComponent();
9.         Suspending += OnSuspending;
10.    }
```

Kód 12 - Ukázka třídy *App*

Zdroj: vlastní

Následujícím dílem konfigurace aplikace je její nastavení pro umožnění autentizace uživatele, kdy je potřeba upravit deklarace manifestu aplikačního balíčku, přičemž je zapotřebí přidat deklaraci *Protokol* do podporovaných deklarací viz. Obrázek 28. Poté se musí nastavit název podporovaného protokolu, což je hodnota před *//* v rámci URL adresy pro přesměrování, která byla nastavena pro cloudovou službu, jedná se například o *financemanager* v rámci *financemanager://easyauth.callback*.



Obrázek 28 - Nastavení deklarace *Protokol* v manifestu

Zdroj: vlastní

### 5.4.3 Off-line používání a synchronizace

Nezbytnou částí aplikace je možnost jejího využití bez připojení k počítačové síti a synchronizace lokálních dat aplikace se serverovou databází. Tyto funkce poskytuje služba Mobile Apps v rámci služby App Service na cloudové platformě Microsoft Azure. (70)

Off-line použití znamená provádět CRUD operace na datech v lokálním uložšti, které je založené na SQLite, ovšem lze implementovat vlastní lokální uložště, jež není založené na SQLite, ale třeba na SQLCipher. A poté co se zařízení připojí k počítačové síti lze veškeré lokální změny synchronizovat se serverovým backend. Pro udržování daných změn je použit tzv. synchronizační kontext, což je tabulka s CUD operacemi, které jsou při *Push* operaci posílány na server. (70)

Synchronizace se sestává z *Push* a *Pull* operací, kdy veškeré změny jsou uloženy v synchronizačním kontextu a nejsou poslané na server, dokud se neprovede *Push* viz. Kód 13, kdy se jedná o řadu REST volání na serverový backend. Naopak *Pull* operace se provádí pro jednotlivé tabulky a slouží pro získání dat ze serveru a jejich následné uložení do lokální databáze. (70)

```
1. private async Task PushAsync()  
2. {  
3.     await App.ServiceClient.SyncContext.PushAsync();  
4. }
```

**Kód 13 - Ukázka metody pro Push operaci**

**Zdroj: vlastní**

## 5.5 Použité technologie

V této části technické dokumentace jsou uvedeny všechny knihovny, které jsou použity v aplikaci včetně jejich informací. Dále jsou uvedené rozšíření do IDE, které byly využity při vývoji a ovlivnily kód aplikace nebo adresářovou strukturu projektu.

### 5.5.1 NuGet balíčky

- **Microsoft.NETCore.UniversalWindowsPlatform**
  - Autor: Microsoft
  - Licence: MIT
  - Verze: 6.0.7
  - Datum vydání verze: 12. 2. 2018
- **Azure Client SDK**
  - **Microsoft.Azure.Mobile.Client**
    - Autor: Microsoft
    - Licence: MIT
    - Verze: 4.0.2
    - Datum vydání verze:
  - **Microsoft.Azure.Mobile.Client.SQLiteStore**
    - Autor: Microsoft
    - Licence: MIT
    - Verze: 4.0.2
    - Datum vydání verze: 12. 10. 2017
- **UWP Community Toolkit**
  - Autor: kolektiv
  - Licence: MIT
  - Verze: 2.1.1
  - Datum vydání verze: 15. 12. 2017
- **Telerik UI for UWP**
  - Autor: Telerik AD
  - Licence: Apache 2.0
  - Verze: 10.0.0.9
  - Datum vydání verze: 17. 1. 2018

- **Newtonsoft.Json**
  - Autor: James Newton-King
  - Licence: MIT
  - Verze: 11.0.1
  - Datum vydání verze: 17. 2. 2018
- **FluentDateTime**
  - Autor: Slobodan Pavkov, Tom Pester, Simon Cropp
  - Licence: Apache 2.0
  - Verze: 1.14.0
  - Datum vydání verze: 27. 8. 2017
- **Microsoft.Services.Store.Engagement**
  - Autor: Microsoft
  - Licence: Microsoft Terms of Use
  - Verze: MIT
  - Datum vydání verze: 30. 11. 2017

### 5.5.2 Rozšíření do IDE

- **CodeMaid**
  - Autor: Steve Cadwallader
  - Verze: 10.4.53
  - Licence: GNU Lesser General Public License v3.0
- **UWP Visual Assets Generator**
  - Autor: Peter Rill
  - Verze: 1.6

## 5.6 Procesy na pozadí

Aplikace *Finance Curator* využívá pro některé své funkce procesy na pozadí, jako například synchronizace dat, posílání notifikací ohledně dat splatnosti dluhů a trvalých plateb, nebo také pro znovu poslání určité notifikace. Kód pro dané procesy se nachází ve složce *BackgroundTasks* v hlavním projektu a soubor pro jeden konkrétní proces je obsažen v projektu *RuntimeComponents*, přičemž jejich registrace a spouštěče se nachází v souboru *App.xaml.cs* v rámci metody *RegisterBackgroundTasks()* (Příloha 1).

Tyto procesy mohou být tzv. *In-process* nebo *Out-of-process*, kdy *In-process* běží ve stejném procesu jako samotná aplikace a *Out-of-process* běží v rámci odděleného procesu s názvem *backgroundtaskhost.exe*. Dalším rozdílem mezi těmito typy je, že procesy, které běží ve stejném procesu jako aplikace, tak mohou shodit aplikaci v případě chyby, ale nejsou složité a není potřeba řešit komunikaci mezi procesy či vytvářet komponenty *Windows Runtime* a následně je registrovat v manifestu aplikačního balíčku. V aplikaci *Finance Curator* jsou používány pouze *Out-of-process* z důvodů požadavků na verzi univerzální API, kdy podpora pro *In-process* procesy byla přidána až ve verzi 1607 a pro funkcionality této aplikace je není třeba implementovat. (71) (72)

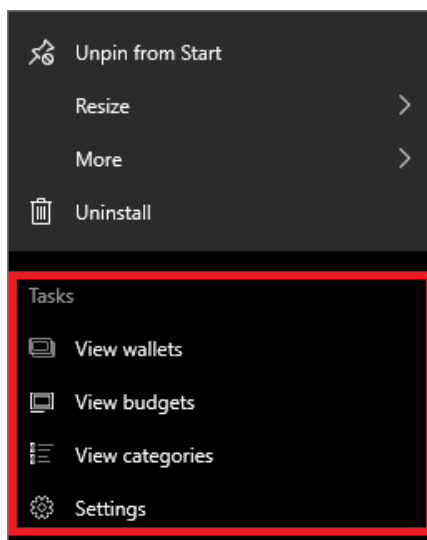
Aplikací implementované procesy mají různé spouštěče a podmínky jejich spuštění, kdy proces pro zachycení a zpracování dat z interaktivní notifikace (Příloha 2) se spustí, pokud uživatel provede akci s danou notifikací. Následně je proces pro posílání interaktivní notifikace (Příloha 3) je spuštěn v případě, že došlo ke změně v historii notifikací aplikace. Dalším je proces na pozadí, jež slouží pro kontrolu dat splatnosti dluhů a případné posílání notifikací s upozorněním (Příloha 4), tento proces je opakován přibližně každých 60 minut. Aplikace také využívá proces pro synchronizaci dat na pozadí (Příloha 5), který je opakován každých 15 minut (minimální systémem povolený čas) a za podmínky, že zařízení má připojení k počítačové síti. Poslední je proces pro kontrolu dat splatnosti trvalých plateb s možným posláním notifikací s upozorněním (Příloha 6), který je opakován každých 24 hodin. (73)

## 5.7 Systémové API

Aplikace pracuje se systémovými API jak univerzálními, tak i specifickými pro určitá zařízení, jako například JumpList, Appointments nebo Contact Manager API, jež poskytují funkce pro přidání zkratk k úlohám, UI pro vybrání kontaktu na zařízení či vytvoření události v kalendáři.

### 5.7.1 JumpList API

Toto API umožňuje přidat seznam zkratk viz. Obrázek 29 k vývojářem definovaným úlohám nebo k často používaným dokumentům, přičemž tento seznam je zobrazen ve vyskakovacím menu po kliknutí na ikonu aplikace v seznamu aplikací nebo na hlavním panelu. Tato API bylo přidáno v *November Update* (sestavení 10586) a není podporováno všemi zařízeními. (74)



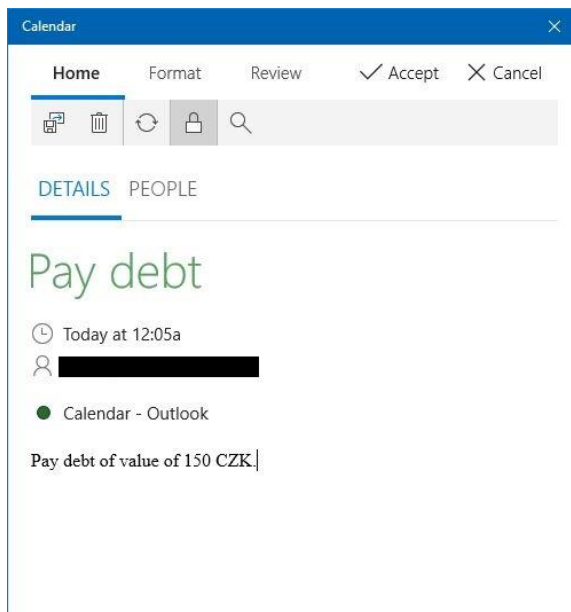
Obrázek 29 - JumpList aplikace *Finance Curator*

Zdroj: vlastní

### 5.7.2 Appointments API

*Appointments API* poskytuje funkce pro vytváření událostí a jejich správu v aplikaci kalendáře. Tato API je v aplikaci *Finance Curator* využito pro přidání dodatečného upozornění o datumu splatnosti dluhu nebo trvalé platby, aby se uživatel nespolehl pouze na aplikaci. Celý

proces začne zobrazením vyskakovacího okna viz. Obrázek 30 s ovládacími prvky a předvyplněnými vstupními poli pro přidání upozornění. (75)

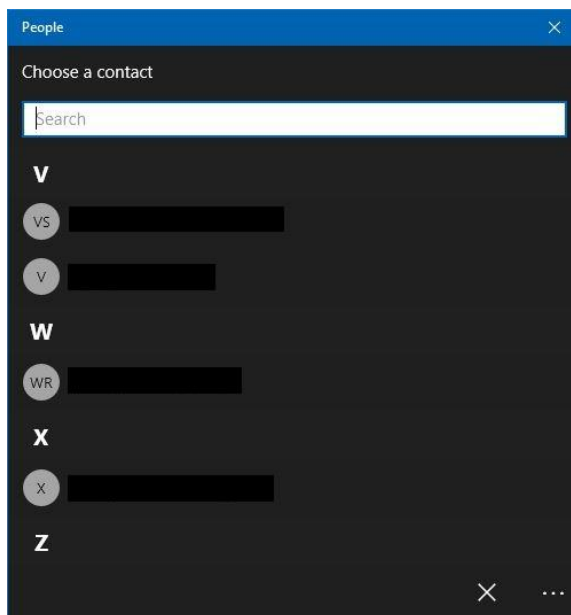


Obrázek 30 - Okno pro přidání upozornění

Zdroj: vlastní

### 5.7.3 Contact Manager API

Toto API poskytuje funkce pro vybrání jednoho nebo více kontaktů a získání jejich informací jako například e-mail, telefonní číslo nebo jméno, přičemž v aplikaci je využíváno pouze jméno pro snazší přidávání účastníků k cestovnímu rozpočtu. Samotné vybrání probíhá přes vyskakovací okno viz. Obrázek 31, které obsahuje seznam kontaktů.

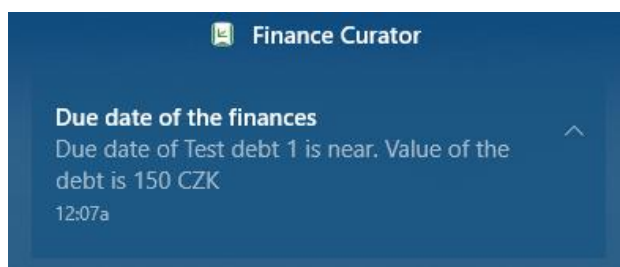


Obrázek 31 - Okno pro vybrání kontaktu

Zdroj: vlastní

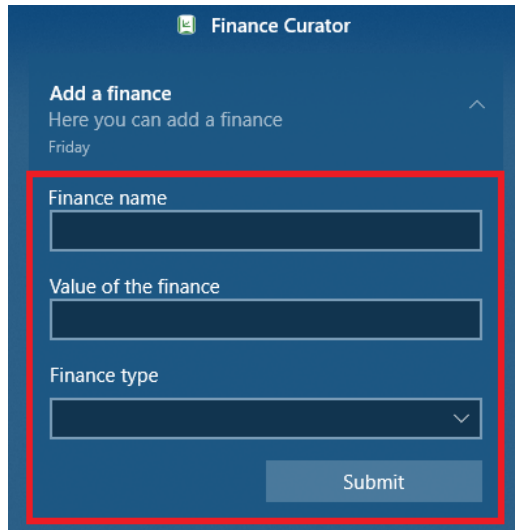
## 5.8 Notifikace

V aplikaci *Finance Curator* je implementována práce s notifikacemi viz. jak s *toast* viz. Obrázek 32, tak i s interaktivními. Struktura *toast* notifikací se sestává z vizuálu, který obsahuje text nebo také může obsahovat obrázky o velikosti 200 KB až 3 MB. Dalším aspektem notifikace je audio, které je přehráno v době jejího zobrazení, přičemž lze změnit ze základního na vlastní. V aplikaci jsou implementovány interaktivní notifikace, jež obsahují stejné části jako *toast*, ale jsou doplněny o ovládací prvky viz. Obrázek 33, kdy tyto notifikace a hodnoty z jejich ovládacích prvků lze zachytit na pozadí pomocí procesu na pozadí nebo mohou předat argument pro nějakou funkci.



Obrázek 32 - Ukázka toast notifikace

Zdroj: vlastní



Obrázek 33 - Ukázka interaktivní notifikace

Zdroj: vlastní



## 5.9 Uživatelské nastavení aplikace

Důležitou součástí aplikací je možnost nastavení některých funkcí uživatelem, kdy se může jednat například o nastavení barevného motivu, povolení notifikací či procesů na pozadí nebo v případě aplikace *Finance Curator* se jedná o nastavení zobrazované měny nebo vracení určitých notifikací či synchronizace dat na pozadí a pro práci s těmito nastaveními má třídu *SettingsHelper*.

Tato nastavení jsou buď lokální nebo tzv. *roaming* a mohou to být hodnoty s datovými typy jako *Int*, *Single*, *Double*, *String*, *DateTime* či *Boolean* nebo se může jednat o soubory. V případě lokálních jde o data, která je potřeba uložit mezi používáním aplikace nebo jde o informace, které jsou specifické pro dané zařízení, respektive nejsou vhodné pro jejich *roaming* mezi zařízeními, přičemž na tyto nastavení nejsou kladeny žádné omezení jejich velikosti. Aplikace *Finance Curator* využívá tento typ nastavení pro ukládání SID uživatele a jeho jméno a pro uživatelská nastavení. (76)

Kromě lokálních lze využít tzv. *roaming* nastavení, které mají na rozdíl od lokálních omezenou velikost a synchronizují se napříč zařízeními, přičemž vše má na starost operační systém. Celý proces se sestává ze zkopírování dat v době jejich změny a následné nahrání na server odkud se pak budou stahovat na jiná zařízení. Příkladem těchto nastavení je nastavení barevného motivu aplikace. (76)

## 6. Uživatelská dokumentace

V tomto úseku jsou uvedeny požadavky na cílové zařízení, dále konfigurace daného zařízení nebo instalace balíčku, aby aplikace mohla být spuštěna. Jsou zde také popsány jednotlivé části celé funkcionality, jako například autentizace uživatele, vytváření peněženek a kategorií, přidávání financí do peněženky a export dat aplikace do zvoleného formátu. Účelem této části je poskytnout průvod jednotlivými funkcemi aplikace pro její správné použití.

### 6.1 Požadavky na zařízení

Tato podkapitola obsahuje dvě tabulky, jedna s minimálními hardwarovými a druhá se softwarovými požadavky na cílové zařízení.

Architektura procesoru	X86, x64, ARM
Velikost operační paměť	<ul style="list-style-type: none"><li>• 1 GB pro 32-bitovou verzi</li><li>• 2 GB pro 64-bitovou verzi</li></ul>
Velikost interní paměti	<ul style="list-style-type: none"><li>• 8 GB pro mobilní zařízení</li><li>• 16 GB pro ostatní zařízení s 32-bitovou verzí</li><li>• 20 GB pro ostatní zařízení s 64-bitovou verzí</li></ul>

**Tabulka 9 - Minimální hardwarové požadavky**

**Zdroj: (77)**

Operační systém	Windows 10 32/64-bit
Verze operačního systému	1607 (Anniversary Update, sestavení 14393)

**Tabulka 10 - Minimální softwarové požadavky**

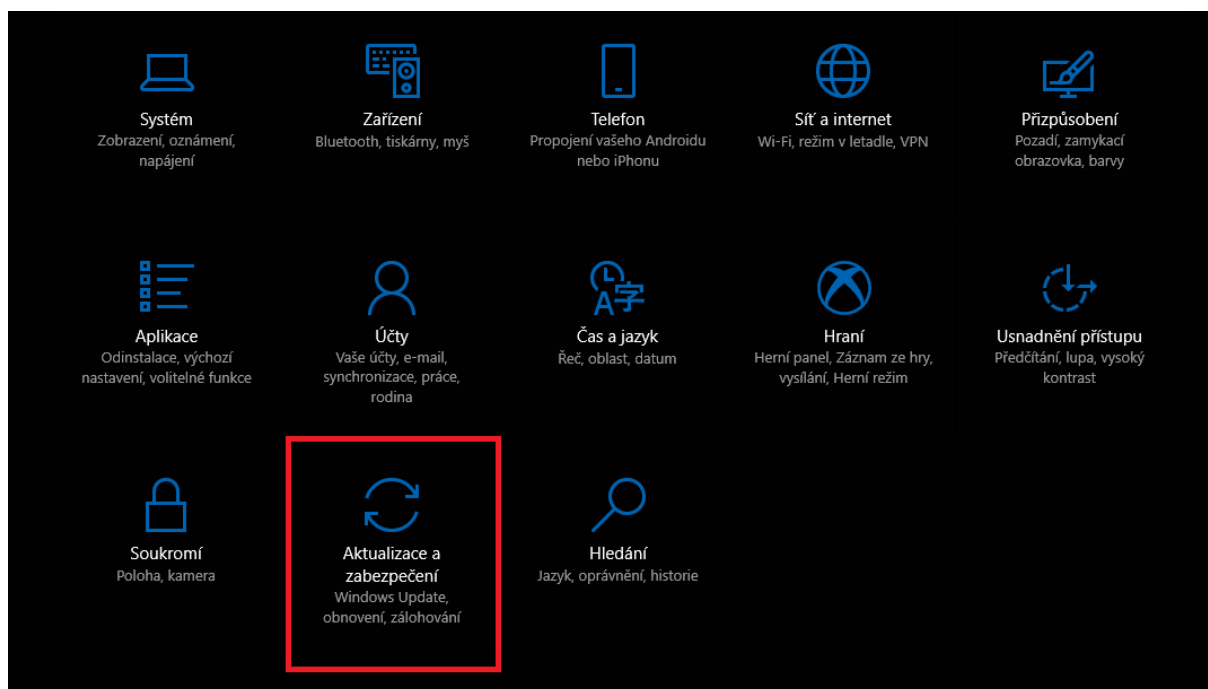
**Zdroj: vlastní**

## 6.2 Konfigurace systému pro instalaci aplikace

V této části je popsán postup pro nastavení vývojářského módu na zařízení s operačním systémem Windows 10 pro nainstalování aplikačního balíčku nebo pro práci se zdrojovými soubory projektu.

Postup konfigurace:

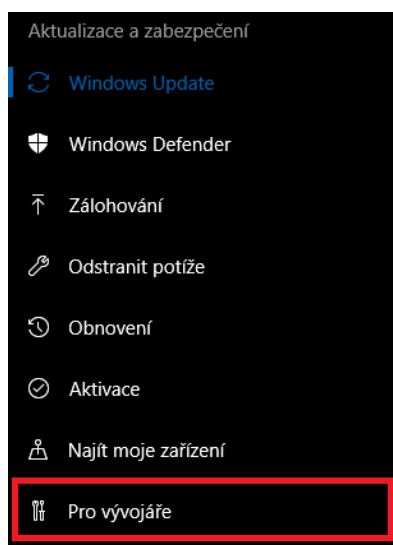
1. Přejít do sekce *Aktualizace a zabezpečení* v nastavení operačního systému viz. Obrázek 34.



Obrázek 34 - Sekce *Aktualizace a zabezpečení* v nastavení operačního systému

Zdroj: vlastní

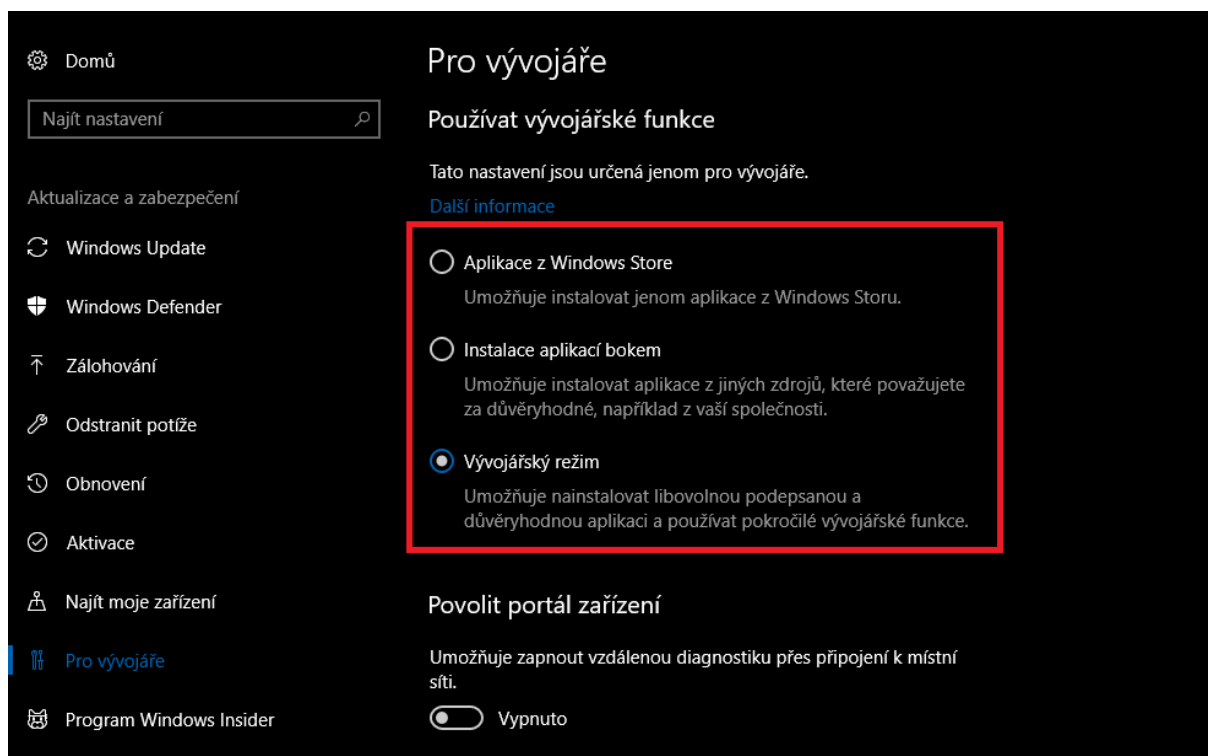
2. Zobrazení možností nastavení v podsekci *Pro vývojáře* viz. Obrázek 35.



Obrázek 35 - Podsekce *Pro vývojáře* v nastavení operačního systému

Zdroj: vlastní

3. Vybrání nastavení viz. Obrázek 36, přičemž jsou k dispozici tři možnosti. První je standardní režim pro instalování aplikací pouze z Microsoft Store, další je *Instalace aplikací bokem* pro instalování z jiných zdrojů. Poslední je *Vývojářský režim* pro jakékoliv aplikace a využívání vývojářských funkcí.



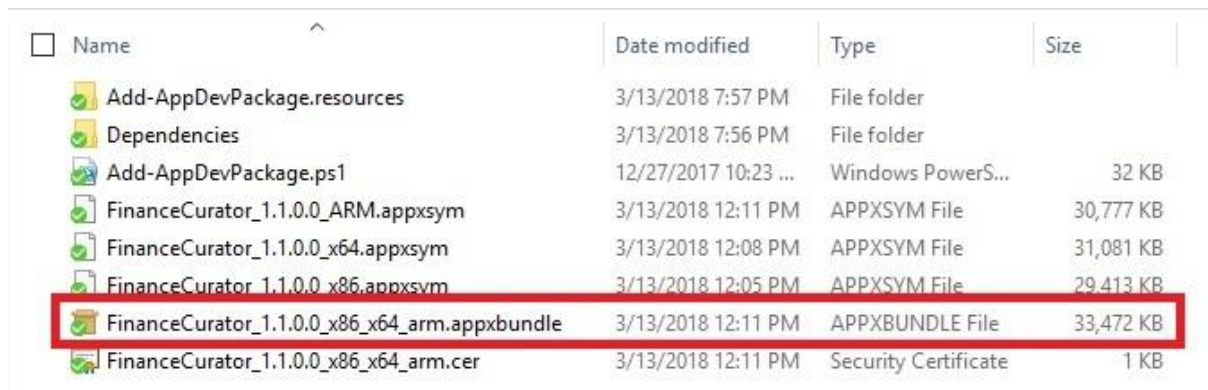
Obrázek 36 – Možnosti nastavení vývojářských funkcí na zařízení

Zdroj: vlastní

Zdroje: (78)

### 6.3 Instalace aplikačního balíčku

V adresáři tohoto dokumentu se nachází složka s názvem *FinanceCuratorAppPackage*, která obsahuje aplikační balíček (soubor s příponou .appxbundle) viz. Obrázek 37. je zde také bezpečnostní certifikát, PowerShell skript, závislosti a další.

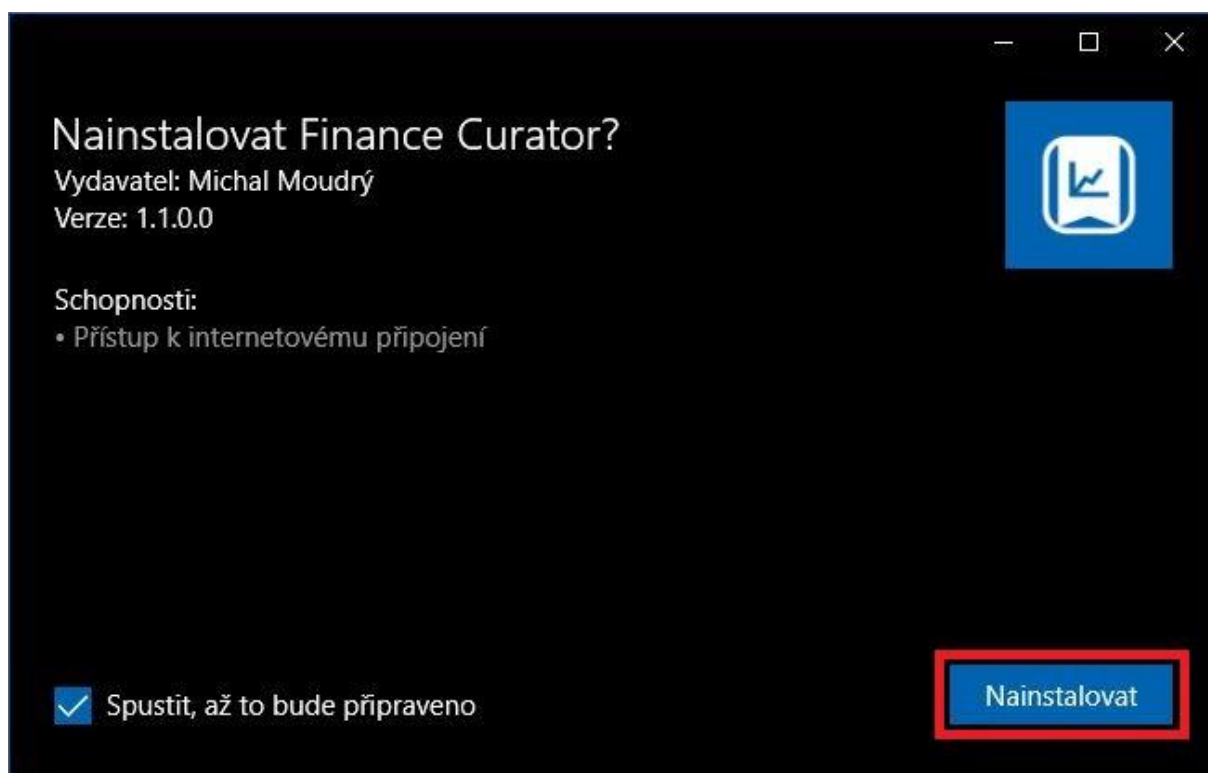


Name	Date modified	Type	Size
Add-AppDevPackage.resources	3/13/2018 7:57 PM	File folder	
Dependencies	3/13/2018 7:56 PM	File folder	
Add-AppDevPackage.ps1	12/27/2017 10:23 ...	Windows PowerS...	32 KB
FinanceCurator_1.1.0.0_ARM.appxsym	3/13/2018 12:11 PM	APPXSVM File	30,777 KB
FinanceCurator_1.1.0.0_x64.appxsym	3/13/2018 12:08 PM	APPXSVM File	31,081 KB
FinanceCurator_1.1.0.0_x86.appxsym	3/13/2018 12:05 PM	APPXSVM File	29,413 KB
<b>FinanceCurator_1.1.0.0_x86_x64_arm.appxbundle</b>	3/13/2018 12:11 PM	APPXBUNDLE File	33,472 KB
FinanceCurator_1.1.0.0_x86_x64_arm.cer	3/13/2018 12:11 PM	Security Certificate	1 KB

Obrázek 37 - Aplikační balíček ve složce FinanceCuratorAppPackage

Zdroj: vlastní

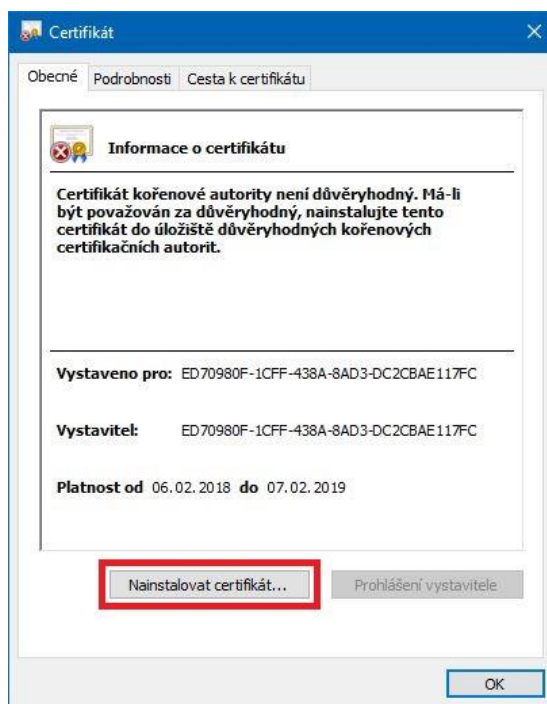
Spuštěním balíčku se spustí instalační proces, kdy prvním krokem je kliknutí na tlačítko s popisem instalovat viz. Obrázek 38. Po úspěšné instalaci se objeví aplikace v seznamu aplikací.



Obrázek 38 - Instalační program pro aplikaci

Zdroj: vlastní

V případě chyby při instalaci balíčku je možné, že se jedná o problém s důvěryhodností balíčku, a proto je zapotřebí nainstalovat na lokální zařízení bezpečnostní certifikát s příponou .cer do uložení důvěryhodných kořenových certifikačních autorit a do uložení důvěryhodných osob, přičemž samotný proces se liší podle zařízení. V případě mobilního zařízení je instalace certifikátu snazší, protože se o to postará instalační program, ale na jiných zařízeních je ho potřeba nainstalovat ručně viz. Obrázek 39.

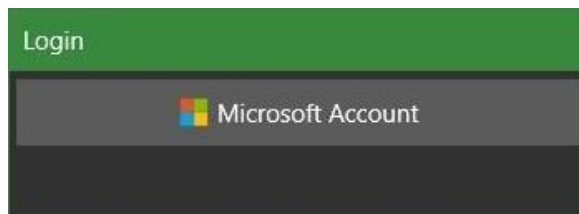


Obrázek 39 - Spuštění instalace bezpečnostního certifikátu

Zdroj: vlastní

## 6.4 Autentizace

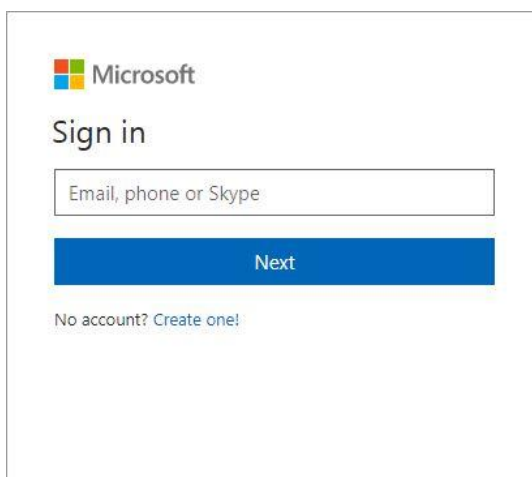
Pro používání aplikace je nutné provést autentizaci, kdy při prvním spuštění se objeví okno s tlačítkem pro přihlášení pomocí Microsoft Account viz. Obrázek 40.



**Obrázek 40 - Okno s tlačítkem pro přihlášení uživatele**

**Zdroj: vlastní**

Po stisknutí daného tlačítka se spustí výchozí webový prohlížeč, ve kterém bude zobrazena webová stránka s formulářem pro přihlášení přes *Microsoft Account* viz. Obrázek 41.



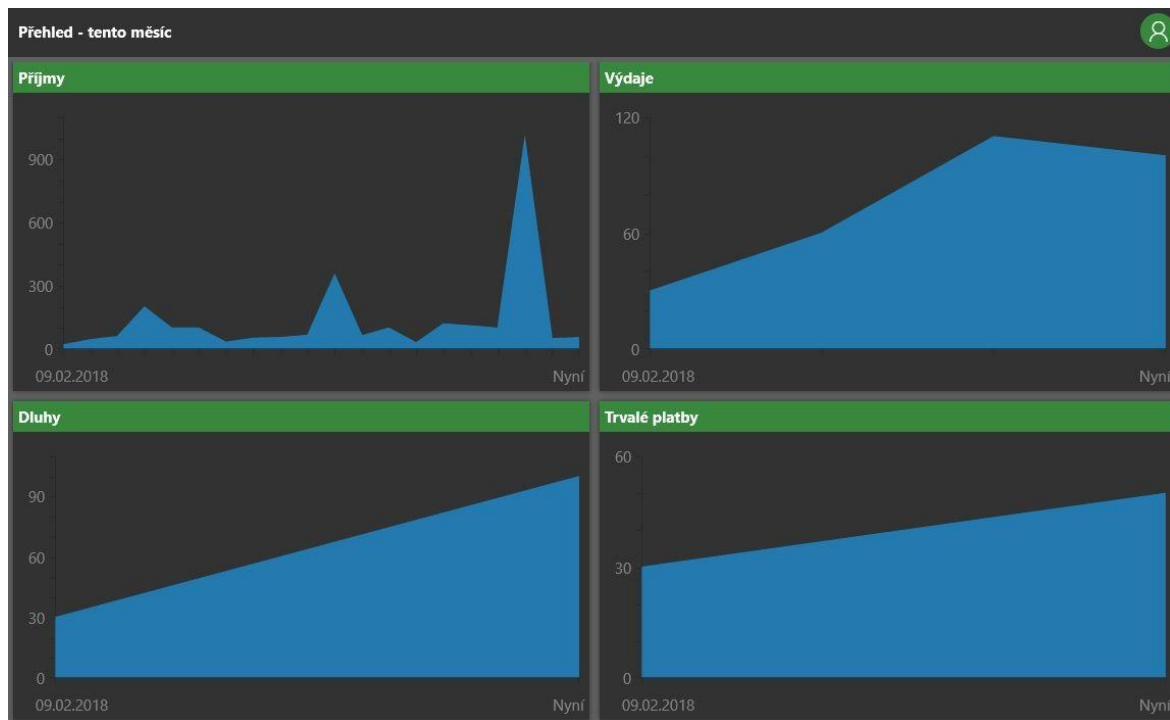
**Obrázek 41 - Formulář pro přihlášení pomocí účtu Microsoft**

**Zdroj: vlastní**

Po správném vyplnění daného formuláře se objeví okno se seznamem všech povolení, které aplikace vyžaduje. Pro dokončení autentizace je nutné tyto povolení odsouhlasit. Po souhlasu se aktivuje aplikace a následuje ukládání uživatelských informací a stahování již existujících dat.

## 6.5 Přehled financí

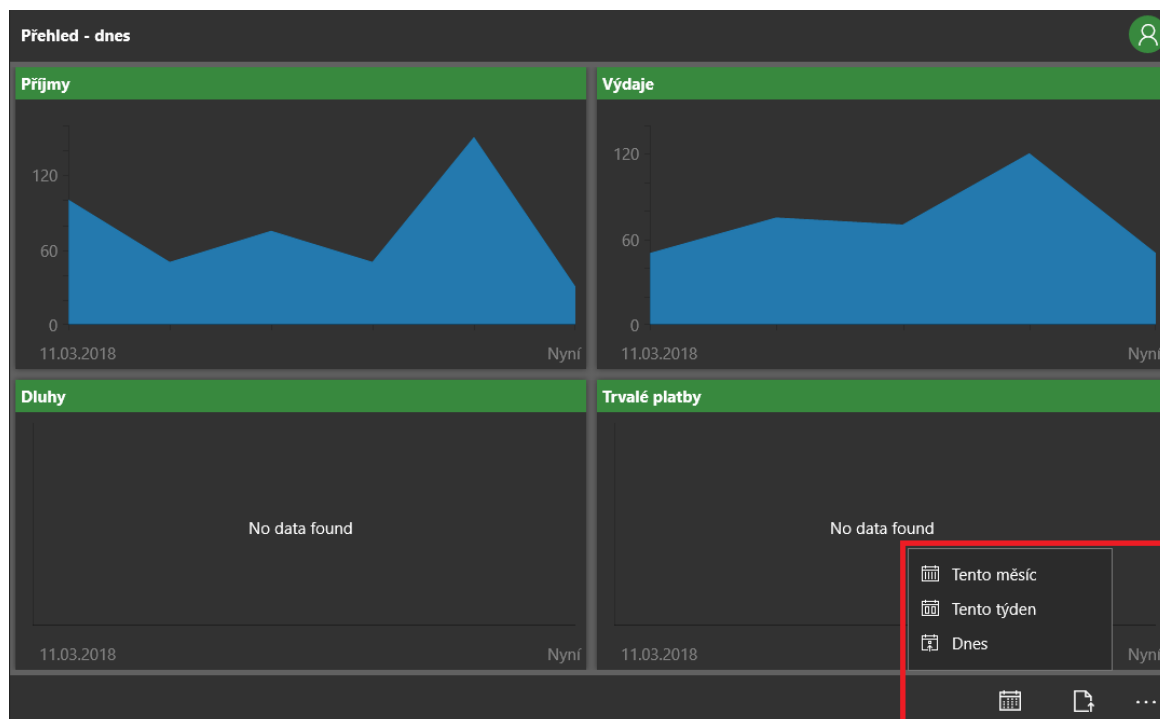
V aplikaci existují celkem 3 možnosti přehledu financí. První je na hlavní stránce, kde jsou na grafu znázorněny součty hodnot financí na časové ose viz. Obrázek 42, přičemž rozmezí dat je možno nastavit na daný den, minulý týden a minulý měsíc po kliknutí na tlačítko s ikonou kalendáře viz. Obrázek 43.



Obrázek 42 - Celkový přehled financí

Zdroj: vlastní

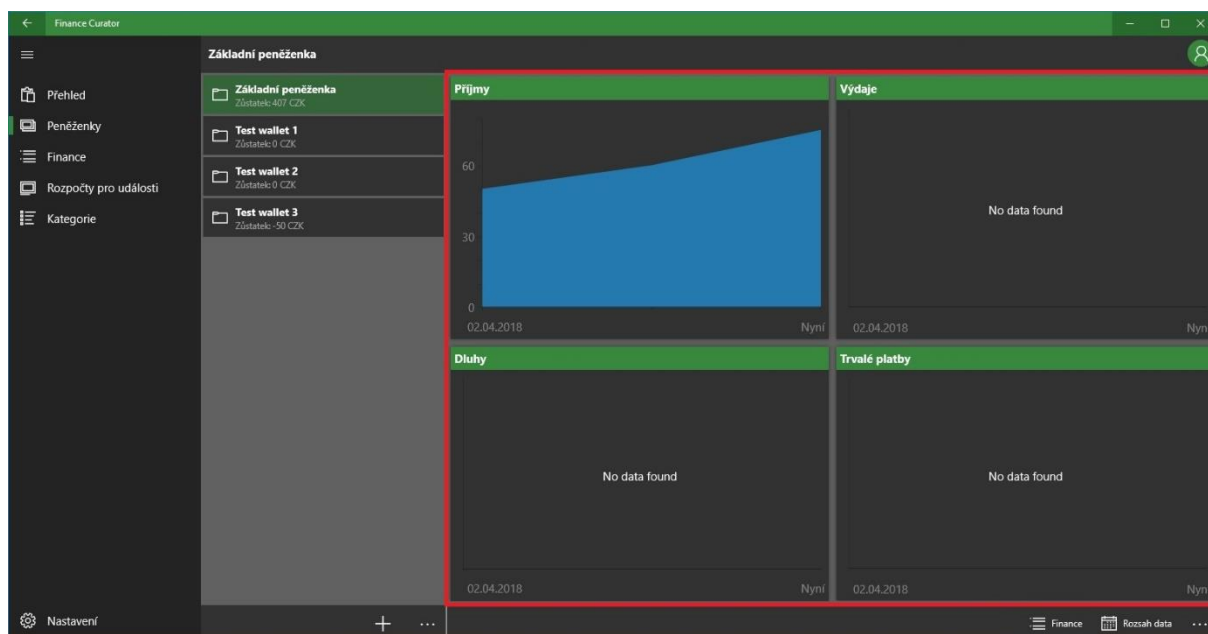




Obrázek 43 - Nastavení rozmezí data pro grafy

Zdroj: vlastní

Druhou možností je přehled financí pro zvolenou peněženku a je zobrazen stejným způsobem jako na hlavní stránce viz. Obrázek 44. Nastavení časového rozmezí funguje stejně jako u předchozí možnosti.

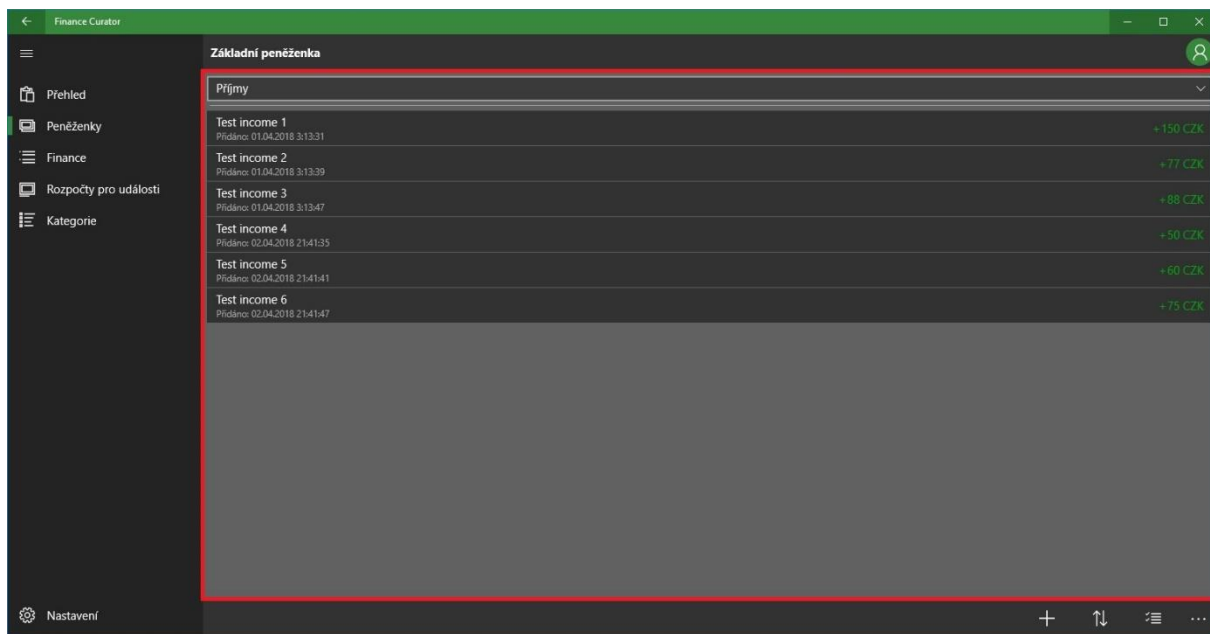


Obrázek 44 - Přehled financí v konkrétní peněženke

Zdroj: vlastní

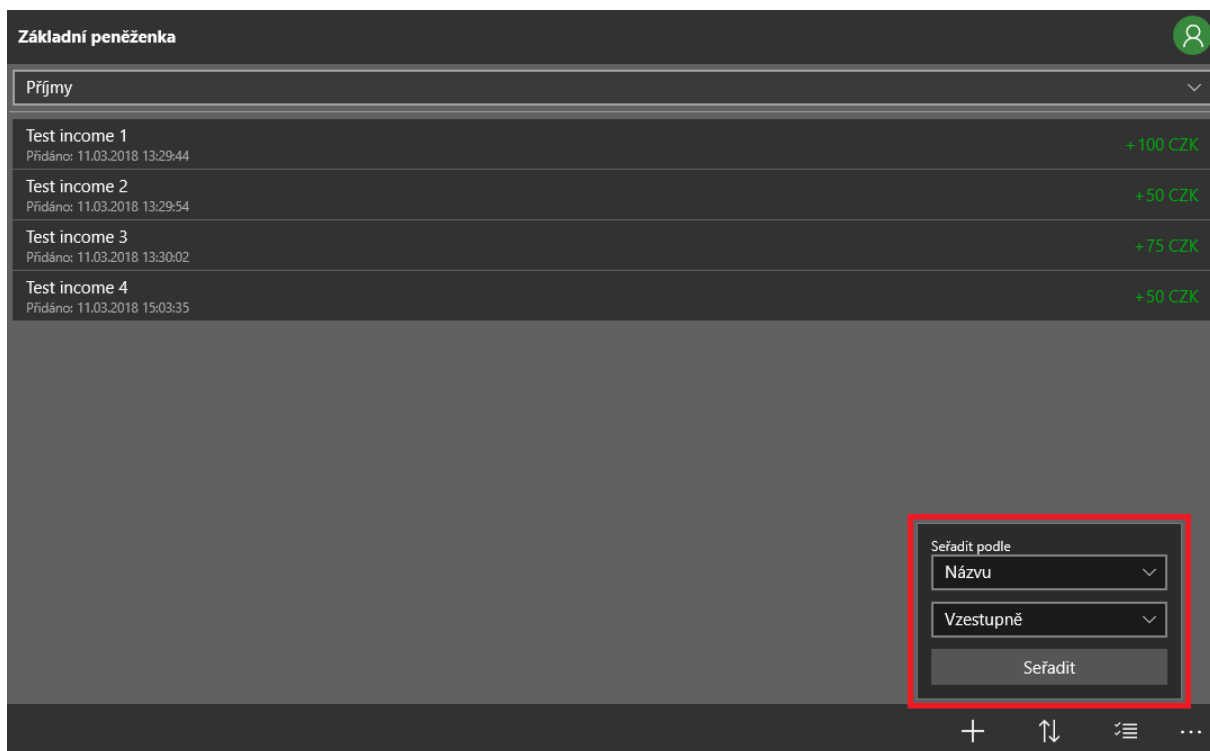
Poslední variantou přehledu financí je zobrazení financí ve vytvořené peněženke v seznamu viz. Obrázek 45, přičemž zobrazování všech typů financí je řešeno ovládacím

prvkem, který se nachází nad seznamem a kliknutím na něj se zobrazí seznam s typy financí a z něhož se zvolí zobrazovaný typ. Po kliknutí a položku seznamu se zobrazí okno se vstupními poli pro úpravu dané finance. Tento seznam je také možno seřadit podle určených parametrů, které jsou v okně viz Obrázek 46.



Obrázek 45 - Přehled financí v seznamu

Zdroj: vlastní

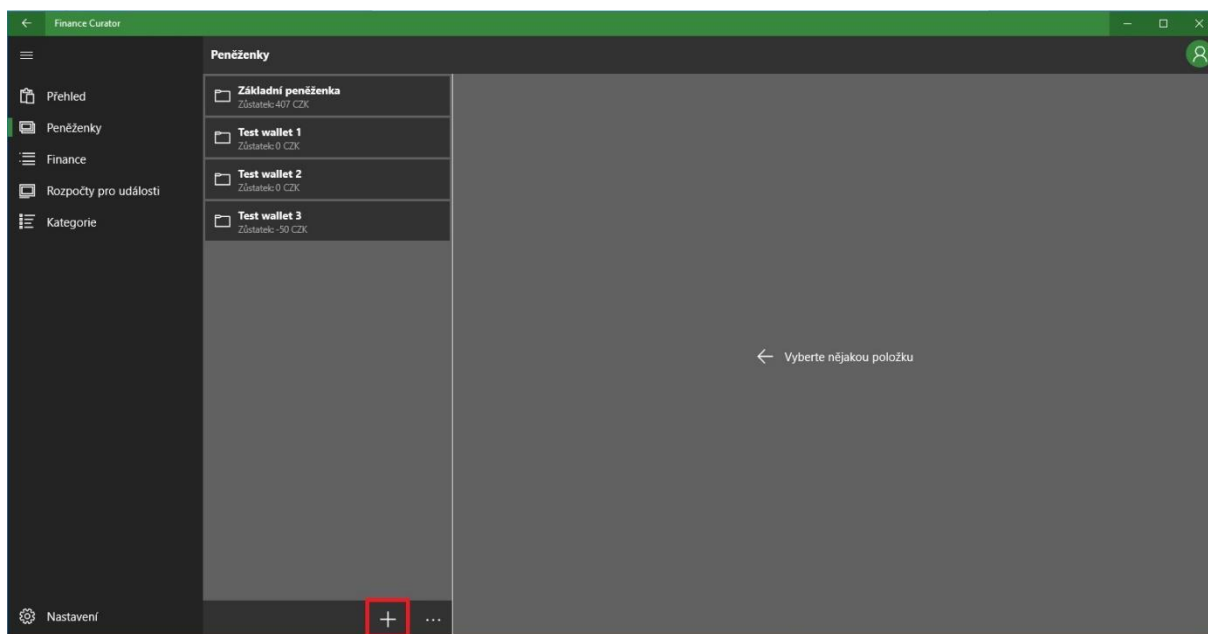


Obrázek 46 - Okno pro nastavení parametrů pro řazení seznamu financí

Zdroj: vlastní

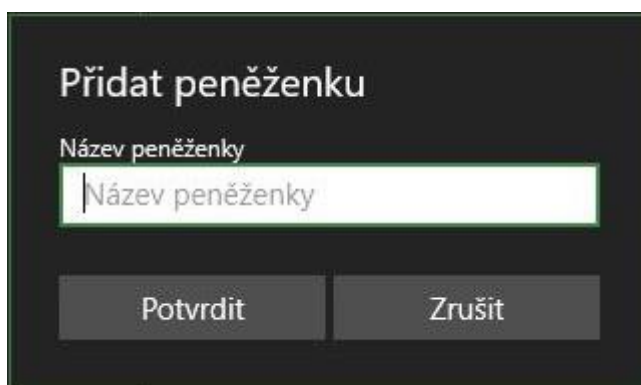
## 6.6 Vytvoření peněženky

Na stránce s názvem *Peněženky* se nachází seznam s existujícími peněženkami a pod ním se nachází tlačítko s ikonou plusu viz. Obrázek 47. Po kliknutí na dané tlačítko se otevře okno viz. Obrázek 48, jež obsahuje vstupní pole pro jméno peněženky a po jeho správném vyplnění je přidána peněženka. Po potvrzení okna se zobrazí upozornění, které obsahuje informaci o úspěšnosti přidání, kdy v případě úspěchu má zelené pozadí viz. Obrázek 49 a pokud ne, tak má pozadí červené, přičemž stejný typ upozornění je i pro ostatní položky.



Obrázek 47 - Seznam přidanych peněženek a tlačítko pro přidání peněženky

Zdroj: vlastní



Obrázek 48 - Okno pro přidávání peněženek

Zdroj: vlastní

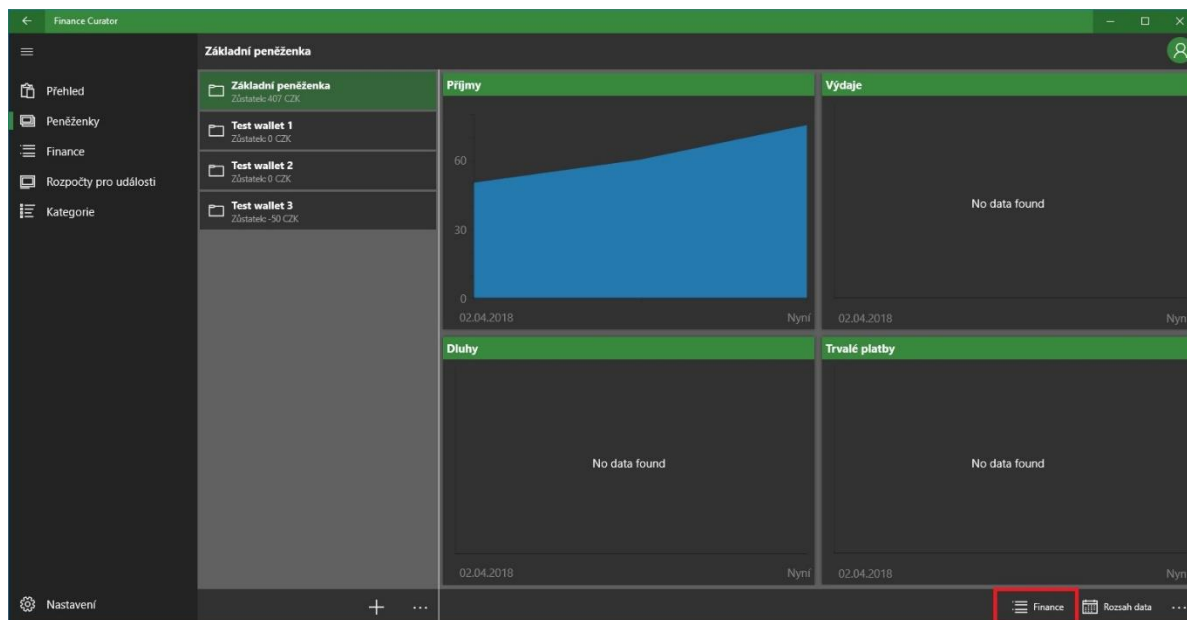


Obrázek 49 - Upozornění o úspěšnosti přidání peněženky

Zdroj: vlastní

## 6.7 Přidání finance do peněženky

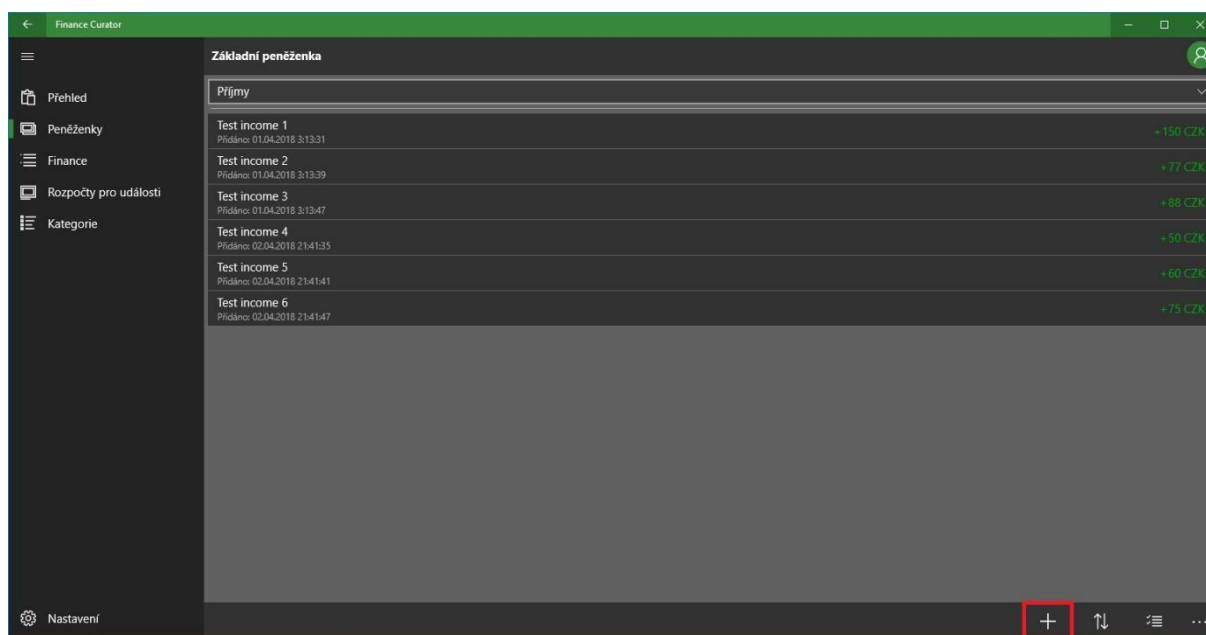
Pro přidání finance do peněženky je nutné přejít na stránku se seznamem financí, na kterou lze navigovat kliknutím na tlačítko s ikonou seznamu, jež se nachází v detailech peněženky viz. Obrázek 50.



Obrázek 50 - Tlačítko pro navigaci na stránku s financemi v peněžence

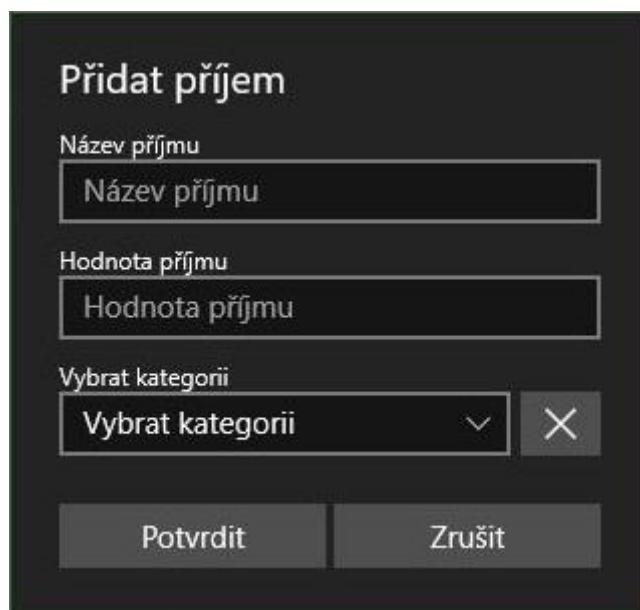
Zdroj: vlastní

Přidání finance závisí na zvolení typu zobrazovaných financí z čehož se pak vychází po stisknutí tlačítka s ikonou plus, které se nachází pod seznamem financí viz. Obrázek 51, kdy například jsou zobrazeny příjmy a zobrazí se tedy okno pro přidání příjmu viz. Obrázek 52.



Obrázek 51 - Tlačítko pro přidání finance do peněženky

Zdroj: vlastní



**Přidat příjem**

Název příjmu  
Název příjmu

Hodnota příjmu  
Hodnota příjmu

Vybrat kategorii  
Vybrat kategorii

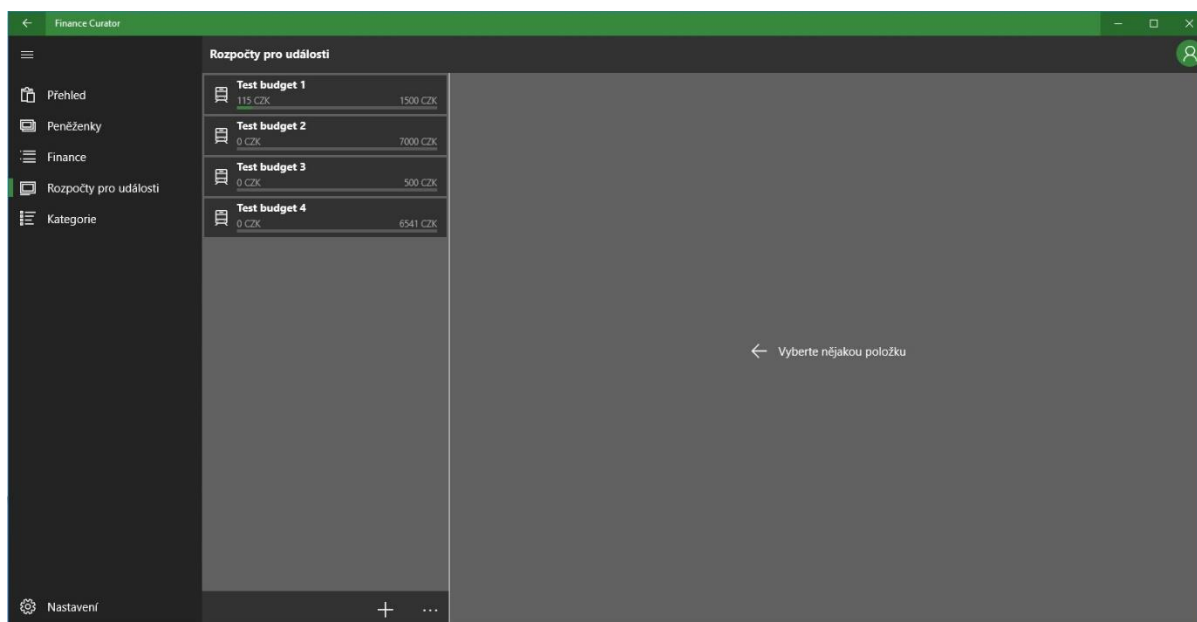
Potvrdit Zrušit

Obrázek 52 - Okno pro přidání příjmu

Zdroj: vlastní

## 6.8 Přehled cestovních rozpočtů

Cestovní rozpočty jsou graficky zobrazeny podobně jako peněženky, jen v seznam mají položky jiné údaje a jejich detail obsahuje informace o datu vytvoření, hodnotě rozpočtu a kolik již bylo uhrazeno. Přičemž rozdíl mezi hodnotou rozpočtu a údajem o celkové uhrazené částce je zobrazen na grafu pod danými informacemi viz. Obrázek 53. Následně je zde i seznam účastníků, u kterých je zobrazeno kolik dluží a kolik zaplatili.

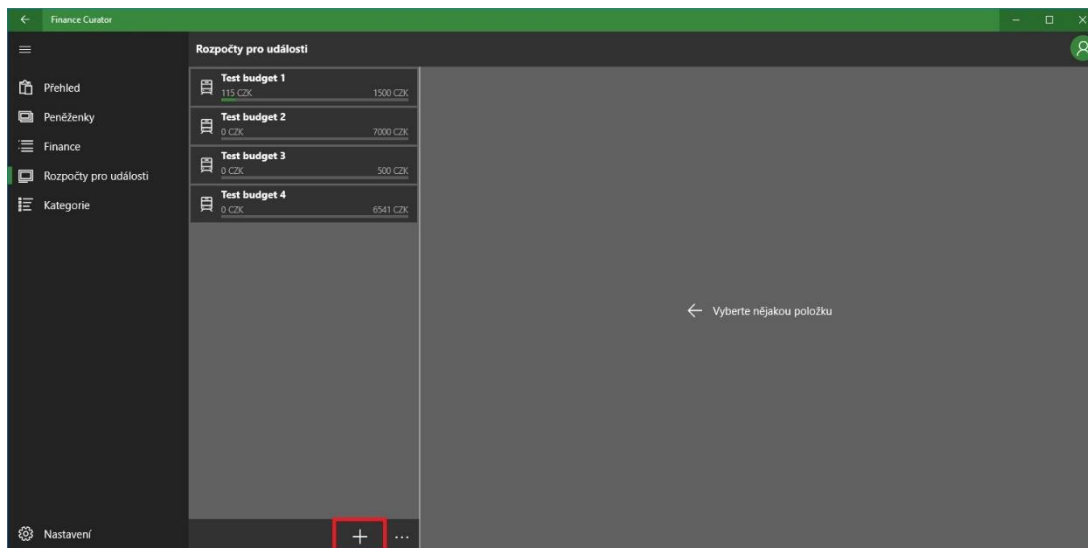


Obrázek 53 - Přehled rozpočtů

Zdroj: vlastní

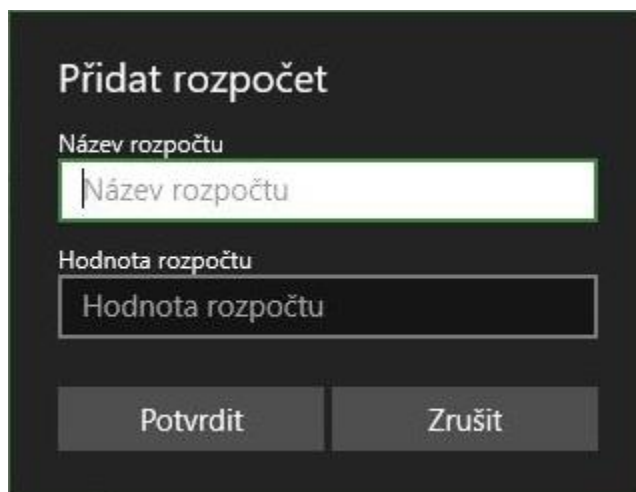
## 6.9 Vytvoření rozpočtu

Na stránce s rozpočty se nachází seznam s existujícími rozpočty a pod ním je tlačítko s ikonou plus viz. Obrázek 54, po kterém se zobrazí okno pro přidání rozpočtu, jež obsahuje vstupní pole pro název a hodnotu nového rozpočtu viz. Obrázek 55.



Obrázek 54 – Tlačítko pro přidání rozpočtu

Zdroj: vlastní

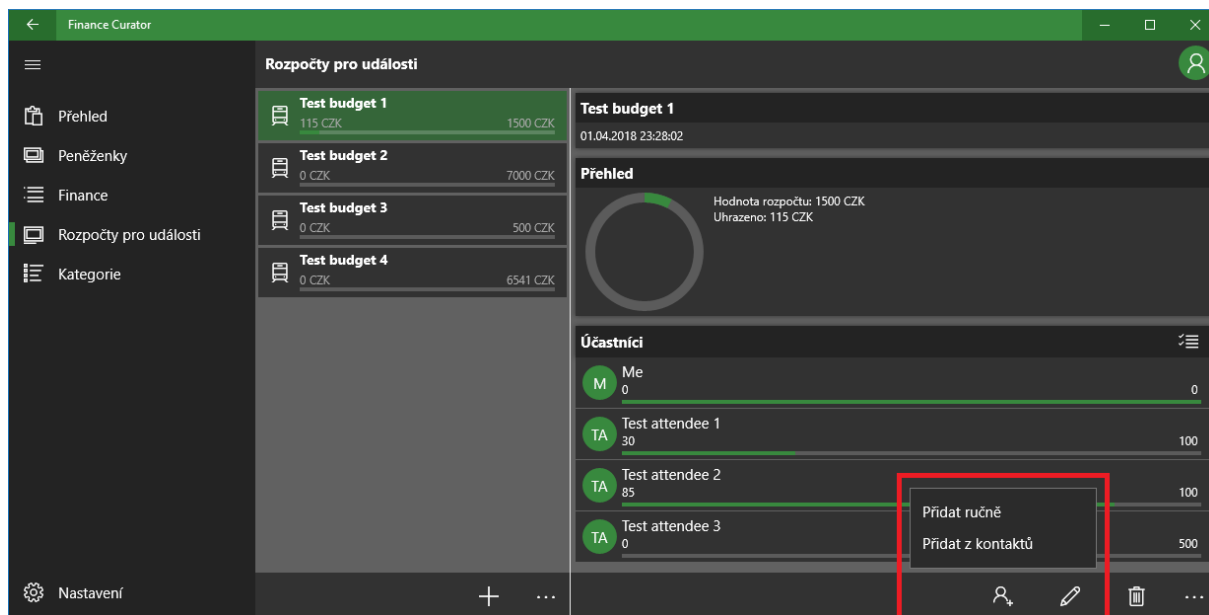


Obrázek 55 - Okno pro přidání rozpočtu

Zdroj: vlastní

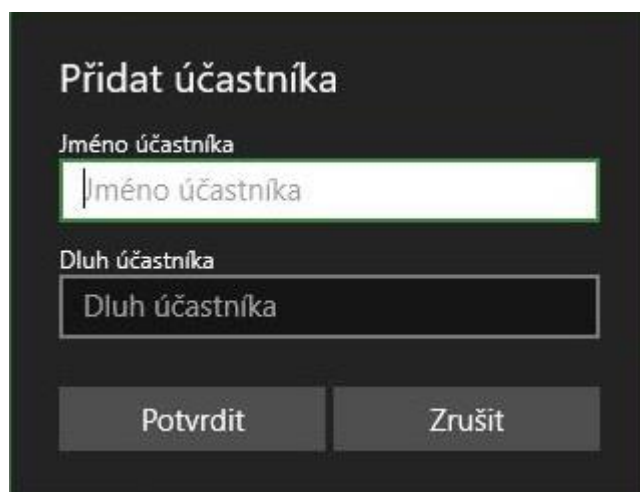
## 6.10 Přidání účastníka k rozpočtu

V detailu rozpočtu se nachází seznam s účastníky, kteří jsou k němu přiřazeni a další lze přidat po stisknutí tlačítka s ikonkou postavy s plusem, poté se zobrazí okno s možnostmi ručního přidání nebo vybrání z kontaktů viz. Obrázek 56. V obou případech se ve výsledku zobrazí okno se vstupními poli viz. Obrázek 57, jen v případě přidání z kontaktů bude pole se jménem předvyplněné.



Obrázek 56 - Tlačítko pro přidání účastníka a možnosti přidání

Zdroj: vlastní

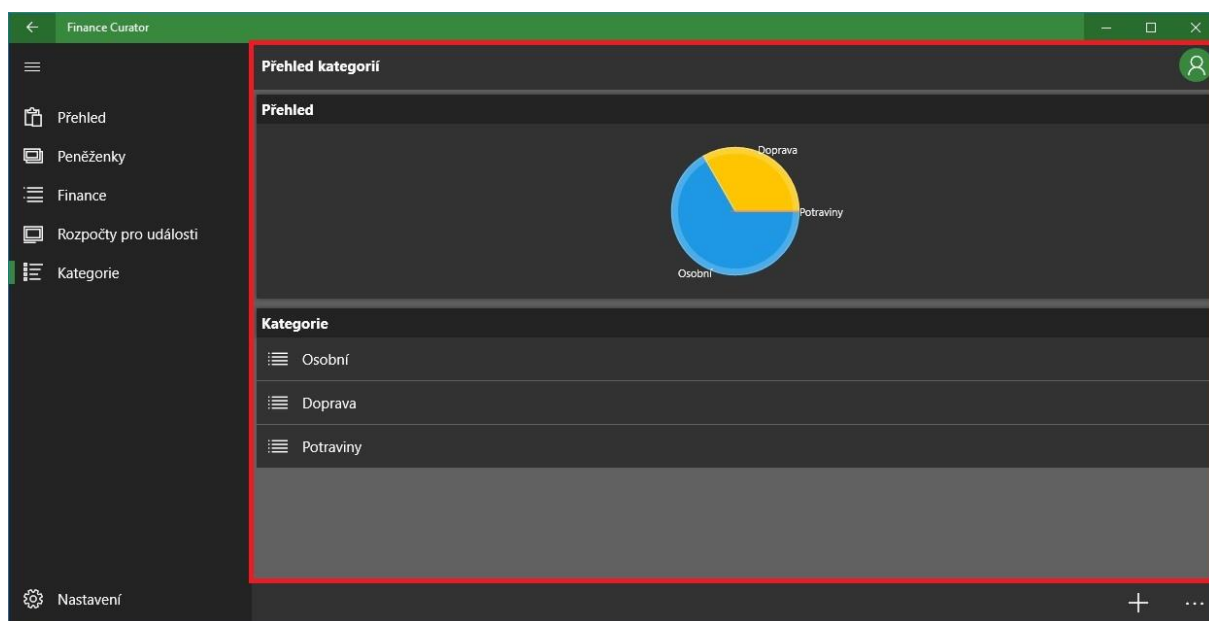


Obrázek 57 - Okno pro přidání účastníka

Zdroj: vlastní

## 6.11 Přehled kategorií

Na stránce *Kategorie* jsou jednotlivé kategorie zobrazeny na koláčovém grafu, jehož části jsou barevně rozděleny viz. Obrázek 58, přičemž hodnota jedné části grafu odpovídá počtu financí v rámci dané kategorie. Pod tímto grafem se nachází seznam s přidávanými kategoriemi a po kliknutí a nějakou položku se zobrazí okno, které obsahuje vstupní pole pro úpravu názvu kategorie a pod ním se nachází seznam s jednotlivými typy financí a jejich počet, jež byly zařazeny do dané kategorie a ještě pod tímto seznamem se nachází tlačítko pro smazání.

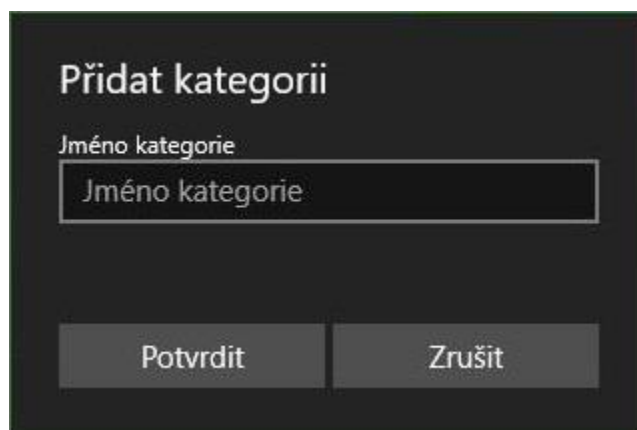


Obrázek 58 - Přehled kategorií v aplikaci

Zdroj: vlastní

## 6.12 Vytvoření kategorie

Pod přehledem přidávaných kategorií se nachází tlačítko s ikonou plusu a po jeho stisknutí se zobrazí okno, jež obsahuje vstupy pro přidání kategorie viz. Obrázek 59.



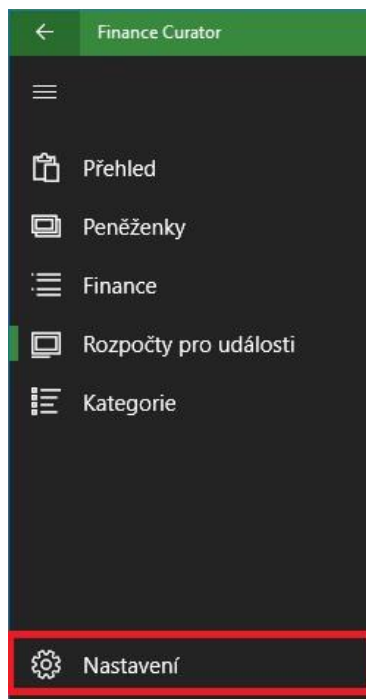
Obrázek 59 - Okno pro přidání kategorie

Zdroj: vlastní



## 6.13 Nastavení aplikace

Odkaz na nastavení aplikace je umístěn v dolní části hlavního navigačního vzorce a je označen ikonou ozubeného kola viz. Obrázek 60.

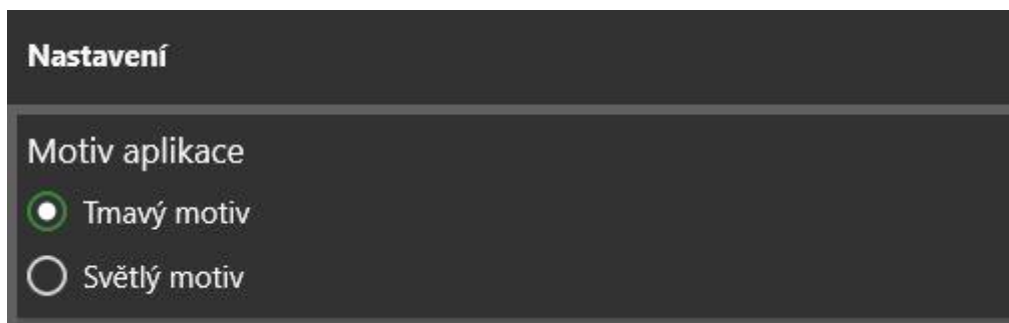


Obrázek 60 - Odkaz na stránku s nastavením aplikace

Zdroj: vlastní

- **Nastavení motivu aplikace**

Aplikace podporuje tmavý a světlý motiv, kdy samotné nastavení se nachází v prvním bloku na stránce s nastavením viz. Obrázek 61, přičemž změna motivu je okamžitá a nevyžaduje restartování aplikace, jako tomu je některých ostatních.

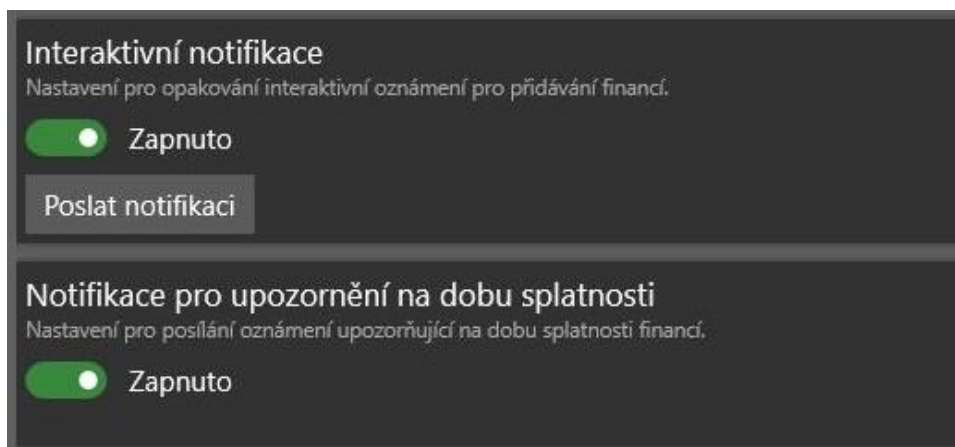


Obrázek 61 - Nastavení motivu aplikace

Zdroj: vlastní

- **Nastavení notifikací**

Nachází se zde také nastavení notifikací, což zahrnuje povolení pro vracení interaktivních notifikací, a jestli chce uživatel obdržet upozornění o blížícím se datu splatnosti některých financí viz. Obrázek 62. Přičemž nastavení opakování vracení notifikací je doplněno o tlačítko pro manuální posílání notifikace.

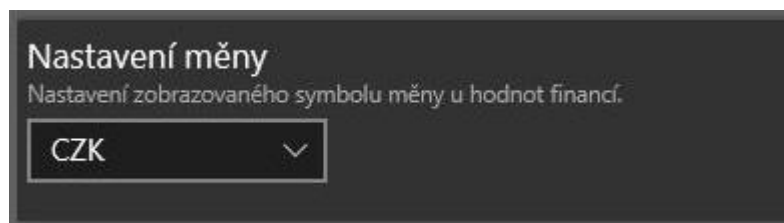


Obrázek 62 - Nastavení notifikací

Zdroj: vlastní

- **Nastavení symbolu zobrazované měny**

V aplikaci lze nastavit symbol zobrazované měny, což je realizováno ovládacím prvkem pro vybírání hodnot viz. Obrázek 63. Přičemž změna se projeví okamžitě po daném výběru.

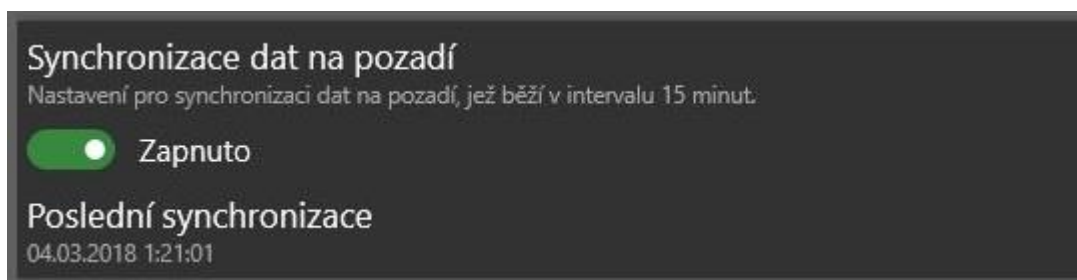


Obrázek 63 - Nastavení symbolu zobrazované měny

Zdroj: vlastní

- **Nastavení synchronizace na pozadí**

Nastavení dané synchronizace se týká jejího povolení a k tomu je zobrazen údaj o datu posledního provedení jakékoliv synchronizace viz. Obrázek 64.

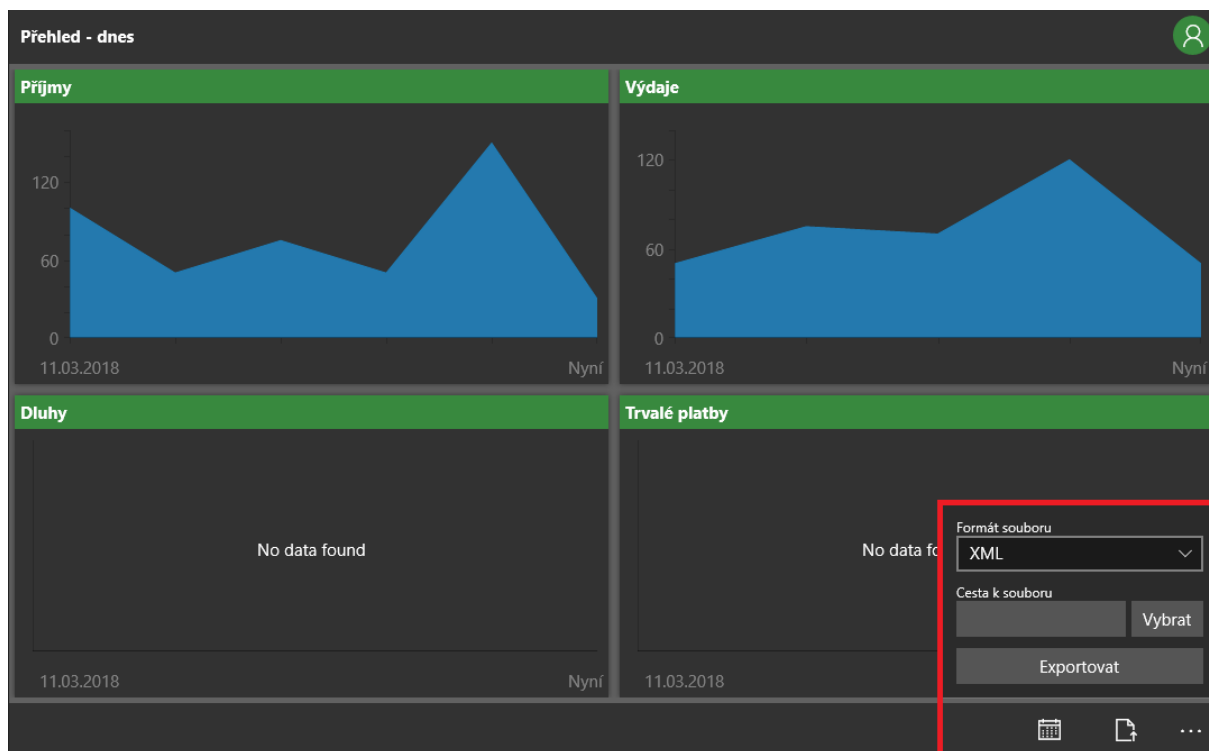


Obrázek 64 - Nastavení synchronizace na pozadí

Zdroj: vlastní

## 6.14 Export dat aplikace

Tlačítko pro export dat aplikace se nachází na hlavní stránce hned vedle tlačítka pro nastavení časového rozmezí pro finance a má ikonu souboru s šipkou nahoru viz. Obrázek 65. Po jeho stisknutí se objeví okno, jež obsahuje ovládací prvek pro vybrání formátu výsledného souboru, pak je možnost vybrání složky společně se zobrazením její cesty, kde bude daný soubor uložen. Poslední věc, co obsahuje je tlačítko pro spuštění exportu.



Obrázek 65 - Okno pro export dat aplikace

Zdroj: vlastní

## 7. Závěr

Cílem této práce bylo vytvoření univerzální aplikace pro operační systém Windows 10, přičemž měla běžet na různých typech zařízení s daným OS. Mezi její hlavní funkce patří evidování financí, jejich řazení do kategorií, jež mohli být vytvořené samotným uživatelem. Dále aplikace umožňovala správu cestovních rozpočtů, což zahrnuje přidávání daných rozpočtů a účastníků, jež se finančně podílely na rozpočtu. Dalším důležitým aspektem aplikace byla autentizace uživatele a synchronizace jeho dat napříč všemi zařízeními, na který je aplikace nainstalována, což bylo dosaženo využitím cloudové služby Azure App Service na platformě Microsoft Azure, jež poskytla potřebnou funkcionalitu, jako práci s poskytovatelem identity pro autentizaci nebo uložení dat v cloudu společně s řešením synchronizace s lokálním uložištěm.

K hlavním funkcím aplikace byly přidána dodatečné funkce, které přidali další funkce pro práci s daty, uživatelským nastavením aplikace nebo usnadnění používání aplikace. Mezi tyto funkce patří export dat, kdy byla přidána podpora pro formáty jako XML nebo HTML a další. Další bylo nastavení, kde uživatel mohl nastavit, jak aplikace bude vypadat nebo jaké symboly budou zobrazeny u jednotlivých financí. Co týče usnadnění používání aplikace, tak se jednalo o interaktivní notifikace, jež umožňují přidat příjem nebo výdaj do základní peněženky.

Aplikace také pracuje s různými systémovými API pro využití systémových aplikací, jako například vytvoření události v kalendáři pro trvalé platby nebo dluhy, což bylo uskutečněno za pomoci Appointments API. Dalším příkladem využití je přidání účastníka k cestovnímu dluhu přes vyskakovací okno pro kontakty uložené na zařízení nebo rozšíření aplikace o tzv. JumpList, což je seznam zkratk pro definované úlohy.

Dalším cílem bylo získat znalosti o vývoji .NET aplikací, respektive vývoj aplikací pro platformy .NET Core a .NET Native, což zahrnovalo naprogramování univerzální aplikace, jež má název Finance Curator a vypracování rešerše na téma *Vývoj na UWP*, dále pochopení životních cyklů aplikace nebo práce s již zmíněnými systémovými API, ale také naučit se pracovat s cloudovou platformou Microsoft Azure, respektive s PaaS službou Azure App Service a jejími vlastními funkcemi.

Mezi další požadavky na tuto práci samozřejmě patřilo vypracování dvou rešerší, a to na témata *Vývoj na UWP*, jež byla nejdůležitější, protože přibližuje čtenáři technologie pro vývoj univerzálních aplikací, jako například systém pro distribuci knihoven nebo samotnou platformu .NET. Druhá rešerše byla na téma *Osobní finance*, která pojednávala o tematické

stránce aplikace, jako například o finančním plánu či rozvaze. Tyto rešerše se nacházejí v kapitolách 3 a 4.

Co se týče dalšího rozšíření této práce, tak byl šlo rozšířit aplikaci o převod mezi měnami, kdy momentálně lze nastavit pouze symbol měny. Dále přidání podpory pro nové aktualizace operačního systému Windows 10 a implementovat nové funkce pro tyto verze, jako například upravení UI aplikace, aby například korespondovalo s Fluent Design. Následně upravit a zrychlit autentizaci uživatele, aby se nemusel využívat webový prohlížeč. Také by bylo vhodné přidat export dat aplikace do formátu PDF, což z důvodu absence určitých knihoven nešlo implementovat. Dále by šlo k univerzální aplikaci přidat webovou aplikaci, jež by byla vytvořena za pomoci ASP.NET Core a využívala by také Azure App Service.

## 8. Použitá literatura a zdroje

1. **Wigley, Andy a Nixon, Jerry.** Windows 10 - An Introduction to Building Windows Apps for Windows 10 Devices. *MSDN Magazine*. [Online] Microsoft, 5 2015. [Citace: 7. 3 2018.] <https://msdn.microsoft.com/magazine/dn973012.aspx>.
2. **Microsoft.** Windows 10 update history. *Windows support*. [Online] Microsoft. [Citace: 7. 3 2018.] <https://support.microsoft.com/en-us/help/4000823>.
3. **Whitney, Tyler.** Intro to the Universal Windows Platform. *Microsoft Docs*. [Online] Microsoft, 27. 10 2017. [Citace: 12. 2 2018.] <https://docs.microsoft.com/en-gb/windows/uwp/get-started/universal-application-platform-guide>.
4. **Radich, Quinn.** What's a Universal Windows Platform (UWP) app? *Microsoft Docs*. [Online] Microsoft, 3. 22 2018. [Citace: 7. 3 2018.] <https://docs.microsoft.com/en-us/windows/uwp/get-started/whats-a-uwp>.
5. **Zheng, Serena.** Introduction to UWP app design. *Microsoft Docs*. [Online] Microsoft, 12. 13 2017. [Citace: 7. 3 2018.] <https://docs.microsoft.com/en-us/windows/uwp/design/basics/design-and-ui-intro>.
6. **Jacobs, Mike.** Screen sizes and break points for responsive design. *Microsoft Docs*. [Online] Microsoft, 30. 8 2017. [Citace: 8. 3 2018.] <https://docs.microsoft.com/en-us/windows/uwp/design/layout/screen-sizes-and-breakpoints-for-responsive-design>.
7. **Foley, Mary Jo.** Microsoft Design Language: The newest official way to refer to 'Metro'. *ZDNet*. [Online] 29. 10 2012. [Citace: 8. 3 2018.] <http://www.zdnet.com/article/microsoft-design-language-the-newest-official-way-to-refer-to-metro/>.
8. **Massey, Stéphane.** Metro Ui Design Principles. [Online] 15. 2 2012. [Citace: 8. 3 2018.] <http://www.stephanemassey.com/metro-design-principles/>.
9. **Shum, Albert, Smuga, Michael a Roberts, Chad.** Windows Phone UI and Design Language. *Channel 9*. [Online] 5. 2 2010. [Citace: 8. 3 2018.] <http://video.ch9.ms/ecn/mix/10/pptx/CL14.pptx>.
10. **Rubino, Daniel.** What's new in Microsoft Design Language 2 for Windows 10. *Windows Central*. [Online] 21. 3 2015. [Citace: 8. 3 2018.] <https://www.windowscentral.com/whats-new-microsoft-design-language-2-windows10>.
11. **Jacobs, Mike.** The Fluent Design System for UWP apps. *Microsoft Docs*. [Online] Microsoft, 7. 3 2018. [Citace: 8. 3 2018.] <https://docs.microsoft.com/en-us/windows/uwp/design/fluent-design-system/index>.

12. **.NET Foundation.** *dotnetfoundation.org*. [Online] .NET Foundation. [Citace: 8. 3 2018.] <https://www.dotnetfoundation.org/>.
13. **Microsoft.** What is .NET? [Online] Microsoft. [Citace: 8. 3 2018.] <https://www.microsoft.com/net/learn/what-is-dotnet>.
14. **Zikmund, Karel.** .NET Standard. *PowerPoint Online*. [Online] 19. 7 2017. [Citace: 14. 2 2018.] <https://view.officeapps.live.com/op/view.aspx?src=http://www.wug.cz/GetFile.ashx?EventStoredFileID=477>.
15. **Petrusha, Ron.** Get started with the .NET Framework. *Microsoft Docs*. [Online] Microsoft, 17. 10 2017. [Citace: 14. 2 2018.] <https://docs.microsoft.com/en-gb/dotnet/framework/get-started/index>.
16. —. .NET Framework Class Library Overview. *Microsoft Docs*. [Online] 30. 3 2017. [Citace: 14. 2 2018.] <https://docs.microsoft.com/en-us/dotnet/standard/class-library-overview>.
17. —. .NET Framework system requirements. *Microsoft Docs*. [Online] 2. 2 2018. [Citace: 14. 2 2018.] <https://docs.microsoft.com/en-gb/dotnet/framework/get-started/system-requirements>.
18. **Wenzel, Maira.** The .NET Framework and Out-of-Band Releases. *Microsoft Docs*. [Online] Microsoft, 30. 3 2017. [Citace: 15. 2 2018.] <https://docs.microsoft.com/en-gb/dotnet/framework/get-started/the-net-framework-and-out-of-band-releases>.
19. **Kolektiv.** Intro to .NET Core. *GitHub*. [Online] 17. 3 2015. [Citace: 15. 2 2018.] <https://github.com/dotnet/coreclr/blob/master/Documentation/README.md>.
20. **Lander, Rich.** .NET Core Guide. *Microsoft Docs*. [Online] 20. 6 2016. [Citace: 15. 2 2018.] <https://docs.microsoft.com/en-gb/dotnet/core/>.
21. **Jacobson, Daniel.** .NET Native – What it means for Universal Windows Platform (UWP) developers. *Windows Blogs*. [Online] 20. 8 2015. [Citace: 12. 2 2018.] <https://blogs.windows.com/buildingapps/2015/08/20/net-native-what-it-means-for-universal-windows-platform-uwp-developers>.
22. **Kolektiv.** .NET Core 2.0 - Supported OS versions. *GitHub*. [Online] 9. 8 2017. [Citace: 15. 2 2018.] <https://github.com/dotnet/core/commits/master/release-notes/2.0/2.0-supported-os.md>.
23. —. Introduction to .NET Core CLI. *GitHub*. [Online] 10. 3 2017. [Citace: 15. 2 2018.] <https://github.com/dotnet/cli/blob/master/Documentation/general/intro-to-cli.md>.
24. **Pardoe, Andrew.** .NET Native deep dive. *PowerPoint Online*. [Online] 18. 6 2014. [Citace: 17. 2 2018.]

<https://view.officeapps.live.com/op/view.aspx?src=http%3a%2f%2ffiles.channel9.msdn.com%2fthumbnail%2f45d78758-8ab8-4e62-8a73-2e6a4027b49c.pptx>.

25. **Petrusha, Ron.** Compiling Apps with .NET Native. *Microsoft Docs*. [Online] Microsoft, 30. 3 2017. [Citace: 17. 2 2018.] <https://docs.microsoft.com/en-us/dotnet/framework/net-native/index>.

26. **Karas, Vítek a Kotas, Jan.** .NET Native & CoreRT. *PowerPoint Online*. [Online] 19. 7 2017. [Citace: 17. 2 2018.] <https://view.officeapps.live.com/op/view.aspx?src=http://www.wug.cz/GetFile.ashx?EventStoredFileID=479>.

27. **Petrusha, Ron.** .NET Native and Compilation. *Microsoft Docs*. [Online] 30. 3 2017. [Citace: 5. 3 2018.] <https://docs.microsoft.com/en-us/dotnet/framework/net-native/net-native-and-compilation>.

28. **Kolektiv.** Intro to .NET Native and CoreRT. *GitHub*. [Online] 18. 11 2015. [Citace: 17. 2 2018.] <https://github.com/dotnet/coreclr/blob/master/Documentation/intro-to-coreclr.md>.

29. —. .NET Standard FAQ. *GitHub*. [Online] 30. 9 2017. [Citace: 4. 3 2018.] <https://github.com/dotnet/standard/blob/master/docs/faq.md>.

30. **Landwerth, Immo.** Introducing .NET Standard. *.NET Blog*. [Online] 26. 9 2016. [Citace: 13. 2 2018.] <https://blogs.msdn.microsoft.com/dotnet/2016/09/26/introducing-net-standard/>.

31. **Brockschmidt, Kraig.** An introduction to NuGet. *Microsoft Docs*. [Online] 10. 1 2018. [Citace: 6. 3 2018.] <https://docs.microsoft.com/en-us/nuget/what-is-nuget>.

32. —. Different ways to install a NuGet Package. *Microsoft Docs*. [Online] Microsoft, 12. 2 2018. [Citace: 6. 3 2018.] <https://docs.microsoft.com/en-us/nuget/consume-packages/ways-to-install-a-package>.

33. —. Package consumption workflow. *Microsoft Docs*. [Online] Microsoft, 6. 6 2017. [Citace: 6. 3 2018.] <https://docs.microsoft.com/en-us/nuget/consume-packages/overview-and-workflow>.

34. —. Hosting your own NuGet feeds. *Microsoft Docs*. [Online] Microsoft, 25. 8 2017. [Citace: 6. 3 2018.] <https://docs.microsoft.com/en-us/nuget/hosting-packages/overview>.

35. **Microsoft.** *Microsoft Azure*. [Online] [Citace: 9. 3 2018.] <https://azure.microsoft.com/cs-cz/>.

36. —. Co je Azure? *Microsoft Azure*. [Online] [Citace: 9. 3 2018.] <https://azure.microsoft.com/cs-cz/overview/what-is-azure/>.



37. —. Produkty Azure. *Microsoft Azure*. [Online] [Citace: 9. 3 2018.] <https://azure.microsoft.com/cs-cz/services/>.
38. —. Výkonnost cloudu pro vývojáře. *Microsoft Azure*. [Online] [Citace: 9. 3 2018.] <https://azure.microsoft.com/cs-cz/overview/productivity/>.
39. —. Inovace pro hybridní cloudové aplikace. *Microsoft Azure*. [Online] [Citace: 9. 3 2018.] <https://azure.microsoft.com/cs-cz/overview/hybrid-cloud/>.
40. —. Azure Stack. *Microsoft Azure*. [Online] [Citace: 9. 3 2018.] <https://azure.microsoft.com/en-us/overview/azure-stack/>.
41. —. Jistota v důvěryhodném cloudu. *Microsoft Azure*. [Online] [Citace: 9. 3 2018.] <https://azure.microsoft.com/cs-cz/overview/trusted-cloud/>.
42. **Dunn, Craig**. I use Mobile Services, how does App Service help? *Microsoft Azure*. [Online] 1. 10 2016. [Citace: 10. 3 2018.] <https://docs.microsoft.com/en-us/azure/app-service-mobile/app-service-mobile-value-prop-migration-from-mobile-services>.
43. —. About Mobile Apps in Azure App Service. *Microsoft Azure*. [Online] 1. 10 2016. [Citace: 10. 3 2018.] <https://docs.microsoft.com/en-us/azure/app-service-mobile/app-service-mobile-value-prop>.
44. **Henderson, Matthew**. Authentication and Authorization in Azure Mobile Apps. *Microsoft Docs*. [Online] 1. 10 2016. [Citace: 12. 2 2018.] <https://docs.microsoft.com/en-us/azure/app-service-mobile/app-service-mobile-auth>.
45. **Wasson, Mike**. Basic web application. *Microsoft Azure*. [Online] 12. 12 2017. [Citace: 10. 3 2018.] <https://docs.microsoft.com/en-us/azure/architecture/reference-architectures/app-service-web-app/basic-web-app>.
46. **Lin, Cephas**. Azure App Service plan overview. *Microsoft Azure*. [Online] 9. 11 2017. [Citace: 11. 3 2018.] <https://docs.microsoft.com/en-us/azure/app-service/azure-web-sites-web-hosting-plans-in-depth-overview>.
47. —. Scale up an app in Azure. *Microsoft Azure*. [Online] 5. 7 2016. [Citace: 11. 3 2018.] <https://docs.microsoft.com/en-us/azure/app-service/web-sites-scale>.
48. **Microsoft**. The MVVM Pattern. *MSDN*. [Online] [Citace: 25. 3 2018.] <https://msdn.microsoft.com/en-us/library/hh848246.aspx>.
49. **testyzucetnictvi.cz**. Slovníček účetních pojmů. [Online] [Citace: 11. 3 2018.] <http://www.testyzucetnictvi.cz/slovnicek-ucetnich-pojmu.php?pojem=penezni-toky>.
50. **ALTAXO**. Cash flow - výkaz peněžních toků. [Online] [Citace: 11. 3 2018.] <https://www.altaxo.cz/provoz-firmy/ucetnictvi-a-dane/dane/cash-flow-vykaz-peneznich-toku>.

51. **Valenta, Jan.** Osobní a rodinné finance. *VŠE*. [Online] 30. 4 2008. [Citace: 12. 3 2018.] <https://vskp.vse.cz/id/1221718>.
52. **Školník, Jiří.** Příjmy a výdaje. *Finanční rozpočet*. [Online] [Citace: 15. 3 2018.] [http://www.gymnazium1.milevsko.cz/dokumenty/ekf1/fr/fin\\_rozp.html](http://www.gymnazium1.milevsko.cz/dokumenty/ekf1/fr/fin_rozp.html).
53. **Železný, Jaromír.** Osobní rozpočet, výdaje, příjmy domácnosti. [Online] 2012. [Citace: 15. 3 2018.] [http://www.ssstavji.cz/assets/File.ashx?id\\_org=400032&id\\_dokumenty=3619](http://www.ssstavji.cz/assets/File.ashx?id_org=400032&id_dokumenty=3619).
54. **Gladiš, Daniel.** Jak sestavit osobní finanční plán. *peníze.cz*. [Online] 10. 10 2002. [Citace: 15. 3 2018.] <https://www.penize.cz/investice/15109-jak-sestavit-osobni-financni-plan>.
55. **iPodnikatel.** Finanční plán podnikání jako součást podnikatelského záměru. *iPodnikatel*. [Online] 27. 2 2012. [Citace: 15. 3 2018.] <http://www.ipodnikatel.cz/Podnikatelsky-zamer/financni-plan-podnikani-jako-soucast-podnikatelskeho-zameru.html>.
56. —. Rozvaha – přehled o majetku podniku a zdrojích jeho krytí. *iPodnikatel*. [Online] 26. 3 2012. [Citace: 18. 3 2018.] <http://www.ipodnikatel.cz/Financni-rizeni/rozvaha-prehled-o-majetku-podniku-a-zdrojich-jeho-kryti.html>.
57. —. Rozvaha – přehled o majetku podniku a zdrojích jeho krytí. *iPodnikatel*. [Online] 26. 3 2012. [Citace: 18. 3 2018.] <http://www.ipodnikatel.cz/Financni-rizeni/rozvaha-prehled-o-majetku-podniku-a-zdrojich-jeho-kryti/Struktura-rozvahy-balance.html>.
58. **Microsoft.** Visual Studio IDE. *Visual Studio*. [Online] [Citace: 18. 3 2018.] <https://www.visualstudio.com/vs/>.
59. —. Develop apps for the Universal Windows Platform (UWP). *MSDN*. [Online] [Citace: 18. 3 2018.] <https://msdn.microsoft.com/en-us/library/dn975273.aspx>.
60. —. Visual Studio 2015 – požadavky na systém. *Microsoft Docs*. [Online] 24. 7 2017. [Citace: 18. 3 2018.] <https://docs.microsoft.com/cs-cz/visualstudio/productinfo/vs2015-sysrequirements-vs>.
61. —. Produktová řada Visual Studio 2017 – požadavky na systém. *Microsoft Docs*. [Online] 30. 9 2017. [Citace: 18. 3 2018.] <https://docs.microsoft.com/cs-cz/visualstudio/productinfo/vs2017-system-requirements-vs>.
62. **Hughes, Lauren.** How Visual Studio generates an app package manifest. *Microsoft Docs*. [Online] 29. 11 2017. [Citace: 18. 3 2018.] <https://docs.microsoft.com/en-us/uwp/schemas/appxpackage/uapmanifestschema/generate-package-manifest>.
63. —. App package manifest. *Microsoft Docs*. [Online] 8. 7 2017. [Citace: 18. 3 2018.] <https://docs.microsoft.com/en-us/uwp/schemas/appxpackage/appx-package-manifest>.

64. **Satran, Michael.** App capability declarations. *Microsoft Docs*. [Online] 26. 10 2017. [Citace: 18. 3 2018.] <https://docs.microsoft.com/en-us/windows/uwp/packaging/app-capability-declarations>.
65. **SQLite.** About SQLite. *SQLite*. [Online] [Citace: 20. 3 2018.] <https://www.sqlite.org/about.html>.
66. —. SQLite is Transactional. *SQLite*. [Online] [Citace: 25. 3 2018.] <https://www.sqlite.org/transactional.html>.
67. **SQLite Tutorial.** What Is SQLite. *SQLite Tutorial*. [Online] [Citace: 25. 3 2018.] <http://www.sqlitetutorial.net/what-is-sqlite/>.
68. **Bheda, Romil.** How To Add Azure Mobile App To An UWP App. *C# Corner*. [Online] 29. 8 2016. [Citace: 19. 3 2018.] <https://www.c-sharpcorner.com/article/how-to-add-azure-mobile-app-to-an-uwp-app/>.
69. **Henderson, Matthew.** How to configure your App Service application to use Microsoft Account login. *Microsoft Docs*. [Online] 1. 10 2016. [Citace: 2. 4 2018.] <https://docs.microsoft.com/en-us/azure/app-service/app-service-mobile-how-to-configure-microsoft-authentication>.
70. **Dunn, Craig.** Offline Data Sync in Azure Mobile Apps. *Microsoft Azure*. [Online] 30. 10 2016. [Citace: 19. 3 2018.] <https://docs.microsoft.com/en-us/azure/app-service-mobile/app-service-mobile-offline-data-sync>.
71. **Whitney, Tyler.** Support your app with background tasks. *Microsoft Docs*. [Online] 21. 8 2017. [Citace: 25. 3 2018.] <https://docs.microsoft.com/en-us/windows/uwp/launch-resume/support-your-app-with-background-tasks>.
72. **Metulev, Nikola.** Background Task Helper. *Microsoft Docs*. [Online] 28. 2 2018. [Citace: 25. 3 2018.] <https://docs.microsoft.com/en-us/windows/uwpcommunitytoolkit/helpers/backgroundtaskhelper>.
73. **Kolektiv.** ToastNotificationActionTrigger Class. *Microsoft Docs*. [Online] [Citace: 25. 3 2018.] <https://docs.microsoft.com/en-us/uwp/api/windows.applicationmodel.background.toastnotificationactiontrigger>.
74. **Microsoft.** JumpList Class. *Microsoft Docs*. [Online] [Citace: 26. 3 2018.] <https://docs.microsoft.com/en-us/uwp/api/windows.ui.startscreen.jumplist>.
75. **Estabrook, Norm.** Manage appointments. *Microsoft Docs*. [Online] 8. 2 2018. [Citace: 27. 3 2018.] <https://docs.microsoft.com/en-us/windows/uwp/contacts-and-calendar/managing-appointments>.

76. **Jacobs, Mike.** Store and retrieve settings and other app data. *Microsoft Docs*. [Online] 14. 11 2017. [Citace: 25. 3 2018.] <https://docs.microsoft.com/en-us/windows/uwp/design/app-settings/store-and-retrieve-app-data>.

77. **Microsoft.** Specifikace a požadavky na systém Windows 10. *Windows*. [Online] [Citace: 26. 3 2018.] <https://www.microsoft.com/cs-cz/windows/windows-10-specifications>.

78. **Kennedy, John.** Enable your device for development. *Windows Dev Center*. [Online] Microsoft, 12. 3 2017. [Citace: 15. 2 2018.] <https://docs.microsoft.com/en-gb/windows/uwp/get-started/enable-your-device-for-development>.

79. **Technopedia.** Operating System (OS). *Technopedia*. [Online] [Citace: 21. 3 2018.] <https://www.techopedia.com/definition/3515/operating-system-os>.

80. **ITBIZ.** API. *ITBIZ*. [Online] [Citace: 2. 4 2018.] <http://www.itbiz.cz/slovník/informacni-technologie-it/api>.

81. **Microsoft.** Windows 10 SDK. *Windows Dev Center*. [Online] [Citace: 20. 3 2018.] <https://developer.microsoft.com/en-us/windows/downloads/windows-10-sdk>.

82. **Jacobs, Mike.** Tiles, badges, and notifications for UWP apps. *Microsoft Docs*. [Online] 19. 5 2017. [Citace: 20. 3 2018.] <https://docs.microsoft.com/en-us/windows/uwp/design/shell/tiles-and-notifications/>.

83. **Brown, Eric.** Who Needs the Internet of Things? *Linux.com*. [Online] 13. 9 2016. [Citace: 20. 3 2018.] <https://www.linux.com/news/who-needs-internet-things>.

84. **ITU.** Internet of Things Global Standards Initiative. [Online] [Citace: 20. 3 2018.] <https://www.itu.int/en/ITU-T/gsi/iot/Pages/default.aspx>.

85. **Hanselman, Scott.** Managing dotnet Core 2.0 and dotnet Core 1.x versioned SDKs on the same machine. *Scott Hanselman*. [Online] 12. 6 2017. [Citace: 2018. 3 20.] <https://www.hanselman.com/blog/ManagingDotnetCore20AndDotnetCore1xVersionedSDKsOnTheSameMachine.aspx>.

86. **Wagner, Bill.** Interoperability (C# Programming Guide). *Microsoft Docs*. [Online] 20. 7 2015. [Citace: 21. 3 2018.] <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/interop/>.

87. **Microsoft.** Application Startup Time. *Microsoft Docs*. [Online] 30. 3 2017. [Citace: 21. 3 2018.] <https://docs.microsoft.com/en-us/dotnet/framework/wpf/advanced/application-startup-time>.

88. —. Container Instances. *Microsoft Azure*. [Online] [Citace: 21. 3 2018.] <https://azure.microsoft.com/cs-cz/services/container-instances/>.

89. —. Azure Files. *Microsoft Azure*. [Online] [Citace: 21. 3 2018.] <https://azure.microsoft.com/cs-cz/services/storage/files/>.
90. —. Blob Storage. *Microsoft Azure*. [Online] [Citace: 21. 3 2018.] <https://azure.microsoft.com/cs-cz/services/storage/blobs/>.
91. —. VPN Gateway. *Microsoft Azure*. [Online] [Citace: 20. 3 2018.] <https://azure.microsoft.com/cs-cz/services/vpn-gateway/>.
92. —. Application Gateway. *Microsoft Azure*. [Online] [Citace: 20. 3 2018.] <https://azure.microsoft.com/cs-cz/services/application-gateway/>.
93. —. What is cloud computing? *Microsoft Azure*. [Online] [Citace: 21. 3 2018.] <https://azure.microsoft.com/en-us/overview/what-is-cloud-computing/>.
94. **M-Files**. Deployment: On-premises, Cloud, or Hybrid. *M-Files*. [Online] [Citace: 21. 3 2018.] <https://www.m-files.com/en/cloud-on-premise-hybrid>.
95. **Kolektiv**. Azure WebJobs SDK. *GitHub*. [Online] 16. 3 2018. [Citace: 21. 3 2018.] <https://github.com/Azure/azure-webjobs-sdk>.
96. **Dostálová, Zuzana**. Frontend vs. Backend. *Czechitas*. [Online] 1. 7 2014. [Citace: 21. 3 2018.] <https://www.czechitas.cz/cs/blog/web/frontend-vs-backend>.
97. **IT Slovník**. log. *IT Slovník*. [Online] [Citace: 21. 3 2018.] <https://it-slovník.cz/pojem/log>.
98. **Pittet, Sten**. Continuous integration vs. continuous delivery vs. continuous deployment. *Atlassian*. [Online] [Citace: 21. 3 2018.] <https://www.atlassian.com/continuous-delivery/ci-vs-ci-vs-cd>.
99. **ManagementMania**. SWOT analýza. *ManagementMania*. [Online] [Citace: 21. 3 2018.] <https://managementmania.com/cs/swot-analyza>.
100. —. Vlastní kapitál (Equity). *ManagementMania*. [Online] [Citace: 21. 3 2018.] <https://managementmania.com/cs/vlastni-kapital-jmeni>.
101. —. Cizí kapitál, cizí zdroje, závazky (Liabilities, Liability). *ManagementMania*. [Online] [Citace: 21. 3 2018.] <https://managementmania.com/cs/cizi-zdroje-kapital>.
102. **Finance**. Co je to účetní závěrka? *Finance*. [Online] 20. 11 2013. [Citace: 21. 3 2018.] <https://firmy.finance.cz/zpravy/finance/243715-co-je-to-ucetni-zaverka/>.
103. **Satran, Michael**. Windows Runtime components. *Microsoft Docs*. [Online] 8. 2 2017. [Citace: 20. 3 2018.] <https://docs.microsoft.com/en-us/windows/uwp/winrt-components/>.
104. **Zatetic**. Full Database Encryption for SQLite. [Online] [Citace: 20. 3 2018.] <https://www.zetetic.net/sqlcipher/>.

105. **Malý, Martin.** REST: architektura pro webové API. *Zdroják*. [Online] 3. 8 2009. [Citace: 20. 3 2018.] <https://www.zdrojak.cz/clanky/rest-architektura-pro-webove-api/>.
106. **Microsoft.** Data Transfer Object. *MSDN*. [Online] [Citace: 2. 4 2018.] <https://msdn.microsoft.com/en-us/library/ff649585.aspx>.
107. **Sharma, Munesh.** POCO Classes in Entity Framework. *C# Corner*. [Online] 2. 3 2015. [Citace: 2. 4 2018.] <https://www.c-sharpcorner.com/UploadFile/5d065a/poco-classes-in-entity-framework/>.
108. **Janovský, Dušan.** URL. *Jak psát web*. [Online] [Citace: 2. 4 2018.] <https://www.jakpsatweb.cz/html/url.html>.
109. **Clayton, Steve.** Modern design at Microsoft. *Microsoft*. [Online] [Citace: 8. 3 2018.] <https://www.microsoft.com/en-us/stories/design/>.

## 9. Slovník pojmů

- OS
  - Základní programové vybavení počítače, které umožňuje spouštět aplikace, dále spravuje HW zdroje zařízení. (79)
- API
  - Jedná se o sadu procedur, funkcí nebo tříd. (80)
- SDK
  - SDK je sada knihoven, metadat a nástrojů pro vývoj aplikací. (81)
- Toast notifikace
  - Vyskakovací okno s textem a slouží pro informování uživatele. (82)
- Interaktivní notifikace
  - Jedná se o toast notifikaci, ale obsah je doplněn o ovládací prvky pro interakci. (82)
- IoT
  - IoT je infrastruktura, která propojuje fyzicky i virtuálně věci, například droni, roboti a další. (83) (84)
- side-by-side instalace
  - Instalace rozdílných verzí .NET Core za účelem vývoje aplikací a jejich testování. (85)
- Interoperabilita
  - Využití nespravovaného kódu (neběží pod správou CLR, například Win32 API, COM a další) v rámci spravovaného kódu. (86)
- Cold startup
  - Spuštění aplikace poprvé po spuštění OS nebo po vypnutí aplikace a po dlouhé době jí znovu spustit. (87)
- Warm startup
  - Spuštění aplikace v době, kdy většina stránek CLR je načtena v operační paměti. (87)
- Azure Container Instances
  - Služba pro spuštění kontejnerů v Azure bez nutnosti nové orchestrace a programovacího modelu. (88)
- Azure File Storage
  - Uložiště spravovaných a sdílených složek v cloudu. (89)

- Azure Blob Storage
  - Uložiště objektů pro nestruturovaná data, které se škáluje automaticky podle požadavků. (90)
- Azure VPN Gateway
  - Jedná se o službu v rámci Microsoft Azure, která slouží pro propojení lokální sítě k Azure. (91)
- Azure Application Gateway
  - Je služba Microsoft Azure, jež poskytuje služby pro vyrovnání zátěže aplikací nebo také WAF. (92)
- On-premises
  - Cloud řešení, které běží na serverech organizace, a ne na serverech poskytovatele cloudových služeb. (93) (94)
- Cloud computing
  - Poskytování výpočetních služeb, jako uložště, databáze nebo software a další přes cloud. (93)
- PaaS
  - Jedná se o službu, která poskytuje prostředí pro vývoj, testování nebo distribuci softwaru. (93)
- Web Jobs
  - Služba, která umožňuje spustit program či skript, jako například procesy na pozadí nebo služby v rámci webové aplikace. (95)
- Backend
  - Technologie, které běží na serveru a uživatel je nevidí. (96)
- Log
  - Záznam činnosti, který je uložen v nějakém souboru (tzv. log file). (97)
- Continuous deployment
  - Obdoba *Continuous delivery*, ale s rozdílem, že všechny změny, jež prošli všemi částmi produkce, budou vydány zákazníkům. (98)
- SWOT analýza
  - Analytická metoda pro ohodnocení vnitřních a vnějších ovlivňujících úspěšnost organizace nebo konkrétní úmysl. (99)
- Vlastní kapitál
  - Kapitál, jež vlastní majitel či majitelé. (100)



- Cizí kapitál
  - Dluh podniku nebo organizace, jež musí být splacen v určené době. (101)
- Účetní závěrka
  - Účetní práce s cílem sestavení účetních výkazů a uzavřením jednotlivých účtů za určité období. (102)
- Windows Runtime komponenty
  - Jsou nezávislé objekty, které lze napsat v C#, JS, VB nebo C++. (103)
- SQLCipher
  - Rozšíření pro SQLite pro zašifrování algoritmem AES databázových souborů. (104)
- REST
  - Jedná se o architekturu rozhraní, jež bylo navrženo v roce 2000 pro distribuované prostředí. (105)
- DTO
  - Objekt, který udržuje data při přenosu. (106)
- POCO
  - Jedná se o objekt, jež není závislý na žádné framework třídě. (107)
- URL
  - Jedná se o unikátní určení zdroje. (108)

## 10. Seznam použitých zkratk

Zkratka	Nezkrácený tvar	V českém jazyce
UWP	Universal Windows Platform	Univerzální platforma windows
UAP	Universal Application Platform	Univerzální aplikační platforma
PPI	Pixel per inch	Pixel na palec
OS	Operating system	Operační systém
UWA	Universal Windows Application	Univerzální windows aplikace
API	Application Programming Interface	Aplikační programové rozhraní
SDK	Software Development Kit	Sada pro vývoj softwaru
ARM	Advanced RISC Machine	Advanced RISC Machine
REST	Representational State Transfer	Representational State Transfer
IDP	Identity Provider	Poskytovatel identit
CSV	Comma-separated values	Hodnoty oddělné čárkami
HTML	HyperText Markup Language	HyperText značkovací jazyk
PDF	Portable Document File	Portable Document File
VB	Visual Basic	Visual Basic
CLR	Common Language Runtime	Common Language Runtime
CIL	Common Intermediate Language	Common Intermediate Language
HW	Hardware	Hardware
IoT	Internet of Things	Internet věcí
IL	Intermediate Language	Intermediate Language
AOT	Ahead of Time	Ahead of Time
JIT	Just in Time	Just in Time
NGEN	Native Image Generator	Native Image Generator
VS	Visual Studio	Visual Studio

CLI	Command Line Interface	Příkazový řádek
UI	User Interface	Uživatelské rozhraní
VSTS	Visual Studio Team Services	Visual Studio Team Services
epx	Effective pixels	Efektivní pixely
MDL	Microsoft Design Language	Microsoft Design Language
CDN	Content Delivery Network	Sít' pro doručování obsahu
PaaS	Platform as a Service	Platforma jako služba
AAD	Azure Active Directory	Azure Active Directory
CD	Continuous deployment	Průběžné nasazení
IDE	Integrated Development Environment	Integrované vývojové prostředí
JS	Javascript	Javascript
WAF	Web Application Firewall	Firewall webových aplikací
AES	Advanced Encryption Standard	Standard pro pokročilé šifrování
SID	Security ID	Bezpečnostní ID
MVVM	Model-View-ViewModel	Model-View-ViewModel
DTO	Data Transfer Object	Data Transfer Object
POCO	Plain Old CLR Object	Plain Old CLR Object
SID	Security Identifier	Bezpečnostní identifikátor
URL	Uniform Resource Locator	Unikátní ukazatel na zdroj

Tabulka 11 - Tabulka se zkratkami

## 11. Seznam obrázků

Obrázek 1 - Přehled UWP .....	4
Obrázek 2 - Ukázka škálování fontu .....	5
Obrázek 3 - Přehled rodin zařízení .....	6
Obrázek 4 - Ukázka aplikace implementující "Fluent Design System" .....	7
Obrázek 5 - Ukázka grafického stylu <i>Metro</i> .....	7
Obrázek 6 - Přehled platformy .NET.....	8
Obrázek 7 - Statistika kódu v CoreFX .....	11
Obrázek 8 - Výsledek implementování .NET Standard v .NET platformě .....	13
Obrázek 9 - Shrnutí API v .NET Standard 2.0 .....	14
Obrázek 10 - Proces tvorby, publikování a distribuce NuGet balíčků .....	16
Obrázek 11 - Proces přijímání NuGet balíčků.....	17
Obrázek 12 - Mobile Apps – platforma pro vývoj aplikací.....	19
Obrázek 13 - Ukázka architektury webové aplikace .....	20
Obrázek 14 - Ukázka částí vzoru MVVM.....	21
Obrázek 15 - Ukázka osobního cashflow .....	23
Obrázek 16 - Proces sestavení finančního plánu .....	26
Obrázek 17 - Ukázka osobní rozvahy.....	27
Obrázek 18 - Přehled projektů a složek v rámci řešení .....	29
Obrázek 19 - Diagram tabulek v lokální databázi .....	35
Obrázek 20 - Diagram tabulek v databázi na serveru.....	36
Obrázek 21 - Vytvoření zdroje Mobile Apps v portálu Azure .....	37
Obrázek 22 - Blade pro vytvoření služby Mobile App.....	38
Obrázek 23 - Odkazy na tabulky a jejich konfiguraci .....	38
Obrázek 24 - Nastavení datového připojení .....	39
Obrázek 25 - Vytvoření SQL serveru.....	39
Obrázek 26 - Aplikace v Microsoft Account Dev Center .....	40
Obrázek 27 - Stránka s konfigurací autentizace uživatele.....	40
Obrázek 28 - Nastavení deklarace <i>Protokol</i> v manifestu .....	41
Obrázek 29 - JumpList aplikace <i>Finance Curator</i> .....	46
Obrázek 30 - Okno pro přidání upozornění.....	47
Obrázek 31 - Okno pro vybrání kontaktu .....	47
Obrázek 32 - Ukázka toast notifikace.....	48

Obrázek 33 - Ukázka interaktivní notifikace .....	48
Obrázek 34 - Sekce <i>Aktualizace a zabezpečení</i> v nastavení operačního systému .....	51
Obrázek 35 - Podsekce <i>Pro vývojáře</i> v nastavení operačního systému.....	51
Obrázek 36 – Možnosti nastavení vývojářských funkcí na zařízení .....	52
Obrázek 37 - Aplikační balíček ve složce FinanceCuratorAppPackage .....	53
Obrázek 38 - Instalační program pro aplikaci .....	53
Obrázek 39 - Spuštění instalace bezpečnostního certifikátu .....	54
Obrázek 40 - Okno s tlačítkem pro přihlášení uživatele.....	55
Obrázek 41 - Formulář pro přihlášení pomocí účtu Microsoft.....	55
Obrázek 42 - Celkový přehled financí .....	56
Obrázek 43 - Nastavení rozmezí data pro grafy .....	57
Obrázek 44 - Přehled financí v konkrétní peněžence .....	57
Obrázek 45 - Přehled financí v seznamu .....	58
Obrázek 46 - Okno pro nastavení parametrů pro řazení seznamu financí.....	58
Obrázek 47 - Seznam přidáných peněženek a tlačítko pro přidání penženky .....	59
Obrázek 48 - Okno pro přidávání peněženek .....	59
Obrázek 49 - Upozornění o úspěšnosti přidání penženky.....	59
Obrázek 50 - Tlačítko pro navigaci na stránku s financemi v peněžence .....	60
Obrázek 51 - Tlačítko pro přidání finance do penženky.....	60
Obrázek 52 - Okno pro přidání příjmu .....	61
Obrázek 53 - Přehled rozpočtů .....	61
Obrázek 54 – Tlačítko pro přidání rozpočtu .....	62
Obrázek 55 - Okno pro přidání rozpočtu .....	62
Obrázek 56 - Tlačítko pro přidání účastníka a možnosti přidání.....	63
Obrázek 57 - Okno pro přidání účastníka .....	63
Obrázek 58 - Přehled kategorií v aplikaci .....	64
Obrázek 59 - Okno pro přidání kategorie .....	64
Obrázek 60 - Odkaz na stránku s nastavením aplikace .....	65
Obrázek 61 - Nastavení motivu aplikace .....	65
Obrázek 62 - Nastavení notifikací .....	66
Obrázek 63 - Nastavení symbolu zobrazované měny.....	66
Obrázek 64 - Nastavení synchronizace na pozadí .....	66
Obrázek 65 - Okno pro export dat aplikace .....	67

## 12. Seznam tabulek

Tabulka 1 - Přehled tříd velikostí .....	6
Tabulka 2 - Minimální HW požadavky od verze 4.5 .....	9
Tabulka 3 - Podporované operační systémy v rámci .NET Core 2.0 .....	10
Tabulka 4 - Přehled kompatibility vybraných platforem s verzí .NET Standard .....	13
Tabulka 5 - Nástroje systému NuGet a jejich dostupnost.....	15
Tabulka 6 - Minimální požadavky pro zpracování zdrojových souborů projektu.....	28
Tabulka 7 - Minimální požadavky na VS 2015 .....	28
Tabulka 8 - Minimální požadavky na VS 2017 .....	28
Tabulka 9 - Minimální hardwarové požadavky .....	50
Tabulka 10 - Minimální softwarové požadavky .....	50
Tabulka 11 - Tabulka se zkratkami .....	83

## 13. Přílohy

```

1. private void RegisterBackgroundTasks()
2. {
3.     BackgroundTaskRegistration notificationBT = BackgroundTaskHelper.Register("Notificat
   ionBackgroundTask", new ToastNotificationActionTrigger());
4.     BackgroundTaskRegistration notificationHistoryBT = BackgroundTaskHelper.Register("No
   tificationHistoryBackgroundTask", "RuntimeComponents.NotificationHistoryBackgroundTask",
   new ToastNotificationHistoryChangedTrigger());
5.     BackgroundTaskRegistration reminderBT = BackgroundTaskHelper.Register("ReminderBackg
   oundTask", new TimeTrigger(60, false));
6.     BackgroundTaskRegistration syncBT = BackgroundTaskHelper.Register("SyncBackgroundTas
   k", new TimeTrigger(15, false), false, true, new SystemCondition(SystemConditionType.Int
   ernetAvailable));
7.     BackgroundTaskRegistration permPaymentBT = BackgroundTaskHelper.Register("PermanentP
   aymentBackgroundTask", new TimeTrigger(1440, false));
8. }

```

### Příloha 1

Zdroj: vlastní

```

1. public async void Run(IBackgroundTaskInstance taskInstance)
2. {
3.     ToastNotificationActionTriggerDetail details = (ToastNotificationActionTriggerDetail
   )taskInstance.TriggerDetails;
4.
5.     if (details != null)
6.     {
7.         string arguments = details.Argument;
8.         string financeName = (string)details.UserInput["financeName"];
9.         string financeValue = (string)details.UserInput["financeValue"];
10.        string financeType = (string)details.UserInput["financeType"];
11.        Wallet defaultWallet = await WalletViewModel.Instance().GetDefaultWallet();
12.
13.        if (ValidateInputs(financeName, financeValue, financeType))
14.        {
15.            switch (financeType)
16.            {
17.                case "1":
18.                    Income income = new Income() { Name = financeName, Value = Convert.T
   oDouble(financeValue), WalletID = defaultWallet.Id, Added = DateTime.Now };
19.                    await IncomeViewModel.Instance().SaveEntity(income);
20.                    defaultWallet.Sum += income.Value;
21.                    break;
22.
23.                case "2":
24.                    Expense expense = new Expense() { Name = financeName, Value = Conver
   t.ToDouble(financeValue), WalletID = defaultWallet.Id, Added = DateTime.Now };
25.                    await ExpenseViewModel.Instance().SaveEntity(expense);
26.                    defaultWallet.Sum -= expense.Value;
27.                    break;
28.
29.                default:
30.                    break;
31.            }
32.            await WalletViewModel.Instance().UpdateEntity(defaultWallet);
33.        }
34.    }
35. }

```

### Příloha 2

Zdroj: vlastní

```
1. public void Run(IBackgroundTaskInstance taskInstance)
2. {
3.     ToastNotificationHistoryChangedTriggerDetail details = (ToastNotificationHistoryChan
4.         gedTriggerDetail)taskInstance.TriggerDetails;
5.     bool isInteractive = false;
6.     if (ApplicationData.Current.LocalSettings.Values["isInteractive"] != null)
7.     {
8.         isInteractive = (bool)ApplicationData.Current.LocalSettings.Values["isInteractiv
9.             e"];
10.    }
11.    if (details != null && ApplicationData.Current.LocalSettings.Values["notificationSet
12.        ting"].Equals(true) && isInteractive)
13.    {
14.        switch (details.ChangeType)
15.        {
16.            case ToastHistoryChangedType.Cleared:
17.                SendInteractiveNotification();
18.                break;
19.            case ToastHistoryChangedType.Removed:
20.                SendInteractiveNotification();
21.                break;
22.            case ToastHistoryChangedType.Expired:
23.                SendInteractiveNotification();
24.                break;
25.            default:
26.                break;
27.        }
28.        ApplicationData.Current.LocalSettings.Values.Remove("isInteractive");
29.    }
30. }
```

### Příloha 3

Zdroj: vlastní



```
1. public async void Run(IBackgroundTaskInstance taskInstance)
2. {
3.     bool reminderSetting = (bool)ApplicationData.Current.LocalSettings.Values["remindersSetting"];
4.     if (reminderSetting)
5.     {
6.         DateTime maxDate = DateTime.Now.AddMinutes(59);
7.         DateTime alteredEntityDueDate;
8.         List<Debt> debts = await DebtViewModel.Instance().GetEntitesAsList();
9.         if (debts.Count > 0)
10.        {
11.            List<Debt> filteredDebts = debts.Where(i => i.DueDate >= DateTime.Now && i.IsPaid.Equals(false)).ToList();
12.            debts.Clear();
13.
14.            //Searching for valid debts.
15.            foreach (Debt debt in filteredDebts)
16.            {
17.                alteredEntityDueDate = new DateTime(debt.DueDate.Year, debt.DueDate.Month, debt.DueDate.Day, debt.DueDate.Hour, debt.DueDate.Minute, debt.DueDate.Second);
18.                if (alteredEntityDueDate <= maxDate)
19.                {
20.                    debts.Add(debt);
21.                }
22.            }
23.
24.            //Sending debt due date reminder notification.
25.            if (debts.Count == 1)
26.            {
27.                NotificationService.Instance().SendInfoNotification(
28.                    ResourceLoaderHelper.GetResourceLoader().GetString("DebtDueDateNotificationTitle"), ${ResourceLoaderHelper.GetResourceLoader().GetString("DebtNotificationPart1")} {debts.First().Name} {ResourceLoaderHelper.GetResourceLoader().GetString("DebtNotificationPart2")} {debts.First().Value} {App.CurrencySymbol}"
29.                );
30.            }
31.            else if (debts.Count > 1)
32.            {
33.                NotificationService.Instance().SendInfoNotification(
34.                    ResourceLoaderHelper.GetResourceLoader().GetString("DebtDueDateNotificationTitle"), ${ResourceLoaderHelper.GetResourceLoader().GetString("DebtNotificationPart1")} {debts.Count} {ResourceLoaderHelper.GetResourceLoader().GetString("DebtsNotificationPart3")} {debts.First().Value} {App.CurrencySymbol} {ResourceLoaderHelper.GetResourceLoader().GetString("Debt").ToLower()} {debts.First().Name}"
35.                );
36.            }
37.        }
38.    }
39. }
```

#### Příloha 4

Zdroj: vlastní

```
1. public async void Run(IBackgroundTaskInstance taskInstance)
2. {
3.     bool backgroundSyncSetting = (bool)ApplicationData.Current.LocalSettings.Values["backgroundSyncSetting"];
4.     if (backgroundSyncSetting)
5.     {
6.         await AzureService.Instance().SyncAsync();
7.     }
8. }
```

#### Příloha 5

Zdroj: vlastní

```
1. public async void Run(IBackgroundTaskInstance taskInstance)
2. {
3.     DateTime maxDate = DateTime.Now.AddDays(1);
4.     List<PermanentPayment> permanentPayments = await PermanentPaymentViewModel.Instance().GetEntitiesAsList();
5.     if (permanentPayments.Count > 0)
6.     {
7.         permanentPayments = permanentPayments.Where(i => i.DueDate <= maxDate && i.DueDate >= Convert.ToDateTime(DateTime.Now.ToString("d"))).ToList();
8.         Wallet wallet;
9.         foreach (PermanentPayment payment in permanentPayments)
10.        {
11.            wallet = await WalletViewModel.Instance().GetEntityByID(payment.WalletID);
12.            wallet.Sum -= payment.Value;
13.            ChangeDueDate(payment);
14.            await PermanentPaymentViewModel.Instance().UpdateEntity(payment);
15.        }
16.    }
17. }
```

#### Příloha 6

Zdroj: vlastní