Mendelova univerzita v Brně
Provozně ekonomická fakulta

Mobilní aplikace pro inventuru majetku

Diplomová práce

Vedoucí práce:
Ing. Jan Přichystal, Ph.D.

Bc. Pavel Fiala

Brno 2016

Na tomto místě bych rád poděkoval svému vedoucímu práce Ing. Janu Přichystalovi, Ph.D. za ochotu při vedení této práce a poskytování cenných rad. Dále bych chtěl poděkovat přátelům a kolegům z firmy ALVAO, kteří mi vždy pohotově poradili a případně doporučili řešení problému. A velké poděkování patří také mojí přítelkyni za věčnou podporu. Díky této podpoře a všem cenným radám se podařilo celou práci úspěšně dokončit.

Čestné prohlášení

Prohlašuji, že jsem tuto práci: **Mobilní aplikace pro inventuru majetku** vypracoval samostatně a veškeré použité prameny a informace jsou uvedeny v seznamu použité literatury. Souhlasím, aby moje práce byla zveřejněna v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů, a v souladu s platnou *Směrnicí o zveřejňování vysokoškolských závěrečných prací.*

Jsem si vědom, že se na moji práci vztahuje zákon č. 121/2000 Sb., autorský zákon, a že Mendelova univerzita v Brně má právo na uzavření licenční smlouvy a užití této práce jako školního díla podle § 60 odst. 1 Autorského zákona.

Dále se zavazuji, že před sepsáním licenční smlouvy o využití díla jinou osobou (subjektem) si vyžádám písemné stanovisko univerzity o tom, že předmětná licenční smlouva není v rozporu s oprávněnými zájmy univerzity, a zavazuji se uhradit případný příspěvek na úhradu nákladů spojených se vznikem díla, a to až do jejich skutečné výše.

Brno 30.12.2016 .....

Abstract

Fiala, P. *Mobile application for assets stocktaking*. Master thesis. Brno: Mendel University in Brno, 2016.

Master thesis deals with assets stocktaking using mobile application at an environment of a company. The theoretical part analyzes the current mobile application for stocktaking and available development technologies. And also describes stocktaking problem in general. The practical part focuses on selection of appropriate development technology. Next the part focuses on the design and implementation of universal mobile application. The mobile application is implemented using the C# programming language and the .NET Framework. Finally the conclusions and recommendations are deduced based on evaluating and discussing of results.

Keywords

Inventura, technologie vývoje, univerzální mobilní aplikace, snímání čárového kódu, C#, Framework .NET.

Abstrakt

Fiala, P. *Mobilní aplikace pro inventuru majetku*. Diplomová práce, Brno: Mendelova univerzita v Brně, 2016

Diplomová práce se zabývá problematikou inventury a inventarizace majetku v prostředí společnosti pomocí mobilní aplikace. V teoretické části práce je rozebrána současná mobilní aplikace pro inventuru majetku, jsou zanalyzovány dostupné technologie vývoje a také popsán problém provádění inventury obecně. Praktická část práce je nejprve zaměřena na výběr vhodné technologie vývoje, dále na návrh vlastního řešení a implementaci univerzální mobilní aplikace, která slouží k provádění inventury majetku. Mobilní aplikace je implementována pomocí .NET Frameworku a programovacího jazyka C#. Na závěr jsou výsledky vyhodnoceny a diskutovány, na základě čehož jsou dedukovány závěry a doporučení.

Klíčová slova

Stocktaking, development technology, universal mobile application, scan bar code, C#, Framework .NET.

Obsah

1 Úvod a cíl práce

1.1 Úvod

Základem každé společnosti je její majetek, který ji mimo jiné podporuje v úspěšném a efektivním podnikání. Často tento majetek bývá velice rozsáhlý a je nutné jej nějakým způsobem evidovat, popřípadě spravovat. V současné době moderních technologií není výjimkou ani elektronická forma evidence či správy majetku, k čemuž slouží typ softwaru, který má v názvu často obsaženo slovní spojeni *Asset management*. Aby mohl tento software efektivně fungovat a reálně znázorňoval obraz majetku společnosti musí být daná evidence pravidelně revidována a aktualizována podle skutečného stavu, k tomu slouží inventura majetku.

Proces inventury majetku probíhá v různých společnostech různě a i její kvalita provedení se může často značně lišit. Jedním jejím typem je inventura fyzická, která se provádí u majetku hmotné povahy a v případě rozsáhlého majetku může být ve všech ohledech velice náročná, zejména pak provádění inventury manuálně bez pomocí jakéhokoliv elektronického zařízení. Mimo jiné z tohoto důvodu pronikly moderní technologie i právě do tohoto procesu. Dnes se již jako základ využívá tzv. elektronických čteček čárových kódu, pomocí kterých se majetek fyzicky eviduje, tedy z majetku je načten čárový kód, který se pak porovnává z evidencí.Nicméně každý asi cítí, že provádění inventury pomocí těchto mobilních čteček stále plně nevyužívá současnou úroveň vývoje moderních technologií.

Rychlost vývoje těchto technologií stále roste a zároveň jejich společným cílem je schopnost přizpůsobit se potřebám člověka. Za možného zástupce těchto technologií lze považovat tzv. chytrá zařízení, která lze potkávat dnes a denně všude kolem nás. Tato technologie proniká do všech možných lidských činností a potřeb. Často může nastávat situace, kdy určitý jedinec používá několik chytrých zařízení, každé v rámci jiné činnosti. Pak se jistě naskytá otázka, zda by nebylo vhodnější, používat jedno chytré zařízení na vše. Tuto cestu vývoje by bylo vhodné využít i pro zmiňovanou inventuru majetku a tedy její provádění realizovat pomocí mobilní aplikace pro chytré zařízení (mobilní telefon nebo tablet), kterou se dále zabývá i tato diplomová práce.

1.2 Cíl práce

Cílem diplomové práce je navrhnout vlastní řešení pro mobilní aplikaci sloužící k inventarizaci majetku. Na základě tohoto návrhu poté provést implementaci daného řešení univerzální Windows 10 aplikace.

K dosažení cíle je nutné zanalyzovat aktuální stav inventarizace v dané společnosti a také se seznámit se současnou aplikací určenou pro inventarizaci. Dále je nutné provést kompletní analýzu dostupných technologií a jejich klíčových vlastností. Poté je potřeba popsat problematiku inventarizace majetku, zásady provedení

a také strukturou majetku. Nakonec je třeba definovat uživatelské scénáře užití dané aplikace.

Pro implementaci vlastního řešení je nutné vybrat z dostupných a používaných technologií na základě jejich pozitivních a negativních vlastností, tu vhodnou, také s ohledem na situaci v dané společnosti. Poté vyhodnotit vlastní řešení a diskutovat dosažení daného cíle.

1.3 Metodika řešení

Prvním krokem vypracování této diplomové práce bude důkladné seznámení ze současnou aplikací pro inventuru majetku a analýza klíčových vlastností a nedostatků, které obsahuje. K tomuto účelu bude aplikace přímo prakticky prozkoumána a prostudována její technická dokumentace. Dále využiji také osobní rozhovor s vývojáři stávající aplikace a interními pracovníky společnosti, kteří mi dokážou lépe definovat důležité vlastnosti, které je potřeba zachovat, dále její slabá místa a potenciální nedostatky. Na základě těchto rozhovorů budou také později přesně specifikovány nejčastější uživatelské scénáře použití aplikace, které budou základem při návrhu vlastního řešení.

V dalším kroku je třeba provést analýzu prostředí – rozebrat jednotlivé používané dostupné technologie vývoje a také se seznámit se strukturou majetku k inventarizaci. K tomu bude využito dostupných literárních publikací, odborných článků a také několik internetových zdrojů. Na základě této analýzy bude později vybrána vhodná technologie pro vývoj požadované mobilní aplikace pro inventuru majetku.

V následujícím kroku bude navrženo vlastní řešení mobilní aplikace pro inventuru majetku. K tomu bude využito objektově orientované analýzy, tedy diagramů jazyka UML. Zejména pak diagram tříd, diagramu případů užití a další. Na základě tohoto návrhu bude poté vlastní řešení implementováno a to pomocí příslušného zvoleného programovacího jazyka, Frameworku .NET a dalších pomocných knihoven. Jako vývojové prostředí bude použit profesionální vývojářský program Visual Studio od firmy Microsoft. Pro vývoj aplikace bylo použito Vodopádového přístupu k vývoji aplikace.

2 Inventarizace ve společnosti

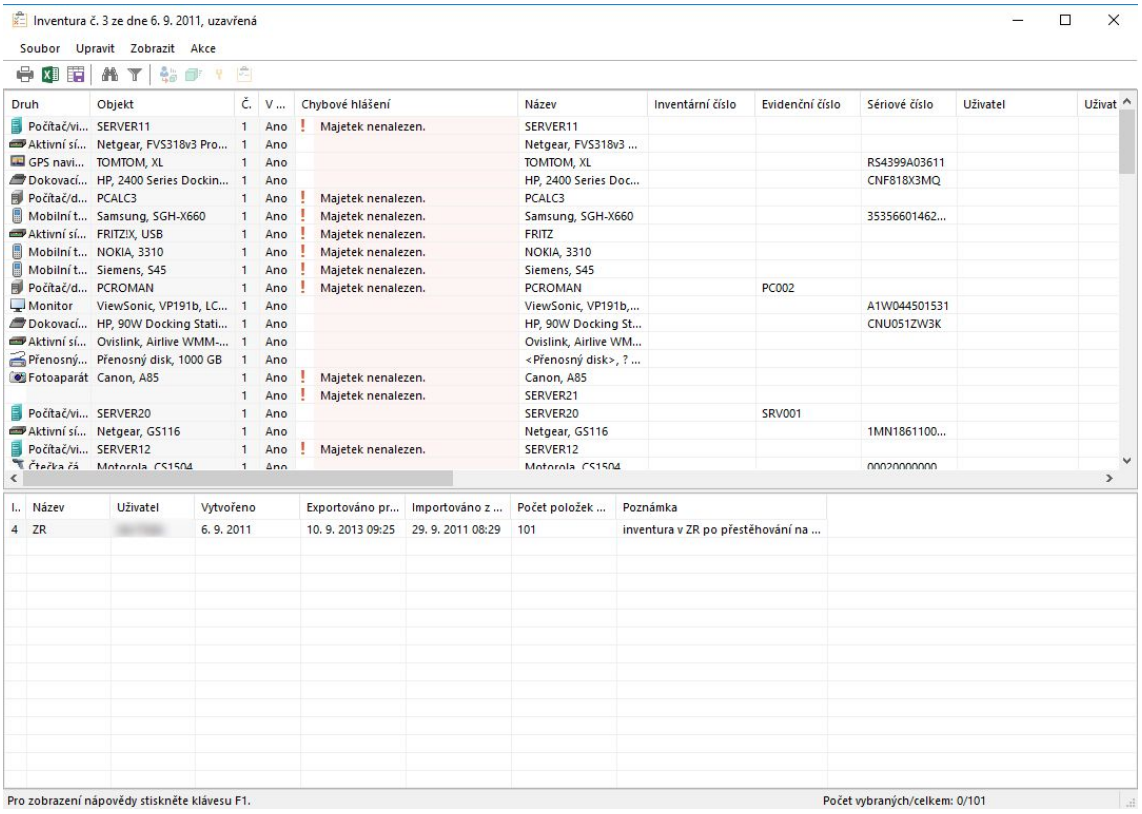
V této kapitole bude nejprve rozebrána aktuální situace inventarizace majetku v dané společnosti. Bude zde objasněno co samotná inventura obnáší a také její požadavky na hardware. Dále také bude definováno kdo inventuru provádí a její perioda opakování. Následovat bude popis samotného průběhu inventury a způsob jejího zpracování.

V druhé části se blíže zaměříme na aplikaci, pomocí které je inventura prováděna. Provedeme její důkladnou analýzu a pokusíme se definovat její klíčové vlastnosti. V neposlední řadě odkryjeme také její omezení a nedostatky.

2.1 Současný stav inventury majetku

Aktuálně provádí inventuru majetku ve společnosti pověřený pracovník, v jehož režii je veškerá správa majetku. Inventura již probíhá elektronicky a to ve třech fázích:

1. Vytvoření nové inventury v konzolové aplikaci pro správu majetku – V této fázi je nejprve potřeba v seznamu inventur přidat novou inventuru, ovšem aby to bylo možné, nesmí existovat jiná otevřená inventura. Je-li tato podmínka splněna lze poté vytvořenou inventuru otevřít a přejít k výběru majetku náležitého k inventuře.
   Ve zmiňované konzolové aplikaci je jednotlivý majetek reprezentován objekty umístěnými hierarchicky do stromu, tedy strom objektů. V tomto stromu vybereme objekt zpravidla nadřazený veškerým objektům majetku, u kterých bude provedena inventura. Poté je již možné vybrat ze záložky Objekty z příslušného gridu majetek a přes příslušnou místní nabídku jej konečně přidat do inventury. Seznam majetku v příslušném okně je zobrazen na obrázku 1 níže.



Obrázek 1: Okno modulu Inventury zobrazující majetek určený k inventuře. Zdroj: ALVAO Documentation Library, 1999

2. Provedení inventury pomocí čtečky čárových kódů – V této fázi je nutné vytvořit již soubor, který bude později vstupem čtečky. V aplikaci pro správu majetku otevřeme okno inventury, v jeho horní části je zobrazen majetek, který byl vybrán k inventuře. Ve spodní části okna příkazem z místní nabídky vytvoříme nový soubor čtečky, do kterého poté přes příkaz Přiřadit majetek do souboru čtečky pod záložkou Akce, přiřadíme majetek do souboru čtečky. Tímto způsobem lze vytvořit i více souborů čtečky, což umožňuje celou inventuru rozdělit na několik částí a provádět ji paralelně více pracovníky.
   Jakmile je soubor pro inventuru připraven jednoduše jej z aplikace vyexportujeme a naimportujeme do čtečky čárových kódů do mobilní aplikace pro inventuru majetku. Poté již nezbývá nic jiného než provést samotnou fyzickou inventuru majetku – tedy načítat čárové kódy jednotlivého majetku, tento proces bude ještě podrobněji zanalyzován v následujícím bodě.

3. Zpracování výsledků – Po dokončení inventury majetku soubor ze čtečky naimportujeme zpět do aplikace pro správu majetku, kde také následně provedeme analýzu výsledků. U inventarizovaných objektů majetku sledujeme hodnoty ve sloupci Chybové hlášení, který informuje o tom, zda byl majetek při inventuře nalezen, zda byla zjištěna změna některých údajů, případně, že se jedná

o nový majetek, který není evidován. V případě zjištění změny údajů je v daném sloupci také uvedeno, jaké údaje byly změněny. Duplicitní údaje jsou zde také brány v úvahu a maji vlastní chybové hlášení ve zmiňovaném sledovaném sloupci. U duplicit je nutné rozhodnout a ručně upravit, který údaj je relevantní a duplicitu odstranit. Informace získané při fyzické mobilní inventuře lze případně v konzolové aplikaci upravit přímo manuálně nebo využít některé z automatizovaných akcí inventury např. při zjištění změny uživatele (tzn. načtený objekt majetku byl nalezen u jiného uživatele), můžeme využít příkazu Převést majetek na uživatele, kterým daný objekt majetku převedeme na aktuálně zjištěného uživatele ve stromu objektů. Po skončení úprav daného objektu inventury, jej označíme jako zpracováno příkazem Zpracovat. Tímto se úpravy uloží nebo v případě nově zjištěných údajů se zapíší do odpovídajících sloupců objektu majetku a nadále se u nichž již nezobrazují chybové hlášení v příslušném sloupci. Takto postupně eliminujeme chybová hlášení, dokud není veškerý inventarizovaný majetek ve stavu Zpracováno.
Posledním krokem je pak samotné uzavření inventury, což provedeme zapsáním data do položky inventury Uzavřeno. V případě úspěšného provedení všech předcházejících kroků inventury je možné zapsat do deníku jednotlivých objektů záznam o jejím úspěšném provedení přes příslušný příkaz z místní nabídky v okně inventury (1).

2.2 Současná aplikace pro inventarizaci majetku

Současná aplikace pro inventuru majetku nese název *ALVAO Mobile Inventory* (2) a byla vyvinuta přímo danou společností. Dále je určena pro zařízení typu Pocket PC/MDA s vestavěnou čtečkou kódů a s operačními systémy Windows Mobile/CE. Nyní již blíže k procesu inventury:

Proces fyzické inventury započíná importem majetku do aplikace, tedy souboru přípravného a exportovaného z konzolové aplikace. Pokračuje nastavením filtru objektů majetku určeného k inventuře a poté se dostává k samotnému snímání čárových kódů na majetku. Následně lze detekované informace upravit a celý proces je ukončen opětovným přenosem souboru zpět do konzolové aplikace, tedy export z mobilní aplikace a import do konzolové aplikace. Jednotlivé fáze budou níže blíže popsány.

1. Import majetku a nastavení filtru – Nutným základem této fáze je import dat (objektů) majetku do mobilní aplikace a to formou soubor ve formátu IXM. Daný soubor je nejprve tedy načten do aplikace a poté je možné zkontrolovat načtený obsah přes příslušný příkazu Menu – Database – Show. Velice užitečným nástrojem aplikace pro efektivní provádění inventury je filtrování objektů majetku. Využití filtru zásadně ovlivňuje orientaci uživatele v inventarizovaných objektech a to především ve větších organizacích s rozsáhlou evidencí majetku.

V současné aplikaci lze filtrovat dle umístění majetku, střediska nebo uživatele, který majetek v rámci organizace vlastní.

2. Snímání kódu – Tato fáze lze považovat za hlavní celého procesu inventury majetku. Jakmile pomocí čtečky sejmeme čárový kód z majetku, je porovnáván se sériovým, inventárním a evidenčním číslem majetku. Na základě toho může následné nastat několik variant výsledku porovnávání:

   a) Majetek byl nalezen v evidenci, mobilní aplikace zobrazí informace o sejmutém majetku a označí jej jako nalezený.

   b) Majetek byl nalezen v evidenci, ale údaje neodpovídají nastavenému filtru. V tomto případě se zobrazí okno s upozorněním, že snímaný majetek neodpovídá zadanému filtru a majetek je možné převést jiného umístění, střediska nebo uživatele.

   c) Majetek nebyl v evidenci nalezen, aplikaci zobrazí zobrazí možnost vytvoření nového majetku a vyplnění jeho údajů.

   d) Snímaný kód je uveden u více objektů majetku, uživatel je vyzván k sejmutí dodatečného unikátního kódu jako např. sériové číslo

   Po sejmutí kódu je nezávisle na výsledku porovnávání také možné údaje o daném majetku upravit přes příslušnou volbu.

3. Zobrazení zbývajícího majetku a Export – Součástí hlavního okna aplikace je mimo jiné informace o počtu objektů majetku, u kterých zbývá načíst čárový kód. Jestliže se uživatel nespokojí pouze s touto informací, může dále využít funkce zobrazení seznamu majetku, kde podle stavu majetku (nalezen/nenalezen) jednoznačně zjistí v jaké fázi snímání kódu majetku se nachází, případně kolik zbývá. V daném seznamu majetku se lze jednoduše orientovat jaký majetek by měl být dále obsažen v místnosti nebo vlastnit uživatel, jak již bylo řešeno výše v souvislosti s filtrováním.
   Zcela posledním krokem procesu inventury je pak export majetku zpět do souboru a následný import do konzolové aplikace spravující majetek, kde lze dále porovnávat skutečný versus evidovaný stav a tedy celkově vyhodnotit výsledky provedené inventury (2).

2.2.1 Klíčové vlastnosti, omezení a nedostatky

- Práce s objekty majetku, využití filtrování – Tuto funkcionalitu lze považovat za jednu z nejdůležitějších pro samotné praktické využití mobilní aplikace. Pomocí filtrů lze jednoduše provádět inventuru i s relativně velkým objemem dat. Při provádění inventury je možné využít filtrů dle umístění, střediska nebo uživatele. Avšak ze strany náročnějších uživatelů může být požadována možnost použít filtr i dle dalších vlastností objektů majetku. Tuto funkcionalitu by tedy bylo možné ještě dále rozvinout a zdokonalit.

- Jednoduché a přehledné UI – Na základně relativně jednoduché funkcionality, kterou současná aplikace pro mobilní inventuru poskytuje je i Uživatelské rozhraní (UI) přehledné a jednoduché na ovládání. Z pohledu uživatele provádějící inventuru to lze považovat za určitý komfort. Zároveň v souvislosti s touto pozitivní vlastností, může být relativně jednoduchá funkcionalita aplikace vnímána jako nedostatek, kdy pokročilý uživatel potřebuje s objekty majetku provádět některé složitější úpravy. Za nedostatek lze také z pohledu UI považovat absenci moderních trendů a prvků v uživatelském rozhraní aplikace. Běžný uživatel bude od aplikace očekávat prvky uživatelského rozhraní, které využívá dnes a denně i v ostatních aplikacích, což souvisí také s použitou technologií.

- Vstup a výstup jako soubor – V době vývoje současné mobilní aplikace se tento typ vstupu a výstupu považoval za standardní a dostačující k danému využití. Ostatně jednalo o nejvhodnější způsob transportu dat mezi konzolovou aplikací pro správu majetku a mobilní aplikaci pro inventuru. Využití souboru je i dnes velice spolehlivou metodou přenosu dat, i když již zároveň částečně na ústupu. Důvodem je rozvoj moderních technologií a s ním přicházející možnost provádět operace online přímo nad cílovou databází.
  Nicméně v tomto konkrétním případě by bylo pro online přenos mezi danými dvěma aplikacemi potřeba využít dalšího softwarového systému, kterým je webová služba. V softwaru, který daná společnost vyvíjí je již webové služby hojně využíváno, jedná se však již o relativně složitý systém, jehož využití v této diplomové práci by přesahovalo její rámec.

- Neuniverzálnost, kompatibilita s pouze jedním typem zařízení – Stávající aplikace pro inventuru je určena pro operační systémy Windows Mobile/CE. Tyto operační systémy jsou aktuálně obsaženy v zařízení typu Pocket PC/MDA, které jsou v dnešní době už spíše na ústupu. Moderním trendem je využití chytrých zařízení – mobilních telefonu nebo tabletu, pro většinu běžných činností, včetně čtení čárových kódů a případné další operace související s jejich zpracováním. Aplikace je tedy pevně fixována pouze na zmiňované typy zařízení. S tímto samozřejmě souvisí i typy využitých technologií při implementaci aplikace. Starší a již méně podporované operační systémy nedisponují širokými možnostmi současných moderních technologií a to jak z pohledu programové funkcionality, tak samotného vzhledu a a designu aplikace. Na současné aplikaci je na první pohled znatelný vývoj s využitím dřívějších technologií. Tuto vlastnost a s ní související považuji jednoznačně za největší nedostatek a tedy tzv. Achilovu patu současné aplikace.
  Při návrhu a následné implementaci mobilní aplikace pro inventuru majetku se pokusíme veškeré nedostatky a omezení současné aplikace eliminovat nebo v případě těch složitějších je alespoň co nejvíce minimalizovat.
  Vzhledem k platformě, na které běží současná aplikace a zachování její určité jednotvárnosti s konzolovou aplikací bude mobilní aplikace navržena a implementována také pro platformu Windows (Univerzální mobilní aplikace). I když

volba jiné mobilní platformy by samozřejmě mohla být také potenciální možností.

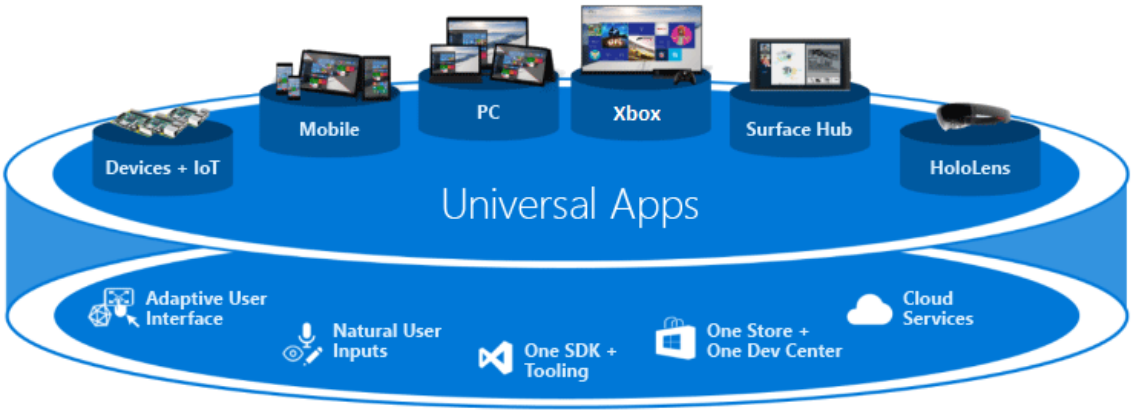
3 Technologie vývoje univerzálních Windows aplikací

V následující kapitole se blíže seznámíme s univerzálními Windows aplikacemi a jejich možnostmi vývoje. Dále zanalyzujeme jednotlivé výhody vývoje pro tuto platformu a také se pokusíme zamyslet na jejími možnými nedostatky, zejména pak při vývoji pro edici Windows 10.

V druhé části pak rozebereme jednotlivé technologie, které je možné při vývoji univerzální Windows aplikací využít.

3.1 Univerzální Windows platforma (UWP)

Aplikace na Univerzální Windows platformě byly poprvé představeny v edici Windows 8 jako Windows Runtime[1] Jak už název *Univerzální* napovídá, jádrem UWP je myšlenka možnosti využívat aplikace této platformy napříč všemi typy zařízení, tedy úplná přenositelnost UWP aplikací. Uživatel si pak může jednoduše zvolit typ zařízení, které je pro daný úkon nejvhodnější. Graficky je univerzálnost UWP znázorněna na obrázku 1.



Obrázek 2: Univerzálnost UWP aplikací. Zdroj: Learn to Develop with Microsoft Developer Network | MSDN, 2016

S příchodem Windows 10 se vývoj Univerzálních Windows aplikací stává snadnějším než dříve. Vývojáři využívají pouze jednu sadu API, jeden aplikační balíček a zároveň mohou svoje aplikace publikovat na jednom místě pro všechny druhy Windows 10 zařízení – od klasického stolního počítače až po Xbox nebo Surface Hub. S čímž také souvisí snazší vývoj a podpora různé velikosti displeje a interakčních – vstupních zařízeních dané aplikace, kdy je její ovládání možné dotykem, myší a klávesnicí, stylusem nebo herním ovladačem.

Níže si uvedeme některé charakteristické vlastnosti, na jejichž základě jsou právě UWP aplikace na Windows 10 svým způsobem speciální:

[1] Jedná se o homogenní architekturu aplikací podporující vývoj v jazyce C++/X, C#, VB.NET a JavaScript. Zároveň nativně podporuje jak x86 tak ARM architekturu.

- Cílové zařízení jsou tzv. device families neboli rodiny zařízení ne konkrétní operační systém – Tedy rodina zařízení identifikuje API knihovny, charakteristiku systému a také jeho chování. V neposlední řadě určuje také sadu zařízení, ve kterých je možné danou aplikaci případně instalovat z obchodu aplikací (Windows store).
- Aplikace jsou zabaleny a distribuovány ve formátu .AppX – To zaručuje snadné nasazení a případné aktualizace.
- API rozhraní jsou k dispozici společné pro celé rodiny zařízení – Základní API rozhraní UWP je společné pro všechny rodiny zařízení Windows. Tedy pokud daná aplikace používá pouze základní API tzv. core APIS, bude bez problému běžet na jakémkoliv Windows 10 zařízením.
- Při specializaci aplikace na určitou rodinu zařízení je možné použít rozšifrující SDK, které obsahují specializované API, díky kterým lze docílit určitého odlehčení aplikace na zařízeních dané rodiny. Zaručuje tedy optimální běh aplikace na daných zařízeních a s tím související i optimální využití zdrojů.
- Adaptivní ovládání a vstup alikace – Jednotlivé prvky uživatelského rozhraní využívají každý pixel displeje efektivně, tzn. je zde zaručena jejich automatická adaptace dle počtů pixelu – velikosti displeje daného zařízení. A jak už bylo řešeno výše, ovládání aplikací lze provést různými typy vstupů. V zásadě existují dvě možností výběru (3; 4).

3.2 Dostupné technologie vývoje

Vývojář Windows 10 Univerzálních aplikací má k dispozici v zásadě dva typy technologií vývoje. Může zvolit mezi API rozhraními Win32 a COM nebo .NET. V obou případech se mu poté dostane možnosti vývoje desktopových aplikací jako je Word, Excel nebo Photoshop, které běží na klasických desktopových Windows a bude schopný plného využití všech výhod daného vývojového prostředí. Nyní blíže zanalyzujeme dané dva typy technologií – popíšeme jejich optimální případy užití a vyčteme jejich klíčové vlastnosti (5).

3.2.1 C++ a Win32

Technologie příhodná pro dosažení nejvyššího výkonu nebo efektivity, zároveň poskytuje přístup k nativním funkcím operačního systému a také se využívá pro DirectX technologie. Jejíž pomocí lze přistupovat tzv. *přímo ke kovu*, tedy vývojář je schopný přímo kontrolovat přidělování paměti výkonové části procesoru (SSE nebo AVX instrukce), prostřednictvím čehož je pak možné dosáhnout nejlepšího výkonu aplikace. C++ kód je možné sdílet mezi Windows, Windows Phone a Windows Store a navíc také mezi jinými platformami než Windows.

Využívá tedy programovací jazyk C++ s API rozhraními COM a není závislý na vysoké úrovni prostředí pro běh, tak jako .NET. Díky tomu se z něj stává správná volba pro přenosné aplikace, kde takové běhové prostředí není k dispozici nebo je příliš náročné na zdroje cílové platformy. Nicméně pozitiva této technologie, přináší na bedra vývojáře vyšší požadavky na znalostí, zkušeností a také větší péči a pozornost nad přístupem do paměti apod.

Níže si pro shrnutí uvedeme výčet některých klíčových vlastností jazyka C++, které umožňují vyvíjet vysoce výkonné aplikace:

- Optimalizace hardwaru včetně striktního kontroly přidělování zdrojů, životnosti objektů, datových typů a dalších.
- Přístup k výkonově orientovaným sadám instrukcí jako SSE a AVX pomocí vlastních funkcí.
- Efektivní a bezpečné generické programování pomocí šablon.
- Velice efektivní a bezpečné využití kontejnerů a algoritmů.
- Využití DirectX zejména pak Direct3D a Direct Compute.
- C++ AMP[2]

Další se tento programovací jazyk využívá při vývoji her a aplikací, které vyžadují podobný grafický a procesní výkon. Umožňuje vývojáři: explicitní správu zdrojů a životnost objektů, optimalizaci na hardwarové úrovni a přímý přístup k DirectX a dalším výkonově orientovaným knihovnám Win32, které podporují vývoj výkonově kvalitních her.

V případě vývoje klasických desktop aplikací jako je tabulkový procesor, textový editor a další, je tento programovací jazyk jasnou volnou a to hlavně pro aplikace silně náročné na výpočty, kdy lze využít velké množství již existujícího zdrojového kódu. V opačném případě se jeví jako lepší volba .NET a příslušné programovací jazyky, které mají dostatečný výkon, poskytují možnost vývoje moderních aplikací s využitím kvalitních UI frameworků a standardních knihoven.

Pro vývoj destopových aplikací v C++ je možné zvolit MFC, Win32 pro UI nebo aplikační framework třetí strany.

- MFC (Microsoft Foundation Class Library) – Solidní kostra aplikace a UI knihovna, která je vývojářům Windows k dispozici od roku 1992. Poskytuje objektově orientované rozhraní pro mnohá předdefinovaná okna, běžné kontroly a další užitečné Windows objekty. I když .NET svým určitým komfortem při vývoji předčí MFC bez větších problémů, MFC je stále nativní UI framework, čímž zůstává jasnou volbou pro mnoho vývojářů klasických Windows desktop aplikací v C++.

[2]Knihovna určena pro paralelní zpracování kódu pomocí grafického procesoru (GPU).

- Win32 – Tzv. handle-based API rozhraní pro Windows platformy v jazyce C. Velice zřídka je volbou pro vývoj moderní aplikace s náročným UI. Obsahuje nespočetné množství funkcí umožnující provádět různé akce s použitím datových typů a struktur operačního systému Windows. Definuje funkce např. pro grafiku, zvuk, ovládací zařízení, sítě a další. Tím se stává zřejmě optimální volbou pro aplikace s nízkými požadavky na UI a nebo pro aplikace, které si vystačí s Windows UI jako např. hry.
- Aplikační frameworky třetích stran – Jelikož programovací jazyk C++ může běžet na širokém spektru platforem a není pevně fixován pouze na Windows nebo .NET runtime, vznikly nové frameworky uživatelského rozhraní, které přináší možnost vývoje multi-platformních aplikací s bohatým uživatelským rozhraním. Některé z nich působí vlastním dojmem, protože obsahují vlastní vzhled. Zatímco jiné používají nativní sadu ovládacích prvků dané platformy. Pomocí těchto frameworků lze sdílet v podstatě kompletní zdrojový kód dané aplikace mezi jednotlivými verzemi aplikace, které běží na Windows nebo jiných platformách, jako je např. OS X nebo Linux (5).

3.2.2 .NET

Tato technologie je vhodná pro programování na vyšší úrovni nebo pro využití méně složitého programovacího jazyka, který může být pro vývojáře snazší. Zároveň je možné výslednou aplikaci poté sdílet mezi Windows Store, Windows Phone a Windows desktop aplikacemi použitím přenosných knihoven tříd. Využívá programovací jazyky C#, C++, F# a Visual Basic.

Při programování v .NETu lze základní desktop aplikace dále rozvíjet a obohatit je o nové vzrušující funkce a to pouze s využitím stávajících schopností a opakovaného použití kódu mezi jednotlivými zařízeními. Níže uvádíme výčet některých features, které .NET nabízí:

- Zvýšená produktivita a bezpečnost běhového prostředí jako je automatická správa paměti kontrola datových typů, zachytávání výjimek a také management vláken (vícevláknové programování).
- Široké spektrum datových typů.
- Datové modelování (ADO, LINQ, datové služby WCF)
- LINQ (Language Integrated Query) – Integrovaný dotazovací jazyk.
- Knihovna poskytující správu data a času
- Serializace
- Webové služby
- Bezpečnost a kryptografie

- Knihovna pro paralelní programování

.NET Framework poskytuje dále několik inovací, které vývojáři usnadní proces rozšíření aplikace na nové platformy bez větších změn v její architektuře. Často je tedy možné použít znovu již stávající kód, což velice zjednodušují následující funkce daného framworku:

- Návrhový vzor Model-View-ViewModel (MVVM) – Tento návrhový vzor usnadňuje vývoj na klientské platformy Microsoft a to jednoznačným oddělením dat, stavu aplikace a uživatelského rozhraní. pomocí čehož lze vytvořit čistý a udržovatelný kód, který lze snadno sdílet mezi více zařízeními.
- Přenosné knihovny tříd – .NET přenosné knihovny jsou velice důležité pro podporu sdílení binárních souborů mezi jednotlivými platformami – Windows Store, Windows Phone a dalších aplikací. Implementace klientské logiky s těmito knihovnami výrazně zjednodušuje tvorbu moderních aplikací s komfortním UI napříč různými platformami.
- Moderní uživatelský prožitek – K uspokojení konceptů, které jsou v současné době vyžadovány uživatelem, využívá nejnovější inovace platformy .NET. Např. konstrukční principy jako *fast and fluid*, *do more with less* a další, které mohou být aplikovány na existující aplikace použitím moderního rozhraní pro XAML design nebo asynchronního programování (5).
- Extensible Application Markup Language (XAML) – Deklarativní jazyk primárně určený k návrhu uživatelského prostředí. Má však i alternativní způsob využití, jako je definice konfigurace aplikace nebo popisu procesů (Workflow Foundation).
  Vše co lze udělat pomocí XAML, je možné zapsat i v jazycích C# nebo Visual Basic .NET. Ze zjednodušeného pohledu pomocí XAML definujeme jaké objekty se mají vytvořit a jaké vlastnosti se jim mají nastavit. Pracuje s objektovým modelem .NET – buď s dodávanými komponenty s .NET Frameworkem nebo s jakoukoliv jinou podstrčenou knohovno, na niž se ze XAML souboru odkážeme pomocí XML namespace (6).

Zaměříme-li se nyní přímo na vývoj UWP aplikací pomocí .NET Frameworku, dostaneme se k jeho speciální verzi s názvem .NET Core Framework. Jedná se o verzi .NETu – modulární implementaci uzpůsobenou pro moderní zařízení a také cloud řešení. Může být jednoduše přenesena a použita v mnoha různých prostředích. Jádro .NET Frameworku je open source a je plně podporován společností Microsoft na systémech WIndows, Linux a Mac OS X (7).

4 Inventura a inventarizace majetku

V této kapitole si definujeme pojem inventura a inventarizace majetku. Přiblížíme si proces inventarizace obecně a také rozebereme strukturu majetku pro inventarizaci, tedy jednotlivé druhy majetku, pro které ji lze provádět.

4.1 Definice pojmů inventura a inventarizace majetku

Inventarizace majetku je nedílnou součástí účetnictví. Principem této administrativní činnosti je srovnání skutečného stavu zásob, dlouhodobého hmotného majetku, pohledávek a závazků se stavem vedeným v účetních knihách. Inventarizace i inventura je nejčastěji prováděna periodicky, např. ke konci každého účetního období. Lze však provést i inventuru mimořádnou. Obecně jejich četnost provádění závisí na činností a oboru podnikání dané společnosti. Existují obory, ve kterých je nutné, kontrolovat stavy častěji než jednou ročně.

Inventarizace a inventura jsou často používány ve stejném významu, nicméně jejich význam není zcela stejný. V případě inventarizace se jedná o širší pojem, jehož samotnou části inventura je, avšak zahrnuje navíc celý soubor dalších činností, jako je např. zjištění inventarizačních rozdílů, vyšetření příčin vzniku inventarizačních rozdílů, sjednání nápravy těchto rozdílů apod. Zatímco inventura se zabývá pouze zjištěním skutečného stavu majetku a závazků k určitému dni u jednotlivých složek. Tyto stavy je možné zjistit fyzickou inventurou u majetku, u kterého lze vizuálně zjistit jeho existenci nebo dokladovou inventurou u závazků a majetku, u kterého nelze vizuálně zjistit jeho existenci a to včetně jiných aktiv, jiných pasiv a skutečností účtovaných v knize podrozvahových účtů (8; 9).

4.2 Proces inventarizace

Povinnost provádět inventarizaci přímo vyplývá ze zákona o účetnictví (10, s. 52) a i když pojem inventarizace není v zákoně přímo vymezen, lze jej na základě jeho funkce, kterou v rámci systému účetnictví plní, vymezit jako proces. Výsledkem tohoto procesu je pak stav majetku a závazků, které jsou zachyceny v účetnictví v souladu se skutečností. Rozdělení daného procesu na několik etap:

1. Zjištění skutečného stavu majetku a závazků – Probíhá v případě hmotného majetku pomocí fyzické inventury (počítání, vážení, měření), v případě nehmotného majetku a závazků prostřednictvím dokladové inventury.
2. Tvorba seznamů skutečně zjištěného majetku a závazků (inventurní soupisy).
3. Srovnání skutečných stavů majetku a závazků se stavy účetními a vyčíslení inventarizačních rozdílů.
4. Vypořádání inventarizačních rozdílů včetně účetních zápisů, které uvádějí do souladu účetní stav se skutečným.

5. Porovnání ocenění majetku a závazků v účetnictví s aktuální tržní cenou – V případě, že je tržní ocenění nižší, je třeba použít zásadu opatrnosti[3].

V zákoně o účetnictví rovněž nejsou definovány technické a organizační otázky průběhu procesu inventarizace. Je tedy výlučně v pravomoci účetní jednotky, jak jednotlivé etapy průběhu inventarizace konkretizuje ve svém vnitřním předpisu týkající se inventarizace. Daný předpis musí respektovat termíny stanovené v zákoně o účetnictví, nicméně může upravovat i další termíny nad rámec zákona.

4.2.1 Druhy inventarizací

1. Z časového hlediska

   a) Řádná

   - Periodická – Účetní jednotky ji provádějí k okamžiku sestavení účetní závěrky.
   - Průběžná – Provádí se postupně v průběhu celého účetního období.

   b) Mimořádná – Provádí se se v mimořádných případech, jako je vznik podniku, jeho sloučení, rozdělení, uzavření dohody o hmotné odpovědnosti zaměstnance, po živelné pohromě, vloupání apod.

2. Dle rozsahu

   a) Úplná – Týká se veškerých složek majetku a závazků.

   b) Dílčí – Týká se jen některých složek majetku a závazků (11).

4.3 Způsoby provedení inventury a struktura majetku pro inventarizaci

Jak už bylo řečeno výše, inventura je součástí procesu inventarizace a jejím účelem je zjišťování skutečného stavu, dá se tedy považovat za nejdůležitější etapu procesu inventarizace. Inventuru dle charakteru kontrolovaného majetku lze provádět dvěma způsoby:

- Fyzická inventura – Jejím účelem je zjištění skutečného stavu u hmotného majetku (dlouhodobý majetek, zásoby, pokladní hotovost apod.). Metodou pro zjištění skutečných stavů je v tomto případě počítání, měření, vážení a ohledávání. Jsou ovšem i případy, u kterých nelze použít jiných přesnějších metod pro zjištění skutečného stavu než je technický propočet – uhlí, písek, tekuté látky, žížaly a další.

[3] v účetnictví se vykazují a do výše zisku promítají všechny předpokládané a očekávané ztráty, rizika a znehodnocení majetku, i když ještě nenastaly a jejich výše není spolehlivě zjistitelná. Tato zásada se realizuje tvorbou rezerv a opravných položek.

Jestliže provádíme fyzickou inventuru u zásob, které jsou skladovány v původních neporušených obalech, lze tyto obaly pouze přepočítat s jejich namátkovým otevřením a ověřením množství, případně kvality těchto zásob. Dále během provádění inventury využít také evidenční podklady (sestavy, karty majetku), samozřejmě s tímto roste náročnost inventury, na druhou stranu stejně tak se zvyšuje její věrohodnost. Případy, kdy je možné se opírat o evidenční podklady a kdy ne, definuje samotná účetní jednotka ve svých vnitropodnikových předpisech.

- Dokladová inventura – Tento způsob provedení inventury slouží k jejímu provedení u majetku, u kterého nelze provést fyzická inventura (pohledávky, položky nehmotné povahy), u rezerv, závazků, časového rozlišení a u některých podrozvahových položek. Při dokladové inventuře je skutečný stav majetku a závazků zjišťován na základě různých písemných dokumentů. Za typický příklad lze považovat materiál nebo zboží na cestě, v tomto případě by měla proběhnout jednoznačně dokladová inventura. Zároveň se při ní ověřují faktury dokládající nákup, zjišťují důvody zpoždění dodávky – ve většině případů je požadováno potvrzení ze skladu, že dodávka nebyla přijata na sklad. Dokladová inventura je dále nutná také u zásob volně ložených, kde např. kvůli nepravidelnosti útvaru, ve kterém jsou skladovány, nelze použít technický výpočet. Rovněž je dokladová inventura použita v případech, kdy v době provedení inventury jsou zásoby např. u dodavatele na zpracování či opravě, zapůjčeny zaměstnanci či společníkovi nebo je dlouhodobý majetek dočasně pronajat apod.

V tabulce 1 lze vidět přehled způsobů inventarizování jednotlivých skupin majetku a závazků. Jak je z uvedené tabulky jasné, inventarizaci podléhají veškeré složky aktiv a pasiv, jediná výjimka je v případě vlastního kapitálu, který je dopočtenou položkou. To stejné platí i pro položky sledované v podrozvahové evidenci[4], jako např. najatý majetek (provádí se fyzickou inventurou), odepsané pohledávky, pohledávky ze smluvních pokut, poskytnuté záruky, ručení ze směnky (provádí se dokladovou inventurou). Co je potřeba odlišit, je tzv. operativní evidence, jež nepředstavuje součást účetního systému společnosti, její obsah, forma vedení a také povinnost inventarizace není nijak právně definována.

Jako specifický případ vzhledem k inventarizaci majetku se jeví inventarizace pozemků, kde by měla být provedena zároveň inventura jak fyzická, tak i dokladová. Dokladovou inventurou je zaručena kontrola stavu v účetní evidenci oproti evidenci v katastru nemovitostí. Jde zde zejména o údaje o vlastníkovi pozemku, výměře pozemku, čísla parcely, druh pozemku a také čísel evidenčního a vlastnického listu. Při dokladové inventuře však mohou vznikat další náklady při získávání výpisů z katastru nemovitostí v případě společnosti s větším počtem pozemků v majetku podniku. Oproti tomu fyzická inventura se přímo v terénu zjišťuje a srovnává

[4]Evidence nejistých nebo podmíněných majetků a závazků, které se nezobrazují v rozvaze.

s evidencí skutečný stav a do inventurních soupisů[5] se zapisují informace jako jsou vyznačení pozemků, jejich zajištění před vniknutím, škodou, povodněmi či dešti, dále také stav pozemku (zamokření či devastace), stav zastavění pozemku a další(11; 12).

4.4 Výsledek inventarizace – inventarizační rozdíly

Pomocí porovnání skutečného stavu majetku a závazků na základě inventarizace a aktuálních zůstatků jednotlivých účtu majetku a závazků, mohou být výstupem tyto situace:

- Rovnovážný stav (ideální situace) – Skutečný a účetní stav se shodují.
- Manko (schodek, přirozené technologický úbytek) – Skutečný stav je nižší než účetní.
- Přebytek – Skutečný stav je vyšší než účetní.

Přirozený technologický úbytek

Tento úbytek zejména zásob může vznikat jako důsledek vyplývající z jejich přirozených vlastností, nelze tomu při manipulaci nebo skladování zabránit, např. seschnutí, vypaření nebo rozprášení. Ustanovení normy těchto přirozených úbytků v zákoně neexistuje, tudíž je opět v pravomoci účetní jednotky, aby ve své vnitřní směrnici stanovila racionální a zdůvodněný metodický postup pro vyčíslení těchto úbytků (statické zjišťování v časové řadě, výpočet na základě fyzikálních vlastností zásob apod.) Stejný postup platí i pro stanovení výši ztratného např. v % z tržeb, avšak tento způsob musí účetní jednotka zároveň obhájit u správce daně, jedná se totiž také o daňově uznatelný náklad.

Manko

Mankem rozumíme rozdíl, který převyšuje normu přirozených úbytků. Co se týče hmotné odpovědnosti lze jej v souladu se zákoníkem práce a zákonem č.40/1964 Sb., občanský zákoník, ve znění pozdějších předpisů, řešit v určitém rozsahu formou pohledávky za těmito odpovědnými osobami. V případě, že se jedná o tzv. zaměnitelné zásoby (mouka hrubá a polohrubá, spojovací materiál, výrobky lišící se pouze detaily atd.), může vzniknout na jednou druhu zásob manko a na druhém přebytek, avšak nejedná se o manko jako takové a nelze jej řešit hmotnou odpovědností, řešením jsou pouze účetní zápisy.

[5] Cílem těchto soupisů je písemnou formou zachytit skutečné stavy majetku a závazků, které byly zjištěny inventurou. Zákon o účetnictví nikterak konkrétně neupravuje jejich obsah, ale požaduje jejich vyhotovení

Schodek na pokladně

Může nastat při inventarizaci peněžních prostředků v hotovosti a cenin, kdy skutečný stav peněžních prostředků dosahuje nižších hodnot, než stav účetní. Rozdíl je v případě podepsané hmotné odpovědnosti předepsán k náhradě odpovědnému pracovníkovi.

Přebytek

V zásadě mohou v účetní jednotce vzniknout dvěma způsoby:

- Přebytky vzniklé chybným zaúčtováním vyskladnění.
- Opakovaným vyskladněním menšího množství než bylo deklarováno nebo naopak přijetím většího množství na sklad (12; 13).

5 Scénáře použití mobilní aplikace pro inventuru

V následující kapitole zanalyzujeme jednotlivé scénáře a příběhy použití univerzální mobilní aplikace pro inventuru majetku. Každý možný scénář nejprve popíšeme a poté definujeme způsob jeho řešení v navrhované aplikaci.

5.1 Načtení/import objektů majetku určených k inventuře

Uživatel, který provádí fyzickou inventuru, má již k dispozici připravený soubor obsahující objekty majetku určené k inventuře. Zvolený soubor chce načíst do aplikace a začít provádět inventuru.

Způsob řešení

Uživatel v mobilní aplikaci zvolí v hlavním menu (postranní lišta) tlačítko sloužící k importu souboru s objekty majetku. Následně se mu zobrazí okno aplikace, ve kterém může procházet lokální úložiště zařízení a vybrat určený soubor k importu. Poté vybere vhodný soubor a výběr potvrdí. V případě úspěšného načtení souboru aplikace vrátí odpovídající informační hlášku a po jejím potvrzení, uživatele přesměruje přímo na nově načtený seznam objektů. Jestliže se soubor nepodaří naimportovat do aplikace je uživatel vyzván k opětovnému výběru souboru s objekty majetku k inventuře. Dané okno lze případně samozřejmě zrušit a ručně zkontrolovat správnost souboru.

Způsob řešení po změně hlavního menu aplikace

Změna pouze v umístění příslušné volby, uživatel ji vybere z výchozí obrazovky aplikace – hlavního dlaždicového menu a scénář dále pokračuje stejně jako v původním způsobu řešení.

5.2 Zobrazení objektů majetku určených k inventuře

Uživatel si chce zobrazit kompletní seznam majetku, u kterého je potřeba provést inventuru a prozkoumat informace o něm.

Způsob řešení

Uživatel z hlavního bočního menu na levé straně aplikace přejde přes příslušné tlačítko na Seznam majetku, kde může nad jednotlivými vlastnostmi majetku využívat filtry nebo upravit vlastnosti některého objektu majetku.

Způsob řešení po změně hlavního menu aplikace

Změna pouze v umístění příslušné volby, uživatel ji vybere z výchozí obrazovky aplikace – hlavního dlaždicového menu a scénář dále pokračuje stejně jako v původním způsobu řešení.

5.3 Způsob řazení objektů majetku v seznamu majetku

Uživatel potřebuje majetek v seznamu majetku seřadit podle jedné určité z jeho vlastností (jeden sloupec) a to vzestupně nebo sestupně. Ve výchozím řazení seznamu se nedokáže zorientovat a lépe se mu pracuje při jiné volbě seřazení, jelikož např. i reálné rozmístění majetku odpovídá seřazení dle jeho jedné určité vlastnosti.

Způsob řešení

Uživatel v seznamu objektů majetku gestem dotyku na název daného sloupce v jeho záhlaví seřadí celý seznam podle daného sloupce vzestupně. Při zopakování stejného gesta dotyku seznam seřadí naopak sestupně.

5.4 Filtrování nad Seznamem majetku

Uživatel provádějící inventuru potřebuje, např. z důvodu velkého množství načtených objektů majetku v aplikaci, využít filtr nad určitou vlastností, aby mohl dále pracovat pouze s majetkem splňující daný filtr (např. zobrazení majetku pouze určitého druhu)

Způsob řešení

V seznamu majetku má uživatel k dispozici filtry nad všemi sloupci tabulky odpovídající vlastnostem objetku majetku.Poté do zvolených filtrů na všemi sloupci tabulky vyplní své požadavky na filtrování. Seznam majetku se automaticky již průběžně obnovuje a zobrazuje pouze objekty majetku odpovídající daným filtrům. Filtr lze samozřejmě využít nad jedním nebo více sloupci zároveň.

5.5 Naskenování čárového kódu určitého objektu majetku

Uživatel má zobrazen Seznam majetku případně s aplikovanými filtry. V seznamu nalezne majetek, u kterého chce sejmout čárový kód a zjistit zda jeho uvedené informace odpovídají realitě.

Způsob řešení

Uživatel na předem vybraném objektu spustí snímání čárového kódu majetku a to tak, že gestem dotyku do prostoru řádku si nejprve zobrazí detail konkrétního majetku a ze zobrazeného detailu vybere odpovídající volbu. Čárový kód sejme

a zobrazí se mu na základě výsledku porovnání sejmutého sériového, inventárního nebo evidenčního čísla s evidencí, určité okno aplikace. Případně aplikace uživateli umožní označit majetek jako nalezený.

5.6 Vyhledání majetku v evidenci sejmutím čárového kódu

Uživatel má k dispozici majetek s čárovým kódem, ale neví o jaký majetek se přesně jedná. Chce tedy nejprve zkusit podle čárového kódu majetek v evidenci vyhledat a poté teprve dále zkoumat, zda evidované a skutečné informace o majetku souhlasí.

Způsob řešení

Uživatel provádějící fyzickou inventuru v hlavním postranním menu na levé straně aplikace vybere volbu *Sejmout a vyhledat* a přímo začne snímání čárového kódu objektivem fotoaparátu daného chytrého zařízení. Potom co kód sejme, zobrazí se mu nalezený objekt majetku s odpovídajícím sériovým, inventárním nebo evidenčním číslem. V případě neúspěšného vyhledávání, může proces opakovat nebo zvolit jiný postup.

Způsob řešení po změně hlavního menu aplikace

Změna pouze v umístění příslušné volby, uživatel ji vybere z výchozí obrazovky aplikace – hlavního dlaždicového menu a scénář dále pokračuje stejně jako v původním způsobu řešení.

5.7 Zobrazení objektů majetku dle uživatele, střediska nebo umístění

Uživatel potřebuje na základě vlastností jednoho vybraného objektu majetku zobrazit všechen majetek daného uživatele, střediska nebo v určitém umístění.

Způsob řešení

Uživatel v seznamu majetku nejprve zobrazí detail majetku, který obsahuje příslušné volby, vybere pak např. volbu *Majetek uživatele* a aplikace do filtru sloupce *Uživatel* automaticky doplní hodnotu stejnou, jakou má momentálně vybraný majetek a zobrazí tedy odpovídající majetek. Obdobným způsobem uživatel zobrazí majetek v daném umístění nebo středisku.

Pozn. Samozřejmě uživatel může celý proces provést i ručně – tzn. vyplnit filtr na určitým sloupcem napsáním hodnoty a zobrazit opět odpovídající majetek.

5.8 Editace vlastností objektů majetku

Uživatel během inventury zjistil nesrovnalost v jedné nebo více vlastností určitého objektu majetku. Chce tedy vlastnost/vlastnosti upravit.

Způsob řešení

Uživatel v seznamu majetku nejprve zobrazí detail majetku, kde vybere volbu *Upravit* a aplikací je mu zobrazen formulář s jednotlivými vlastnostmi daného majetku, kde každou z nich může editovat, jak je potřeba. Poté buď příslušným tlačítkem změny uloží a je přesměrován zpět na Seznam majetku nebo naopak tlačítkem *Zrušit* se vrátí na Seznam majetku bez uložení jakýchkoliv změn.

5.9 Export majetku po provedené inventuře

Uživatel již dokončil fyzickou inventuru u veškerého majetku načteného v aplikaci a nyní chce výsledky inventury opět nahrát do konzolové aplikace k dalšímu zpracování. K tomu potřebuje majetek z mobilní aplikace zpět vyexportovat.

Způsob řešení

Uživatel v hlavním postranním menu mobilní aplikace zvolí příslušnou volbu pro export. Aplikace zobrazí okno pro uložení souboru s vyexportovaným majetkem, kde uživatel definuje jméno daného souboru a jeho umístění v lokálním úložišti. Poté již pouze export potvrdí a aplikace provede požadovaný export do specifikovaného souboru.

Způsob řešení po změně hlavního menu aplikace

Změna pouze v umístění příslušné volby, uživatel ji vybere z výchozí obrazovky aplikace – hlavního dlaždicového menu a scénář dále pokračuje stejně jako v původním způsobu řešení.

6 Návrh vlastního řešení

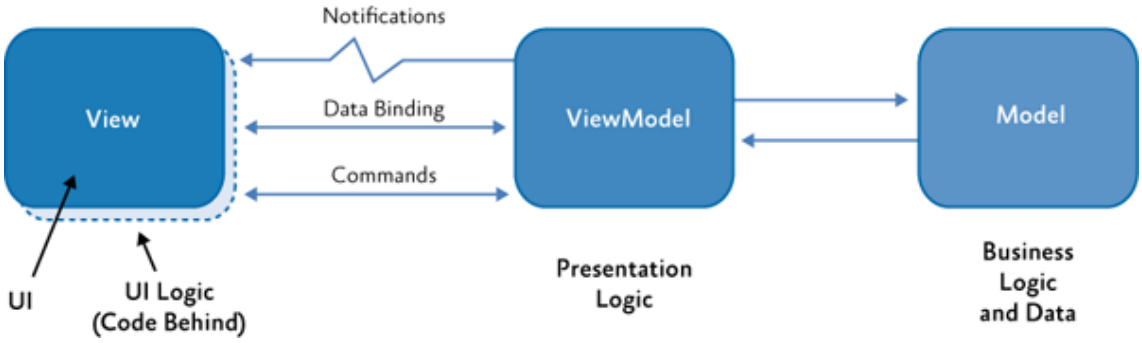
V následující kapitole bude postupně popsáno navržené řešení univerzální mobilní aplikace pro inventuru majetku. Popíšeme její jednotlivé části v rámci použití návrhového vzoru Model-View-ViewModel (MVVM) a rozebereme navržený diagram tříd a případů užití aplikace, k jejich vytvoření využijeme UML notace.

6.1 Architektura aplikace - Návrhový vzor MVVM

Návrhový vzor Model-View-ViewModel je primárně určen pro WPF[6] aplikace, nicméně lze bez problému využít i při vývoji UWP aplikací.

Tento návrhový vzor nabízí komfortní možnost, jak oddělit logiku aplikace a uživatelské rozhraní. Díky mohu bývá zpravidla méně kódu, je přehlednější a případné změny v kódu nejsou extrémně složité na implementaci. Principem MVVM je tedy tvorba třídy držící stav aplikace, tzv. ViewModel. Této třídy se pak uživatelské rozhraní dotazuje a podle výsledků vykresluje jednotlivé ovládací prvky. Na stejném principu probíhá i opačný proces, kdy uživatel zadá nebo požaduje informace prostřednicím UI, tyto požadavky jsou propagovány do ViewModelu (14).

Graficky znázorněn princip návrhové vzoru MVVM lze vidět na obrázku 3 níže.



Obrázek 3: Princip návrhového vzoru MVVM. Zdroj: Learn to Develop with Microsoft Developer Network | MSDN, 2016

V případě navrhované aplikace je Model reprezentován přímo samostatnou datovou třídou, jelikož jsou data ukládána pouze v paměti aplikace a také z důvodu její co nejnižší technické náročnosti. Zároveň se nejedná o složité datové struktury, tudíž není potřeba využití databázového úložiště.

Jednotlivé Views navrhované aplikace zastupují jednotlivé stránky aplikace, tzn. co stránka aplikace to jeden View. Zároveň je v nich s výhodou využito jazyka XAML (Extensible Markup Language), který byl již krátce zmiňován. XAML

[6]Windows Presentation Foundation je framework pro komplexní tvorbu bohatých formulářových aplikací, který je součástí .NET frameworku. Disponuje širokou paletou formulářových prvků a také umožňuje bohaté stylování vzhledu aplikace.

výchází z XML, tedy značkovacího jazyka navrženého pro použití vlastních značek (tagů) a s možností velice širokého využití. Co se výhod jazyka XAML týče, za jednu z nejdůležitějších lze považovat zmiňované oddělení designu a aplikační logiky, což odpovídá také principu návrhovému vzoru MVVM. Další výhoda přichází v podobě kratšího a srozumitelnějšího zápisu kódu a možnosti použití vizuálního návrhu GUI (Grafického uživatelského rozhraní) aplikace.

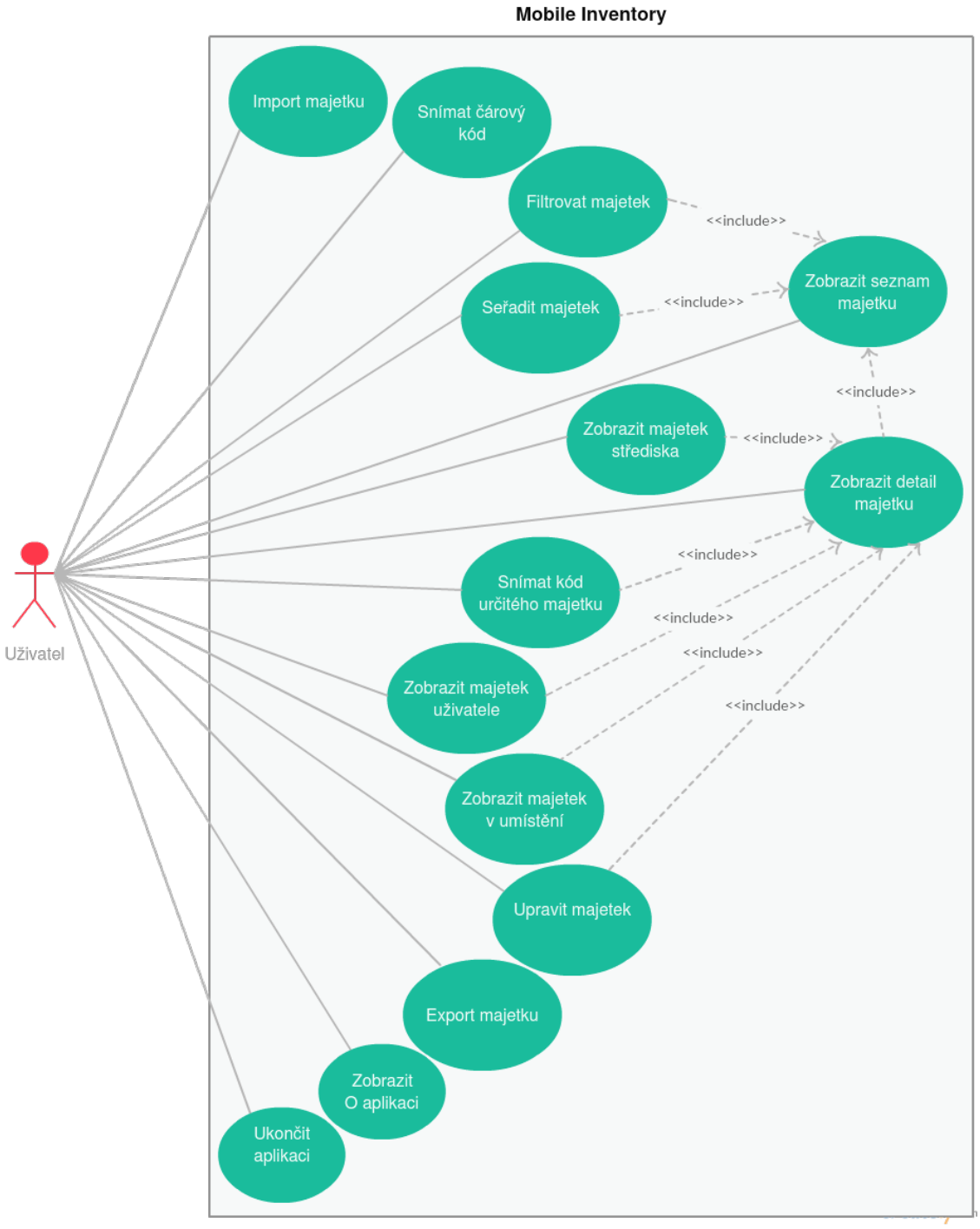
Funkce navrženého řešení se jednak týkají různých operací přímo nad View a jelich programovou logiku zajišťuje code-behind jednotlivých Views. Další funkce související s úpravou přímo nad daty jsou implementovány v již zmiňovaném ViewModelu aplikace.

6.2 Diagram případů užití

Nyní si přiblížíme diagram případů užití navrhované aplikace a provedeme specifikaci hlavních případů užití aplikace.

Jak můžeme vidět na obrázku 4, aktérem aplikace je běžný uživatel, který provádí inventuru majetku. Může se jednak o administrativního pracovníka či přímo správce majetku, to zda v rámci celé společnosti nebo např. určitého oddělení, to již závisí na konkrétní struktuře dané společnosti. V diagramu jsou zobrazeny případy užití se složitější vnitřní logikou, ty se týkají hlavní funkcionality aplikace, jedná se zejména o Import majetku, Snímat majetek, Filtrovat majetek, Zobrazit detail majetku, Upravit majetek a další. Mezi méně složité diagramy užití pak spadá diagram užití Zobrazit O aplikaci a Ukončit aplikaci.

Za zmínku dále stojí i speciální vazba *include* mezi některými případy užití, která vyjadřuje určitou závislost mezi dvěma případy užití. Případ užití napojený pomocí této vazby, se spustí vždy, když je spuštěn případ, na který je napojen. V navrhované aplikaci se tato vazba objevuje např. mezi případem užití Filtrovat majetek a Zobrazit Seznam majetku, znamená to tedy, že v případě spuštění případu užití Filtrovat majetek je vždy spuštěn i případ užití Zobrazit Seznam majetku. Podobně když uživatel chce upravit majetek, musí nejprve zobrazit detail majetku a aby zobrazil detail majetku, musí zobrazit Seznam majetku, tyto všechny tři případy užití mají mezi sebou tedy zmiňovanou vazbu.



Obrázek 4: Diagram případů užití aplikace

6.2.1 Případ užití – Import majetku

Případ užití umožňuje uživateli načíst ze souboru majetek určený k inventuře.

Účastníci

- Uživatel
- Systém

Vstupní podmínky

Uživatel musí mít k dispozici soubor obsahující majetek k inventuře. Soubor se musí vyskytovat v lokální, síťové nebo sdílené složce daného zařízení a musí uživatel musí mít u daného souboru právo na čtení.

Základní tok

1. Systém zobrazí okno pro výběr souboru.
2. Uživatel vybere soubor, který chce načíst a provést na něm inventuru a okno potvrdí.
3. Systém načte majetek ze souboru do aplikace a vrátí uživateli okno potvrzující úspěšné načtení.
4. Uživatel úspěšné načtení bere na vědomí - okno potvrdí.
5. Systém spustí případ užití *Zobrazit Seznam majetku*.

Alternativní tok

3.1 – Pokud uživatel vybere neplatný soubor, systém zobrazí uživateli zprávu o chybě s výzvou na kontrolu struktury souboru a následného opětovného pokusu o import.
3.2 – Uživatel soubor opraví a pokračuje na bodu 2. základního toku.

Podmínky pro ukončení

Nejsou specifikovány – Případ užití končí spuštěním dalšího případu užití.

6.2.2 Případ užití – Snímat čárový kód

Případ užití umožňuje uživateli sejmout čárový kód z objektu majetku, vyhledat daný majetek v seznamu majetku, zobrazit jeho detail a porovnat reálné informace o daném majetku s vedenou evidencí.

Účastníci

- Uživatel
- Systém

Vstupní podmínky

V systému musí být načten soubor s majetkem určeným k inventuře. V chytrém zařízení, na kterém je aplikace spuštěna, musí být k dispozici kamera (objektiv fotoaparátu).

Základní tok

1. Systém uživateli zpřístupní režim kamery/fotoaparátu pro sejmutí čárového kódu.
2. Uživatel sejme čárový kód z majetku pomocí objektivu fotoaparátu daného zařízení.
3. Systém vyhledá majetek s odpovídajícím sériovým, inventárním nebo evidenčním číslem a zobrazí detail daného majetku.
4. Uživatel prověří reálné informace o daném majetku s těmi evidovanými.

Alternativní tok 1

3.1 – Pokud systém nenajde žádný majetek se stejným sériovým, inventárním nebo evidenčním číslem, zobrazí odpovídající zprávu uživateli s doporučením vyhledat majetek v seznamu majetku a zkontrolovat u něj daná identifikační čísla – spustí případ užití Zobrazit Seznam majetku.
3.2 – Uživatel chybné údaje opraví a celý případ užití opakuje nebo pokračuje případem užití *Snímat kód určitého majetku*.

Alternativní tok 2

4.1 – Pokud evidované a reálné informace neodpovídají, uživatel je upraví přes příslušnou volbu v detailu objektu majetku a poté pokračuje bodem 4. základního toku.

Podmínky pro ukončení

Snímaný čárový kód je nalezen u některého objektu majetku.

6.2.3 Případ užití – Filtrovat majetek

Případ užití umožňuje uživateli využívat nad seznamem majetku filtrování dle jedné nebo více vlastností objektu majetku.

Účastníci

- Uživatel
- Systém

Vstupní podmínky

V systému musí být načten soubor s majetkem určeným k inventuře, majetek je pak zobrazen v seznamu majetku.

Základní tok

1. Systém spustí případ užití *Zobrazit Seznam majetku*.
2. Uživatel vyplní hodnotu, dle které chce filtrovat nad jedním nebo více vlastnostmi majetku do příslušných polí pro filtrování.
3. Systém zobrazí majetek odpovídající zadanému filtru.

Alternativní tok 1

1.1 – Uživatel může nejprve majetek seřadit tedy projít případ užití *Seřadit majetek* a poté pokračovat bodem 2. základního toku.

Alternativní tok 2

3.1 – Pokud zadanému filtru nevyhovuje žádný odpovídající objekt majetku, systém nezobrazí v seznamu majetku žádný majetek, filtr opraví a tok pokračuje na 2. bodu základního toku.

Podmínky pro ukončení

Uživateli je zobrazen majetek odpovídající zadaným filtrům.

6.2.4 Případ užití – Seřadit majetek

Případ užití umožňuje uživateli řazení majetku v seznamu majetku a to dle jedné z jeho vlastností sestupně nebo vzestupně.

Účastníci

- Uživatel
- Systém

Vstupní podmínky

V systému musí být načten soubor s majetkem určeným k inventuře, majetek je pak zobrazen v seznamu majetku. Ve výchozím stavu je zobrazený Seznam majetku seřazen dle sloupce *Název* vzestupně.

Základní tok

1. Systém spustí případ užití Zobrazit Seznam majetku.
2. Uživatel klikne (případně využije gesto dotyku) na název sloupce, dle kterého chce zobrazený Seznam majetku seřadit.
3. Systém zobrazí Seznam majetku seřazený dle daného sloupce.

Alternativní tok 1

1.1 – Uživatel může nejprve využít filtrů nad seznamem majetku tedy projít případ užití *Filtrovat majetek* a poté pokračovat bodem 2. základního toku.

Alternativní tok 2

2.1 – Pokud uživatel požaduje seřazení v opačném pořadí (sestupně/vzestupně), klikne znovu na název příslušného sloupce.

Podmínky pro ukončení

Uživateli je zobrazen Seznam majetku seřazený dle požadovaného sloupce.

6.2.5 Případ užití – Zobrazit Seznam majetku

Případ užití umožňuje uživateli zobrazit seznam majetku, který je určen k inventuře a byl v rámci případu užití *Import majetku* načten do aplikace.

Účastníci

- Uživatel
- Systém

Vstupní podmínky

V systému musí být bez chyb načten soubor s majetkem určeným k inventuře.

Základní tok

1. Systém zobrazí seznam načteného majetku určeného k inventuře ve výchozím stavu.

Podmínky pro ukončení

Uživateli je korektně zobrazen seznam majetku, který je určený k inventuře.

6.2.6 Případ užití – Zobrazit detail majetku

Případ užití umožňuje uživateli zobrazit detail majetku, tedy strukturované zobrazení jednotlivých vlastností daného objektu majetku.

Účastníci

- Uživatel
- Systém

Vstupní podmínky

V systému musí být načten soubor s majetkem určeným k inventuře, majetek je pak zobrazen v seznamu majetku.

Základní tok

1. Systém spustí případ užití *Zobrazit Seznam majetku*.
2. Uživatel klikne (gesto dotyku) na objekt majetku, u kterého chce zobrazit jeho detail.
3. Systém zobrazí detail daného majetku, který obsahuje strukturovaný výpis jeho vlastností a tlačítka určené k dalším operacím s příslušným objektem majetku.

Podmínky pro ukončení

Korektní zobrazení detailu objektu majetku.

6.2.7 Případ užití – Snímat kód určitého majetku

Případ užití umožňuje uživateli nad konkrétním objektem majetku snímat jeho čárový kód a následně jej porovnat s vedenou evidencí.

Účastníci

- Uživatel
- Systém

Vstupní podmínky

V systému musí být načten soubor s majetkem určeným k inventuře, majetek je pak zobrazen v seznamu majetku.

Základní tok

1. Systém spustí případ užití *Zobrazit detail majetku*.
2. Uživatel vybere příslušnou volbu pro snímání čárového kódu majetku v detailu konkrétního objektu majetku.
3. Systém zpřístupni uživateli řežím kamery/fotoaparátu pro sejmutí čárového kódu.
4. Uživatel sejme čárový kód z majetku.
5. Systém porovná snímaný kód se sériovým, evidenčním a inventárním číslem daného objektu majetku a v případě že údaj souhlasí, umožní uživateli majetek označit jako nalezený.

Alternativní tok 1

5.1 – V případě, že majetek je nalezen v evidenci, ale neodpovídá danému umístění, uživateli nebo středisku, systém zobrazí okno s odpovídajícím varováním a doporučením upravit hodnoty daných vlastností.
5.2 – Uživatel vlastnosti opraví a pokračuje bodem 2. základního toku.

Alternativní tok 2

5.3 – Pokud majetek s daným sériovým, inventárním nebo evidenčním číslem nebyl v evidenci nalezen, systém zobrazí okno s odpovídajícím chybou a nutností kontroly daných údajů u objektu majetku. Dokud uživatel údaj neopraví a neprojde celým základním tokem, systém mu nedovolí označit majetek jako nalezený.

Podmínky pro ukončení

Nalezení odpovídajícího majetku v evidenci.

6.2.8 Případ užití – Zobrazit majetek střediska

Umožňuje uživateli zobrazit v seznamu majetku všechen majetek dle hodnoty vlastnosti *Středisko* konkrétního objektu majetku.

Účastníci

- Uživatel
- Systém

Vstupní podmínky

V systému musí být načten soubor s majetkem určeným k inventuře, majetek je pak zobrazen v seznamu majetku.

Základní tok

1. Systém spustí případ užití *Zobrazit detail majetku*.
2. Uživatel klikne (gesto dotyku) v rozbaleném detailu objektu majetku na volbu *Majetek střediska* umožnující zobrazit majetek daného střediska.
3. Systém zobrazí Seznam majetku s automaticky vyplněným filtrem nad příslušných sloupcem *Středisko*, tzn. majetek stejného střediska jako vybraný objekt majetku.

Alternativní tok

3.1 – Pokud vybraný majetek nespadá pod žádné středisko, systém vrátí okno chybového hlášení informující o dané skutečnosti a daný případ užití je ukončen.

Podmínky pro ukončení

Korektní zobrazení majetku střediska vybraného objektu nebo chybové hlášení informující o absenci hodnoty ve vlastnosti *Středisko* vybraného objektu majetku.

6.2.9 Případ užití – Zobrazit majetek uživatele

Umožňuje uživateli zobrazit v seznamu majetku všechen majetek dle hodnoty vlastnosti *Uživatel* konkrétního objektu majetku.

Účastníci

- Uživatel
- Systém

Vstupní podmínky

V systému musí být načten soubor s majetkem určeným k inventuře, majetek je pak zobrazen v seznamu majetku.

Základní tok

1. Systém spustí případ užití *Zobrazit detail majetku*.
2. Uživatel klikne (gesto dotyku) v rozbaleném detailu objektu majetku na volbu *Majetek uživatele* umožnující zobrazit majetek daného uživatele.
3. Systém zobrazí Seznam majetku s automaticky vyplněným filtrem nad příslušných sloupcem *Středisko*, tzn. majetek stejného uživatele jako vybraný objekt majetku.

Alternativní tok

3.1 – Pokud vybraný majetek nespadá pod žádného uživatele, systém vrátí okno chybového hlášení informující o dané skutečnosti a daný případ užití je ukončen.

Podmínky pro ukončení

Korektní zobrazení majetku uživatele vybraného objektu nebo chybové hlášení informující o absenci hodnoty ve vlastnosti *Uživatel* vybraného objektu majetku.

6.2.10 Případ užití – Zobrazit majetek v umístění

Umožňuje uživateli zobrazit v seznamu majetku všechen majetek dle hodnoty vlastnosti *Umístění* konkrétního objektu majetku.

Účastníci

- Uživatel
- Systém

Vstupní podmínky

V systému musí být načten soubor s majetkem určeným k inventuře, majetek je pak zobrazen v seznamu majetku.

Základní tok

1. Systém spustí případ užití *Zobrazit detail majetku*.
2. Uživatel klikne (gesto dotyku) v rozbaleném detailu objektu majetku na volbu *Majetek v umístění* umožnující zobrazit majetek v daném umístění.
3. Systém zobrazí Seznam majetku s automaticky vyplněným filtrem nad příslušných sloupcem *Umístění*, tzn. majetek ve stejném umístění jako vybraný objekt majetku.

Alternativní tok

3.1 – Pokud vybraný majetek nespadá do žádného umístění, systém vrátí okno chybového hlášení informující o dané skutečnosti a daný případ užití je ukončen.

Podmínky pro ukončení

Korektní zobrazení majetku v umístění jako vybraný majetek nebo chybové hlášení informující o absenci hodnoty ve vlastnosti *Umístění* vybraného objektu majetku.

6.2.11 Případ užití – Upravit majetek

Umožňuje uživateli upravit jednotlivé vlastnosti konkrétního objektu majetku

Účastníci

- Uživatel
- Systém

Vstupní podmínky

V systému musí být načten soubor s majetkem určeným k inventuře, majetek je pak zobrazen v seznamu majetku.

Základní tok

1. Systém spustí případ užití *Zobrazit detail majetku*.
2. Uživatel klikne (gesto dotyku) v rozbaleném detailu objektu majetku na volbu *Upravit* umožnující úpravy vlastností vybraného objektu majetku.
3. Systém zobrazí stránku aplikace *Upravit majetek* umožnující úpravy jednotlivých vlastností vybraného objektu majetku.
4. Uživatel provede potřebné změny hodnot vlastností objektu majetku a potom je potvrdí příslušnou volbou.
5. Systém zkontroluje data od uživatele a poté mu vrátí okno s úspěšným uložením provedených změn.
6. Uživatel okno potvrdí.
7. Systém uživatele přesměruje zpět na zobrazení detailu daného objektu majetku.

Alternativní tok 1

4.1 – Pokud uživatel provede potřebné změny hodnot vlastností objektu majetku a ale poté je nepotvrdí, ale naopak zruší přes příslušnou volbu, tok pokračuje bodem 7 základního toku.

Alternativní tok 2

5.1 – V případě, že jsou zadaná data od uživatele neplatná, systém uživatele na skutečnost upozorní a nedovolí mu data uložit.
5.2 – Uživatel opraví zadaná data, která jsou neplatná a tok pokračuje bodem 4. základního toku.

Podmínky pro ukončení

Korektní uložení potřebných změn ve vlastnostech vybraného objektu nebo zrušení zadaných změn příslušnou volbou.

6.2.12 Případ užití – Export majetku

Případ užití umožňuje uživateli po provedené inventuře vyexportovat objekty majetku zpět do souboru k následnému opětovnému importu do konzolové aplikace pro správu majetku.

Účastníci

- Uživatel
- Systém

Vstupní podmínky

Uživatel musí mít právo k zápisu dat v daném umístění, kam chce soubor uložit (lokální, sdílená nebo síťová složka), případně právo k zápisu do již konkrétního souboru.

Základní tok

1. Systém zobrazí okno pro výběr cílového umístění ukládaného souboru a také pro zadání jeho názvu.
2. Uživatel vybere umístění ukládaného souboru, jeho název buď nechá výchozí nebo zvolí vlastní a okno potvrdí.
3. Systém uloží majetek z aplikace do daného souboru a vrátí uživateli okno potvrzující úspěšné uložení.

Alternativní tok

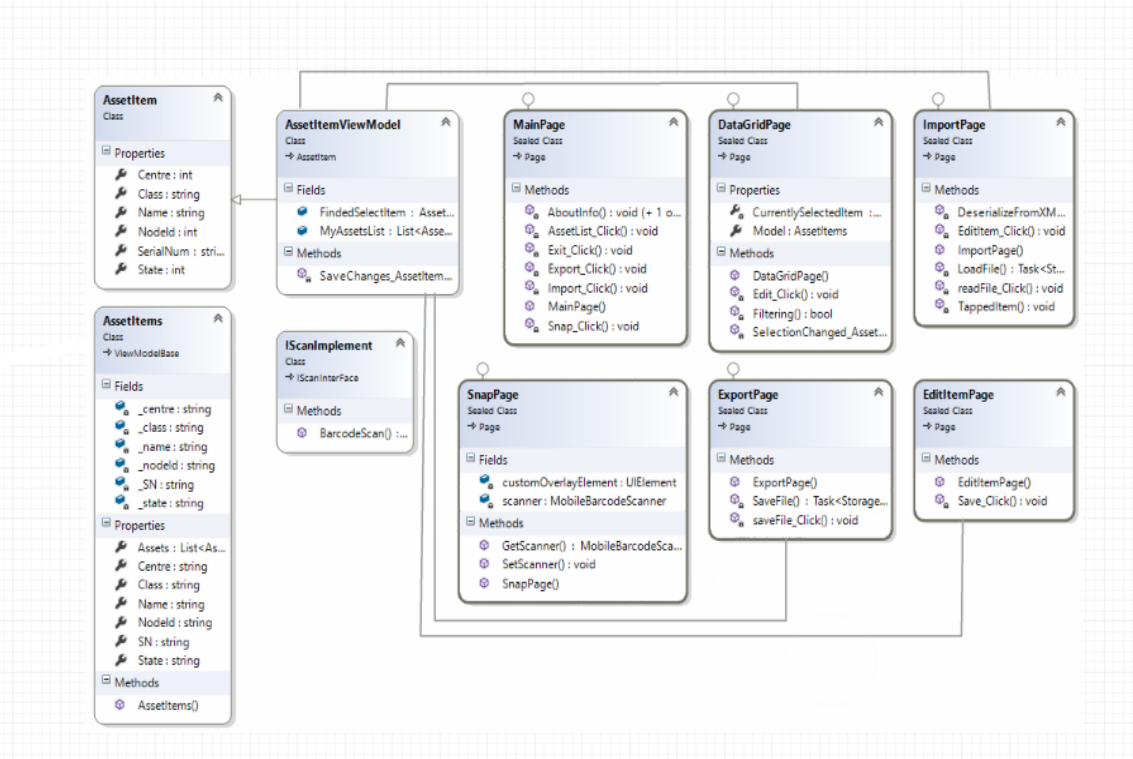
3.1 – Pokud uživatel nemá právo k zápisu ve vybraném umístění pro uložení souboru, případně se uložení majetku do souboru z jiného důvodu nepodaří, systém zobrazí uživateli zprávu o chybě s výzvou na kontrolu jeho oprávnění.
3.2 – Uživatel provede danou kontrolu a pokračuje na bodu 2. základního toku.

Podmínky pro ukončení

Úspěšný export inventarizovaného majetku do zvoleného souboru a úložiště.

6.3 Diagram tříd

Diagram tříd navrhované aplikace byl navrhnut přímo ve vývojovém prostředí Visual Studio 2015 community, které bylo později využito i pro implementaci. Dále lze říci, že diagram tříd navrhované aplikace je velíce specifickým případem, jelikož samotný typ aplikace a její vnitřní logika se nevyznačuje výraznými vazbami mezi jednotlivými třídami, kromě jednoduchých asociací mezi třídami ViewModelu a jednotlivých Views a samozřejmě vazby mezi ViewModelem a Modelem aplikace. Možnou příčinou je i využití návrhového vzoru MVVM, kde vazby mezi jednotlivými částmi aplikace často také nemusí být jednoznačně zřetelné. Diagram tříd navrhované aplikace lze vidět na obrázku 5 níže. (U některých tříd nejsou zobrazeny některé méně důležité atributy z důvodu jejich velkého množství a tedy úspory místa.)



Obrázek 5: Diagram tříd aplikace

Jednoznačně největší zastoupení v diagramu tříd navrhované aplikace mají třídy reprezentující jednotlivé Views aplikace, které mají na starosti vykreslení jednotlivých stránek UI navrhované aplikace. Konkrétně se jedná o třídy s příponou *Page*. Většina metod těchto tříd jsou tzv. event handlers[7]

Jednou ze tříd patřících do View je třída *MainPage*, která zobrazuje základní menu aplikace, obsahuje tedy hlavně metody pro obsluhu událostí typu kliknutí/gesto dotyku na tlačítka hlavního dlaždicového menu. Další důležitou relativně obsáhlou třídu je třída *DataGridPage*, která zajišťuje zobrazení Gridu, tedy seznamu majetku a jeho související funkcionalitu filtrování, řazení apod., čemuž se podrobněji budeme věnovat v kapitole o implementaci navrhované aplikace. Ostatní třídy části Views aplikace zajišťují zobrazení výsledků dalších funkcionalit aplikace, jak již jejich název napovídá. *ImportPage* a *ExportPage* vykreslují UI a obsluhují události pro import a export majetku, *SnapPage* UI pro snímání kódu a konečně *EditPage* vykresluje UI a také obsluhuje události související s editací objektu majetku.

Co se týče části ViewModelu ta je zastoupena třídou *AssetItemViewModel*, která slouží k uchování datové části v aplikaci a zároveň obstarává možnost editace datové složky aplikace, tedy možnost úprav jednotlivých vlastností objektů majet-

[7]Metody sloužící k obsluze událostí. V okamžiku, kdy je některá událost vyvolána, provede se kód její obslužné metody (event handler).

ku. Druhým zástupcem části ViewModelu je třída *IScanImplement*, která zajišťuje čistě vnitřní logiku snímání čárového kódu a spolupracuje tedy s View *SnapPage*.

Zbývající třídy reprezentují Model navrhované aplikace. Jde o třídu *AssetItem*, která charakterizuje jeden objekt majetku a obsahuje veškeré jeho možné vlastnosti. Za určitou pomocnou třídu lze považovat *AssetItems*, která je potřeba pro operaci serializace a deserializace objektů AssetItem z/do XML souboru. Je tedy nutnou součástí pro možnost importu majetku do aplikace a po provedení inventury opětovného export majetku z aplikace do souboru XML.

7 Implementace vlastního řešení

V následující kapitole rozebereme samotnou implementaci navrhovaného řešení. Nejprve zmíníme využité programové prostředky a zvolené technologie vývoje. Dále popíšeme její jednotlivé funkce jednak z pohledu jejich programového řešení (pohled na zdrojový kód) a také z pohledu řešení designového, tedy jak jednotlivé funkce může využívat uživatel provádějící fyzickou inventuru majetku.

7.1 Zvolené technologie vývoje a programové prostředky

Pro implementaci mobilní aplikace pro inventuru majetku jsem zvolil Framework .NET. Tuto volbu jsem upřednostnil hlavně z důvodu bohaté základny rozsáhlých knihoven, kterými .NET disponuje. Zároveň se jedná o relativně bezpečnou technologii, která se stále velice rychle vyvíjí.

V případě programovacího jazyku jsem dal přednost jazyku C#, který by svými možnostmi měl plnohodnotně posloužit k požadovaným výsledkům. Jelikož je navrhovaná aplikace pro platformu Windows, jevilo se jako nejvhodnější zvolit i vývojové prostředí od společnosti Microsoft. Přijatelnou volbou bylo Visual Studio Community 2015, tedy bezplatná edice nejnovější verze tohoto vývojového prostředí.

7.2 Responzivní design – Univerzální aplikace

Mobilní aplikace je navrhována jako univerzální, tzn. že její jednotlivé prvky uživatelského rozhraní se automaticky přizpůsobí různým velikostem displeje a rozlišení na různých mobilních zařízeních. I když tyto vlastnosti mají jednotlivé prvky UI již do určité míry automaticky implementovány, často je potřeba využít ještě dalších nástrojů dostupných při vývoji těchto typu aplikací.

Třída VisualStateManager

Velice užitečná třída podporující responzivní design aplikací a je součástí již zmiňovaného XAML jazyka. Její podstata je postavena na dvou velice účinných API sadách, pomocí níchž lze definovat určité vizuální stavy aplikace za konkrétních podmínek. Jedná se o *VisualState.StateTriggers* a *VisualState.Setters*, již z názvu je možná patrné jejich použití.

První jmenované triggery jsou postaveny na tzv. *AdaptiveTrigger*, který má k dispozici vlastnosti *MinWindowWidth* a *MinWindowHeight*, jedná se tedy o definici minimální šířky/výšky displeje v pixelech. Druhé jmenované settery pak slouží vyloženě k definici hodnot vlastností jednotlivých elementů uživatelského rozhraní, při různých velikostech displeje. Celý mechanismus se až nápadně podobá známému principu podmíněného příkazu IF <podmínka> THEN <příkazy>..., kde je v tomto případě část <podmínka> zastoupena jednotlivými triggery a část <příkazy> jsou provedeny pomocí setterů.

RelativePanel

Prostřednictvím zmiňovaného *VisualStateManageru* lze dynamicky ovlivňovat relativně velké množství vlastností jednotlivých UI elementů jazyka XAML. Nicméně tímto způsobem nelze docílit přímo komplexního responzivního designu, kde jsou prvky uživatelského rozhraní a jejich umístění závislá vyloženě jeden na druhém a jejich umístění je tzv relativní. Požadované chování nabízí *RelativePanel*. V kombinaci s vlastnostmi, které lze využít relativně k jinému objektu, lze docílit kýženého efektu komplexního responzivního designu. Velice užitečnými vlastnostmi v tomto případě jsou obecně známé Margin a Padding a také další vlastnosti, které nabízí přímo třída *RelativePanel* pro dané použití, např. Below, LeftOf nebo RightOf, které přímo definují pozici jednoho elementu oproti druhému (15).

Použití v navrhované aplikaci

Jak bylo zmíněných mechanismů využito v navrhované aplikace, lze vidět na ukázce kódu na obrázku 6. Daná ukázka je součástí View hlavního menu aplikace, které je tvořeno dlaždicemi umožnující uživateli zvolit jednotlivé funkce aplikace. Definujeme zde pozice dlaždic při třech různých minimálních šířkách displeje zařízení v pixelech. Dané tři definice dělíme do tzv. *VisualState*.

První z nich s názvem *NarrowState* definuje základní rozložení hlavního menu aplikace, tedy rozložení hlavního menu na zařízeních se šířkou displeje v intervalu 0-639 pixelů, což je definováno v příslušném triggeru. Následně pomocí zmiňovaných setterů definujeme relativní pozici jednotlivých dlaždic, na což v tomto případě stačí atribut *Margin*. V následujících dvou případech pro zařízení se středním rozlišením displeje *MediumState*(VGA až po HD) a velkým rozlišením displeje *WideState*[8] neboli širokým zobrazením (FullHD a více) jsou s výhodou využity navíc zmírňované vestavěné atributy RelativePanelu *RightOf* a *Below*. Samozřejmě využití triggeru je velice účinné i při jednoduché změně orientace zobrazení ze zobrazení na výšku na zobrazení na šířku neboli z Portrait na Landscape, což patří mezi základní funkce všech současných chytrých zařízeních.

[8]Tato definice je v ukázce zastoupena pouze příslušným komentářem jelikož je postavena na stejném principu jako předcházející definice a případný ukázkový obrázek zdrojového kódu by pak dosahoval zbytečně rozsáhlých rozměrů.

```
<VisualStateManager.VisualStateGroups>
    <VisualStateGroup>
        <VisualState x:Name="NarrowState">
            <VisualState.StateTriggers>
                <AdaptiveTrigger MinWindowWidth="0" />
            </VisualState.StateTriggers>
            <VisualState.Setters>
                <Setter Target="Header.Margin" Value= "0,15,0,10" />
                <Setter Target="Snap.Margin" Value="43,85,179.667,-185" />
                <Setter Target="AssetList.Margin" Value="186,85,9.2,-185" />
                <Setter Target="Import.Margin" Value="43,248,179,-348" />
                <Setter Target="Export.Margin" Value="186,248,31,-348" />
                <Setter Target="About.Margin" Value="43,411,179,-511" />
                <Setter Target="Exit.Margin" Value="185,411,32,-511" />
                <Setter Target="Footer.Margin" Value= "0,-120,0,11" />
            </VisualState.Setters>
        </VisualState>
        <VisualState x:Name="MediumState">
            <VisualState.StateTriggers>
                <AdaptiveTrigger MinWindowWidth="640"/>
            </VisualState.StateTriggers>
            <VisualState.Setters>
                <Setter Target="Header.Margin" Value= "0,0,20,0" />
                <Setter Target="Snap.Margin" Value="80,65,179.667,-185" />
                <Setter Target="AssetList.Margin" Value="226,20,15,-230" />
                <Setter Target="Import.Margin" Value="80,155,179,-348" />
                <Setter Target="Export.Margin" Value="226,110,15,-393" />
                <Setter Target="About.(RelativePanel.RightOf)" Value="AssetList" />
                <Setter Target="About.Margin" Value="0,20,0,-230" />
                <Setter Target="Exit.(RelativePanel.RightOf)" Value="Export" />
                <Setter Target="Exit.Margin" Value="0,110,0,-393" />
                <Setter Target="Footer.Margin" Value="110,-620,50,-50" />
            </VisualState.Setters>
        </VisualState>
        <VisualState x:Name="wideState">
            <!-- Definice WideState -->
        </VisualState>
    </VisualStateGroup>
</VisualStateManager.VisualStateGroups>
```

Obrázek 6: Ukázka kódu mechanismu VisualStateManager

7.3 Výchozí obrazovka aplikace – Hlavní menu

Hlavní menu aplikace již bylo mírně zmíněno v předchozí kapitole. Nyní si jej ještě blíže rozebereme a popíšeme důvody právě této volby hlavního menu.

Prvotní myšlenkou při návrhu a implementaci hlavního menu aplikace byla možnost použití velice moderního a častého prvku mobilních aplikací s názvem *Split view*. Tento prvek také přímo podporuje responzivní design univerzálních aplikací. Jak už název napovídá, jedná se o prvek, který okno aplikace dělí na dvě části, kde jedna část je určena pro obsah dané stránky aplikace a druhá část zobrazuje menu aplikace. Toto menu může dynamicky dosahovat různých zobrazení právě v závislosti na velikosti displeje(rozlišení) cílového chytrého zařízení (16).

Nicméně při testování navržené aplikace využívající tento užitečný prvek a také rozsáhlé konzultaci s potenciálními cílovými uživateli a dalšími vývojáři jsem od tohoto řešení nakonec ustoupil a to z několika zásadních důvodů. Jako hlavní důvod se jeví silná náročnost navrhované aplikace na využitý prostor displeje cílového zařízení. Jednoduše lze tedy říci, že přidaná hodnota postranního menu při využití *Split view* není dostatečná vysoká na to, aby se jevilo jako efektivní, obětovat značné množství prostoru displeje právě tomuto elementu.

Samotná navigace v aplikaci totiž není natolik složitá, aby bylo potřeba daného postranního menu na každé dílčí stránce aplikace. Stejně tak při analýze uživatelských scénářů aplikace nebylo prokázáno, že by postranní menu bylo zásadním prvkem při užíváni aplikace, uživatel si bohatě vystačí s klasickou navigací pomocí šipky zpět. Zkrátka důležitější je pro uživatele při použití této aplikace maximální využití prostoru pro přímo obsahovou část dané dílčí stránky aplikace např. při zobrazení rozsáhlého seznamu majetku a operací s ním souvisejících, kde je o prostor spíše nouze, než že by bylo nutné mít k dispozici ještě případné postranní menu. Zároveň se aplikace použitím dlaždicového menu více přibližuje design trendům klasických Windows aplikací. Vzhled hlavního menu aplikace při zobrazení na výšku je k dispozici obrázku 12 v příloze B a při zobrazení na šířku na obrázku 13 v příloze C.

Co se týče definování stylů navrhované aplikace, pomocí kterých definujeme způsob zobrazení jednotlivých elementů uživatelského rozhraní. Výchozí definice jsou uloženy v samostatném souboru, který je pro to přímo určený a je uložen v příslušné složce *Styles*. Další specifické definice jsou obsaženy již v konkrétním souboru, v kterém jsou i poté použity.

7.4 Import a export majetku

Vstup a výstup aplikace je již zmiňovaný XML soubor obsahující objekty majetku. Práci s XML souborem, tedy i jeho načtení či zapsání do souboru umožňuje třída *Stream* a *XmlSerializer*, které jsou součástí jazyka C#. Jednotlivé objekty majetku jsou ze souboru namapovány na objekty třídy *AssetItem*, která je součástí Modelu aplikace. K namapování slouží také pomocná třída Modelu aplikace *AssetItems*.

K samotnému výběru souboru ať pro načtení či zápis zprostředkovává užitečná třída *FileOpenPicker*, která je opět součástí programovacího jazyka. Ukázka programového řešení importu majetku do aplikace pomocí výše uvedených tříd můžeme vidět v metodách *DeserializeFromXML* a *LoadFile* na obrázku 7 níže. Export majetku je pak založen na stejném principu, jen opačným způsobem. Místo čtení souboru a deserializace XML je využito zápisu a serializace do XML.

```
private async Task<AssetItems> DeserializeFromXML(StorageFile file)
{
    AssetItems assetItems = new AssetItems();
    try
    {
        XmlSerializer serializer = new XmlSerializer(typeof(AssetItems));
        var stream = await file.OpenStreamForReadAsync();
        StreamReader reader = new StreamReader(stream);
        assetItems = (AssetItems)serializer.Deserialize(reader);
        reader.Dispose();

    }
    catch (Exception e)
    {
        MessageDialog loadFileFailedDialog = new MessageDialog("Soubor se nepodařilo načíst," +
                                " proveďte prosím kontrolu struktury souboru a akci opakujte!");
        loadFileFailedDialog.Title = "Mobile Inventory";
        loadFileFailedDialog.Commands.Add(new UICommand("Ok"));

        await loadFileFailedDialog.ShowAsync();
        return assetItems;
    }
    return assetItems;

}
private async Task<StorageFile> LoadFile()
{
    StorageFile file;
    FileOpenPicker openPicker = new FileOpenPicker();
    openPicker.FileTypeFilter.Add(".ixm");
    file = await openPicker.PickSingleFileAsync();
    return file;
```

Obrázek 7: Namapování objektů z XML pomocí XML serializace

7.5 Zobrazení seznamu majetku a detailu majetku

Zobrazení seznamu majetku hraje velice důležitou roli při provádění mobilní inventury pomocí navrhované aplikace. Pro zobrazení kvalitního seznamu majetku s danou funkcionalitou je potřeba využít tzv. DataGrid, tedy komponentu umožnující zobrazení seznamu určitých dat a nad tímto seznamem provádět operace jako řazení, filtrování, grupování apod. V aplikacích typu WPF nebo Windows Forms je k tomuto účelu k dispozici *DataGrid* nebo *DataGridView*, bohužel v případě UWP aplikací podobná komponenta chybí. Využil jsem tedy alespoň open-source knihovnu *MyToolkit.Extended*, která nabízí jednoduchý DataGrid (17).

Sám DataGrid je založen na mechanismu události *PropertyChanged*, která pomocí interface *INotifyPropertyChanged* dokáže sledovat a případně reagovat na změnu stavu jakékoliv vlastnosti objektu Modelu aplikace. díky tomu se veškeré změny v Modelu aplikace ihned promítnou do vlastností objektů majetku v seznamu majetku. Podobným způsobem lze mechanismu využít i pro filtrování nad daným seznamem. Jak lze vidět na obrázku 8, daného mechanismu využíváme také ke sledování změn v poli pro filtrování, jakmile, jedno z daných vstupních polí obsahuje nějaké znaky, v seznamu majetku promítne pouze majetek odpovídající filtru.

```
public DataGridPage()
{
    InitializeComponent();
    Model.PropertyChanged += (sender, args) =>
    {
        if (args.IsProperty<AssetItems>(m => m.Centre) || args.IsProperty<AssetItems>(m => m.User)
        || args.IsProperty<AssetItems>(m => m.Place) || args.IsProperty<AssetItems>(m => m.Class)
        || args.IsProperty<AssetItems>(m => m.Name))
        {
            DataGrid.SetFilter<AssetItem>(p => Filtering(p, Model));
        }
    };
}

private bool Filtering(AssetItem item, AssetItems model)
{
    return (item.Name ?? "").ToLower().Contains((model.Name ?? "").ToLower()) &&
           (item.User ?? "").ToLower().Contains((model.User ?? "").ToLower()) &&
           (item.Place ?? "").ToLower().Contains((model.Place ?? "").ToLower()) &&
           (item.Class ?? "").ToLower().Contains((model.Class ?? "").ToLower()) &&
           (item.Centre ?? "").ToLower().Contains((model.Centre ?? "").ToLower());

}
```

Obrázek 8: Programové zpracování filtrování nad seznamem majetku

Vykresleni DataGridu seznamu majetku má na starosti dané View využívající zmiňovaný jazyk XAML. Pomocí tzv. *Data Bindingu* jsme schopni relativně jednoduše zobrazit data jednotlivých objektů aplikace. Tento mechanismus hojně využíváme napříč celou aplikaci. V případě, že chceme pouze data zobrazit, využijeme *One-Way* Binding, pokud však potřebujeme data zobrazit, ale zároveň i nabídnout možnost úprav, musíme využít Binding typu *Two-Way*, který je k tomu přímo určen. Ukázka použití v navrhované aplikaci je zobrazen na obrázku 9 níže.

```
<AutoSuggestBox Grid.Column="0" Text="{Binding Name, Mode=TwoWay,
    UpdateSourceTrigger=PropertyChanged}"
                PlaceholderText="Název"  />
<AutoSuggestBox Grid.Column="1" Text="{Binding User, Mode=TwoWay,
    UpdateSourceTrigger=PropertyChanged}"
                PlaceholderText="Uživatel"/>
<AutoSuggestBox Grid.Column="2" FontSize="10" Text="{Binding Place, Mode=TwoWay,
    UpdateSourceTrigger=PropertyChanged}"
                PlaceholderText="Umístění"/>

<AutoSuggestBox Grid.Column="3" Text="{Binding Centre, Mode=TwoWay,
    UpdateSourceTrigger=PropertyChanged}"
                PlaceholderText="Středisko"/>

<AutoSuggestBox Grid.Column="4" Text="{Binding Class, Mode=TwoWay,
    UpdateSourceTrigger=PropertyChanged}"
                PlaceholderText="Třída"  />
```

Obrázek 9: XAML Data Binding pro filtrování v seznamu majetku

Další možností, kterou zmiňovaný DataGrid umožňuje, je zobrazení detailu vybraného majetku v seznamu majetku, což se do navrhované aplikace velice hodí, jelikož zpravidla není reálné zobrazit všechny evidované vlastnosti jednotlivých objektů majetku v seznamu majetku a to hlavně z důvodu nedostatku místa na mobilních zařízeních. Zároveň je detail majetku vhodným místem pro umístění dalších dostupných operací nad vybraným majetkem, jako jsou: možnost upravit majetek, snímat čárový kód daného objektu majetku, zobrazit majetek uživatele, zobrazit majetek v umístění a zobrazit majetek střediska. Zobrazení majetku v DataGridu lze vidět na obrázku 14 v příloze D a následné zobrazení detailu majetku na dalším obrázku 15 v příloze E (18).

7.6 Snímaní čárového kódu a další podpůrné funkce

Snímání čárového kódu lze považovat za hlavní funkci navrhované mobilní aplikace pro inventuru majetku. Opět pro tento účel existuje několik knihoven, které by bylo možné použít. Jako nejlepší kandidát pro navrhovanou aplikaci se jevila knihovna *ZXing.Net* s licenci Apache License 2.0, která není překážkou použití pro dané účely. V případě vývoje UWP aplikací jsou tyto rozšiřující knihovny dostupné ve formě tzv. NuGet package, které lze do projektu ve Visual Studiu nahrát přímo přes příslušný příkazový řádek, určený právě pro tento účel. Na obrázku 10 níže je zobrazena metoda zavádějící prostředí pro snímání kódu (19).

```
private MobileBarcodeScanner _scanner;
private async void MainPage_Loaded(object sender, RoutedEventArgs e)
{
    _scanner = new MobileBarcodeScanner(this.Dispatcher);
    _scanner.UseCustomOverlay = false;
    _scanner.TopText = "Namiřte kameru přímo na čárový kód";
    _scanner.BottomText = "Kamera automaticky kód sejme\r\n\rPro zrušení" +
                          " použijte tlačítko zpět";

    var result = await _scanner.Scan();
    ProcessScanResult(result);
}
```

Obrázek 10: Inicializační metoda pro snímání čárového kódu

Vnitřní události jako jsou spuštění fotoaparátu chytrého zařízení a také sejmutí čárového kódu má v režii daná externí knihovna, na nás je pouze obsloužení příslušného objektu typu *MobileBarcodeScanner*,případná editace jeho dalších orientačních prvků a samozřejmě vyhodnocení sejmutého čárového kódu, které je implementováno v metodě *ProcessScanResult*. Prostředí pro snímání čárového kódu vidíme na obrázku 11.

Obrázek 11: Prostředí snímání čárového kódu.

Ve zmiňované metodě provádějící vyhodnocení sejmutého kódu se nejprve zkontroluje, zda dodaný výsledek snímání kódu obsahuje nějaká data, v případě že ano, je uložen jako klasický řetězec string. Následně je provedeno porovnání naskenovaného řetězce se sériovými, evidenčními a inventárními čísly objektů majetku určených

k inventuře. Poté v případě, že nebyl majetek nalezen, je o dané skutečnosti uživatel informován a pokud byl majetek nalezen, je zavolána příslušná metoda pro zpracování v závislosti na místě spuštění snímání kódu, jestliže bylo snímání spuštěno z Hlavního menu aplikace je zavolána metoda *SnapAsset*, která uživatele přesměruje na detail nalezeného majetku, naopak pokud je snímání majetku spuštěno nad konkrétním majetkem, příslušná metoda *ScanCurrentAsset* zpracuje výsledek dle scénáře v podkapitole 6.2.7.

Další funkcionalita aplikace neobsahuje již nikterak složitou nebo zajímavou implementaci, jedná se o možnost upravit majetek, což je uživateli umožněno přes příslušnou volbu a následné zobrazení formuláře s jednotlivými vlastnostmi konkrétního objektu majetku a poté možnost uložení změn nebo jejich zrušení. Funkce zobrazení majetku uživatele, v umístění a střediska jsou implementovány dynamických dosazením hodnoty do příslušného filtru, pomocí čeho je dosaženo zobrazení kýženého majetku.

8 Zhodnocení výsledků

8.1 Shrnutí

Práce se zabývá problematikou fyzické inventury majetku pomocí mobilní aplikace disponující čtečkou čárového kódu a poté její následnou implementací pro Univerzální Windows platformu (UWP). Návrh, implementace a také testování aplikace je prováděno v prostředí dané firmy.

V kapitole 2 je obecně popsána inventarizace v dané společnosti. Především se zde rozebírá současný stav inventury majetku a poté současná aplikace pro fyzickou inventuru majetku. Popsány jsou její klíčové vlastnosti, některá omezení a zásadní nedostatky.

V kapitole 3 je provedena analýza technologie vývoje univerzálních Windows aplikací. Je zde popsána samotná Univerzální Windows platforma a veškeré její přednosti. Dále se kapitola podrobněji věnuje dostupným technologiím vývoje a to technologii C++ a Win32 a technologii využívající Framework .NET.

Vůbec obecné vymezení pojmu a definice inventury a inventarizace majetku je provedeno v kapitole 4. Dále popisuje druhy inventarizací, způsoby provedení inventury a také stukturu majetku pro inventarizaci.

Kapitole 5 byla provedena analýza scénářů použiti navrhované mobilní aplikace a jednotlivé scénáře zde byly neformálně popsány. Vždy pomocí požadavku od uživatele a následném způsobu řešení vaplikaci.

Praktická část je započata návrhem vlastního řešení kapitola 6. Zde je nejprve popsána architektura aplikace pomocí návrhového vzoru MVVM a poté formálně popsány případy užití aplikace pomocí diagramu případu užití a specifikace jeho jednotlivých případu užití. Rozebrán je i diagram tříd a také příslušnost jednotlivých tříd do konkrétní části aplikace z pohledu návrhového vzoru. V kapitole 7 je již popsána implementace vlastního řešení aplikace. Popisujeme zde zvolené programové prostředky a technologie. Dále také vlastnosti aplikace, které z ní dělají univerzální aplikaci, tedy řešení responzivního designu v případě daného typu aplikací. Praktická část práce je ukončena ukázkami a popisem programového řešení jednotlivých funkcí aplikace a v neposlední řadě také ukázkami uživatelského rozhraní výsledné aplikace.

8.2 Diskuze

Výsledkem práce je univerzální mobilní Windows 10 aplikace pro inventuru majetku pomocí chytrého zařízení. Aplikace byla vytvořena pomocí programovacího jazyka *C#* a Frameworku *.NET*. Zásadní funkcí aplikace je snímání čárového kódu objektu majetku a následné vyhodnocení porovnávání s evidovanými daty. Nicméně neméně důležitou funkcí je i co nejvíce seznamu majetku a příslušnými funkcemi, podporující orientaci v něm a to zejména vzhledem k omezení související s možnou velikostí displeje cílového chytrého zařízení. Prototyp aplikace bude sloužit dané společnosti jako náhrada stávající aplikace pro fyzickou inventuru, která byla určena pouze

pro zařízení typu Pocket PC/MDA s vestavěnou čtečkou kódů a s operačními systémy Windows Mobile/CE.

Během návrhu a implementace aplikace bylo nutné překonat několik překážek. Značnou překážkou byla absence hlubší znalosti a zkušenosti technologií vývoje pro platformu Windows a i zvoleného programovacího jazyka jako takového. Avšak tato překážka byla překonána rozšířeným studiem dané problematiky a odbornou konzultací se spolupracovníky dané společnosti, kteří v daném prostředí vyvíjí software již několik let. Dalším překážkou při implementaci byla absence některých důležitých mechanismů programovacího jazyka v případě vývoje právě aplikací pro Univerzální Windows platformu. Tyto nedostatky byly překonány využitím rozšířených open-source knihoven dané platformy.

Při testování první verze aplikace se ukázalo, že potenciálním problémem může být nedostatek místa na cílových zařízeních s menší velikostí displeje a to hlavně při práci v seznamu majetku, kde nelze místo v podstatě žádným relevantním způsobem ušetřit. Ve zmiňované první verzi aplikace byla k dispozici navigace pomocí postranního panelu, ale po opakované analýze a konzultaci s cílovými uživateli bylo nakonec rozhodnuto od tohoto řešení ustoupit a zvolit spíše klasickou Windows navigaci pomocí dlaždicového menu.

Za hlavní přínos vytvořené aplikace oproti současně používané aplikaci lze jednoznačně považovat její relativně velký krok vpřed z pohledu modernizace a to v podstatě v rámci celé aplikace. Vytvořením aplikace pro Univerzální Windows platformu byl celkově posunut vpřed uživatelský zážitek z užívání aplikace. Aplikace disponuje moderními prvky a funkcemi uživatelského rozhraní, které podporuje intuitivní používání aplikace. Zároveň je určena pro více typů chytrých zařízeních, tzn. že uživatel již není vázán pouze na jedno zařízení. Kromě toho byly rozšířeny a vylepšeny i některé další funkce, kterými již disponuje současně používaná aplikace, jako např. rozšířená možnost filtrování, či zobrazení majetku uživatele, v umístění nebo střediska. Zároveň byla snaha i o případné zjednodušení používání aplikace.

Jistě je zde prostor i pro další vylepšení, které se nachází např. v umožnění uložení použitých filtrů a řazení v seznamu majetku či možnosti širších úprav zobrazení seznamu majetku, jako zobrazení pouze sloupců dle vlastní volby nebo editace jejich pořadí. Dalším vylepšením bylo mohlo být také již zmiňované online řešení, tzn. aplikace by pracovala online přímo nad cílovou databází, i když v tomto případě by se jednalo o značně složité vylepšení, jak již bylo řečeno v kapitole 2.2.1.

Realizovaná mobilní aplikace není nikterak komerčním řešením a je šitá na míru k použití spolu se zmiňovanou konzolovou aplikací pro správu majetku. Splňuje však stanovená kritéria a podporuje svůj případný budoucí vývoj.

9 Závěr

Cílem této práce bylo navrhnout vlastní řešení pro mobilní aplikaci sloužící k inventarizaci majetku a na základě tohoto návrhu poté provést implementaci daného řešení univerzální Windows 10 aplikace.

Cíl práce byl splněn, navržená a poté vytvořená aplikace pro inventuru majetku odpovídá potřebám provádění fyzické inventury a nabízí funkcionalitu k danému účelu určenou. Aplikace byla podrobena základním testováním dle uvedených scénářů použití aplikace. Avšak její testování i nadále probíhá, z důvodu eliminace všech případných chyb a problému při použití i méně častých uživatelských scénářů.

10 Literatura

[1] Inventura. *ALVAO Documentation Library.* [online] Žďár nad Sázavou: ALVAO, 1999 [cit. 2016-11-10] Dostupné z:
https://doc.alvao.com/support/doc/cs/alvao_8_2/list_of_windows/alvao_asset_management_console/tools/modules/stocktaking/default.aspx

[2] ALVAO Mobile Inventory. *ALVAO Documentation Library.* [online] Žďár nad Sázavou: ALVAO, 1999 [cit. 2016-11-10] Dostupné z:
https://doc.alvao.com/support/doc/cs/alvao_8_2/modules/alvao_mobile_inventory/default.aspx

[3] Intro to the Universal Windows Platform. *Learn to Develop with Microsoft Developer Network | MSDN.* [online] USA: Microsoft, © 2016 [cit. 2016-11-10] Dostupné z:
https://msdn.microsoft.com/en-us/windows/uwp/get-started/universal-application-platform-guide

[4] Windows 10 Development Introduction. *Simply easy learnings | TutorialsPoint.* [online] Indie: TutorialsPoint, © 2016 [cit. 2016-11-14] Dostupné z:
https://www.tutorialspoint.com/windows10_development/windows10_development_introduction.htm

[5] Choose Your Technology (Windows). *Learn to Develop with Microsoft Developer Network | MSDN.* [online] USA: Microsoft, © 2016 [cit. 2016-11-14] Dostupné z:
https://msdn.microsoft.com/en-us/library/windows/desktop/dn614993(v=vs.85).aspx

[6] XAML overview. *Learn to Develop with Microsoft Developer Network | MSDN.* [online] USA: Microsoft, © 2016 [cit. 2016-11-14] Dostupné z:
https://msdn.microsoft.com/en-us/windows/uwp/xaml-platform/xaml-overview

[7] Microsoft .NET - .NET and Universal Windows Platform Development. *Learn to Develop with Microsoft Developer Network | MSDN.* [online] USA: Microsoft, © 2016 [cit. 2016-11-22] Dostupné z:
https://msdn.microsoft.com/en-us/magazine/mt590967.aspx

[8] Strnad, A. *Inventarizace koncových stanic ve střední firmě do 250PC.* [online] Unicorn College Praha: Bakalářská práce, 2014 [cit. 2016-11-22] Dostupné z:
http://prev.unicorncollege.cz/european-it-center/strnad-antonin/attachments/BP_Strnad_Antonin.pdf.

[9] Mališková, I. *Využití logistického informačního systému při provádění inventarizace majetku a zásob ve firmě.* [online] UTB Zlín: Bakalářská práce, 2011 [cit. 2016-11-23] Dostupné z: http://digilib.k.utb.cz/handle/10563/16874.

[10] Ministerstvo financí ČR. *Úplné znění zákona č. 563/1991 Sb., o účetnictví. 1991, část pátá.* [online] Ministerstvo financí [cit. 2016-11-24] Dostupné z: http://www.mfcr.cz/assets/cs/media/Zak_1991-563_UZ-zakona-o-ucetnictvi-s-vyznacenim-zmen.pdf

[11] Inventura a inventarizace. *Podnikátor - pomůže Vám v podnikání | Podnikátor.* [online] Podnikátor.cz, © 2012 [cit. 2016-11-28] Dostupné z: http://www.podnikator.cz/provoz-firmy/ucetnictvi-a-dane/danova-evidence/n:16972/Inventura-a-inventarizace

[12] Svoboda, P. a Bohušová, H. Inventarizace majetku a závazků *Účetní kavárna – Čerstvé informace. Horké diskuse. - Komunitní portál účetních expertů.* [online] Praha: Wolters Kluwer ČR, © 2016 [cit. 2016-11-28] Dostupné z: http://www.ucetnikavarna.cz/archiv/dokument/doc-d2820v3638-inventarizace-majetku-a-zavazku.

[13] Schiffer, V. *Inventarizace v praxi: otázky a odpovědi.* Praha: Grada, 2006. Účetnictví a daně (Grada). ISBN 80-247-1921-5.

[14] The MVVM Pattern. *Learn to Develop with Microsoft Developer Network | MSDN.* [online] USA: Microsoft, © 2016 [cit. 2016-12-23] Dostupné z: https://msdn.microsoft.com/en-us/magazine/mt590967.aspx

[15] Make your app look great on any size screen or window (10 by 10) - Building Apps for WindowsBuilding Apps for Windows. *Windows Blog.* [online] USA: Microsoft, © 2016 [cit. 2016-11-22] Dostupné z: https://blogs.windows.com/buildingapps/2015/09/01/make-your-app-look-great-on-any-size-screen-or-window-10-by-10/#FjptUiL1dypoBgEu.97

[16] Split view. *Learn to Develop with Microsoft Developer Network | MSDN.* [online] USA: Microsoft, © 2016 [cit. 2016-12-23] Dostupné z: https://msdn.microsoft.com/en-us/windows/uwp/controls-and-patterns/split-view

[17] GitHub - MyToolkit/MyToolkit: MyToolkit for .NET. *How people build software · GitHub.* [online] USA: GitHub, Inc., © 2016 [cit. 2016-12-23] Dostupné z: https://github.com/MyToolkit/MyToolkit

[18] Troelsen, A. *Pro C# 5.0 and the .NET 4.5 framework.* New York: Apress, 2012. ISBN 978-1-4302-4233-8.

[19] ZXing.Net - Documentation. *ZXing.Net - Home.* [online] USA: Microsoft, Inc., © 2006-2017 [cit. 2016-12-23] Dostupné z: `https://zxingnet.codeplex.com/documentation`

Přílohy

A Inventarizace majetku a závazků

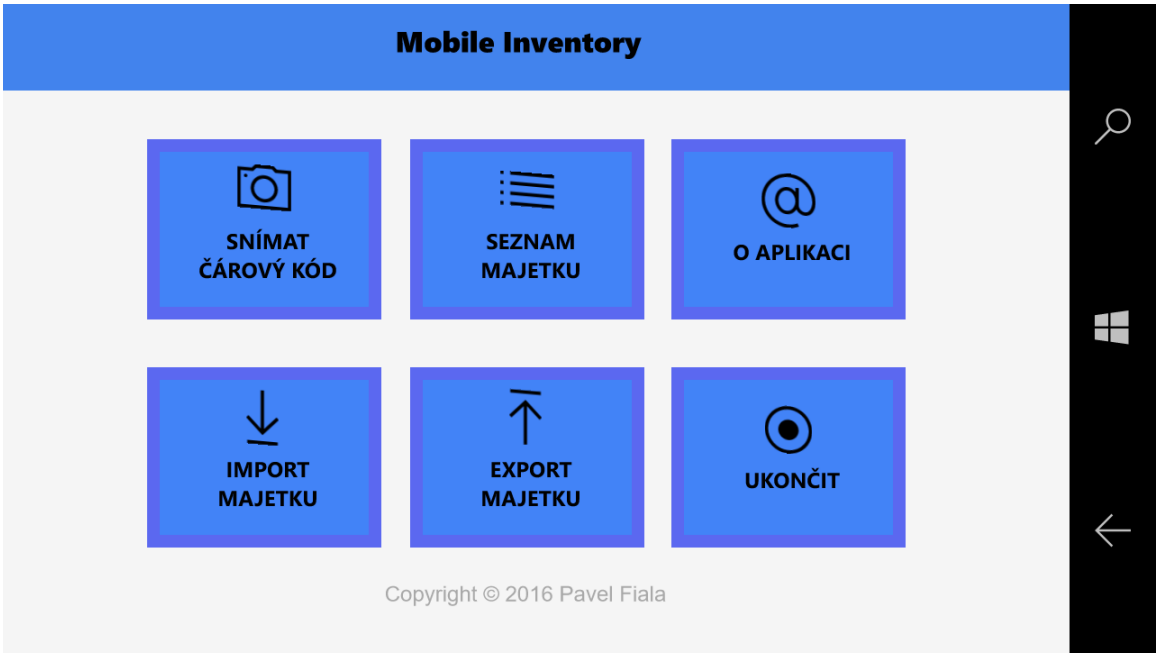
Tabulka 1: Provedení inventury pro různé druhy majetku a závazků

Položka (účtová skupina)	**Forma inventury**
01 – Nehmotný dlouhodobý majetek	Dokladová inventura. Pokud to umožnuje forma majetku (software, projekty, výsledky výzkumu či vývoje), rovněž fyzická inventura
02 – Hmotný odpisovaný majetek 21 – Peníze	Fyzická inventura
03 – Hmotný neodpisovaný majetek	Fyzická inventura, u pozemků fyzická i dokladová
04 – Nedokončený dlouhodobý majetek 05 – Zálohy na dlouhodobý majetek 07, 08 – Oprávky k dlouhodobému nehmotnému a hmotnému majetku 09 – Opravné položky k dlouhodobému majetku 22 – Účty v bankách 26 – Převody mezi finančními účty 29 – Opravné položky k finančnímu majetku 23, 46 – Krátkodobé a dlouhodobé bankovní úvěry 24 – Krátkodobé finanční výpomoci 31 – Pohledávky 32 – Krátkodobé závazky 33, 34 – Zúčtování se zaměstnanci a institucemi 34 – Zúčtování daní a dotací 35, 36 – Pohledávky a závazky ke společníkům, účastníkům sdružení a členům družstva 37 – Jiné pohledávky a závazky 38 – Přechodné účty aktiv a pasiv včetně dohadných účtů 39 – Opravné položky ke zúčtovacím vztahům 45 – Rezervy 47 – Dlouhodobé závazky 48 – Odložený daňový závazek a pohledávka	Dokladová inventura
06 – Dlouhodobý a krátkodobý finanční majetek	Dokladová inventura, s výjimkou cenných papírů v hmotné podobě
1 – Zásoby	Většinou fyzická inventur, s výjimkou záloh na zásoby, záloh na cestě a také opravných položek k zásobám.

B Hlavní menu při zobrazení na výšku

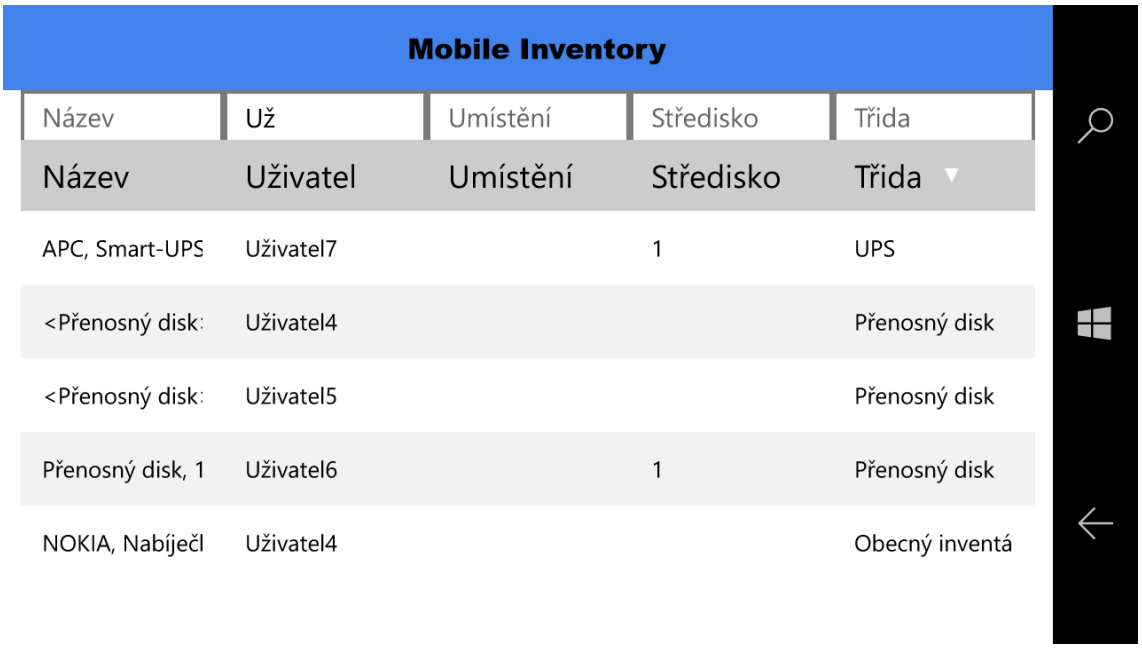
Obrázek 12: Hlavní menu při zobrazení na výšku (portrait)

C Hlavní menu při zobrazení na šířku



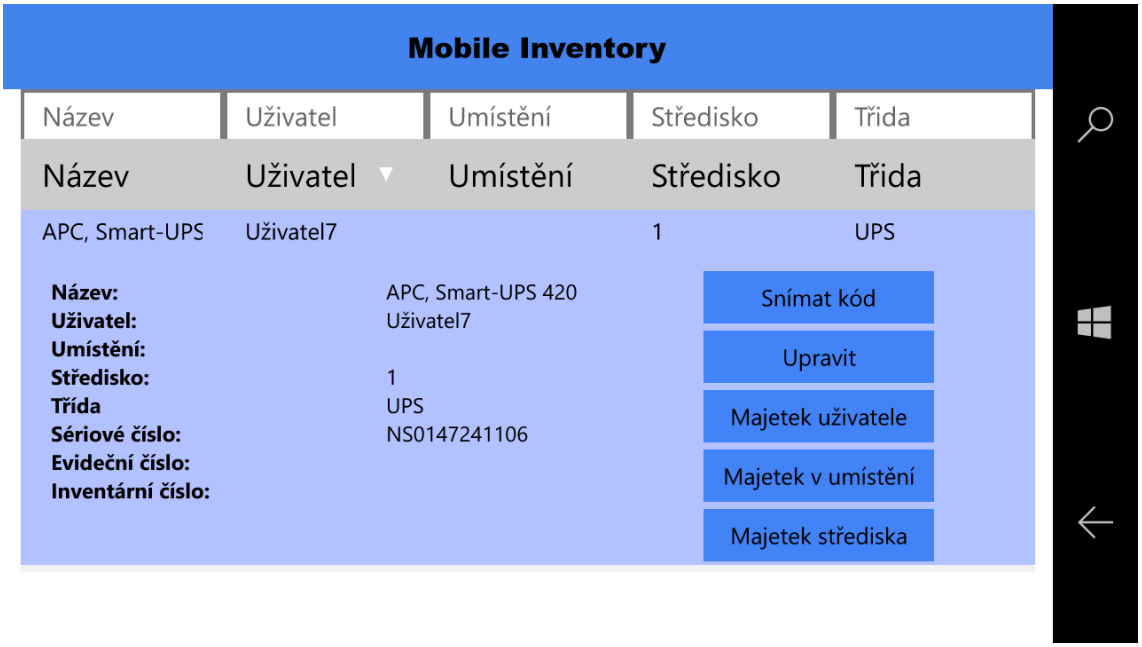
Obrázek 13: Hlavní menu při zobrazení na šířku (landscape)

D Zobrazení objektů majetku v DataGridu



Obrázek 14: Zobrazení objektů majetku v DataGridu

E Detail vybraného majetku v seznamu majetku



Obrázek 15: Zobrazení detailu majetku v DataGridu

F Obsah přiloženého DVD

Na přiloženém DVD jsou umístěna následující data:

- MobileInventory.zip – Přeložený spustitelný soubor MobileInventory.exe, určený pro mobilní zařízení s operačním systémem Windows 10.
- MobileInventory_solution.zip – Zdrojové soubory mobilní aplikace.