**ŽILINSKÁ UNIVERZITA V ŽILINE**

FAKULTA RIADENIA A INFORMATIKY

28360720191487

**Implementácia systému na správu tovaru v sklade s webovým rozhraním na prehľad aktuálne naskladnených položiek**

BAKALÁRSKA PRÁCA

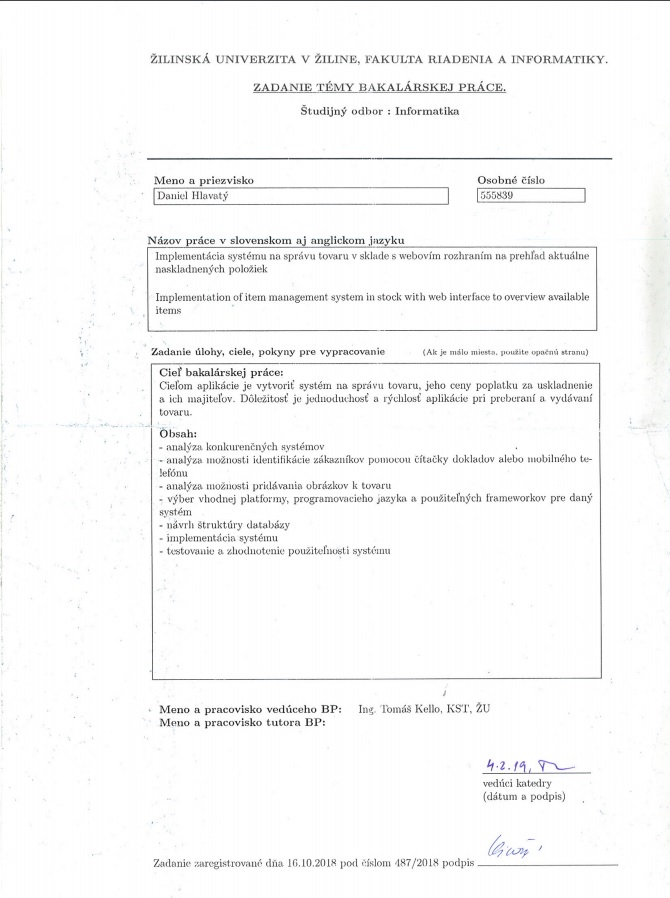
Daniel Hlavatý

Študijný odbor: Informatika

Vedúci práce: Ing. Tomáš Kello

Registračné číslo: 487/2018

Žilina, 2019



**Abstrakt v štátnom jazyku**

V bakalárskej práci bude vytvorený jednoducho ovládateľný systém na správu tovaru. Popíše postup, výber technológii a samotnú implementáciu systému. Výsledkom bude jednoduchá aplikácia na správu schopná pridávať a meniť údaje o tovare a zákazníkoch, a druhá je webová aplikácia, slúžiaca na prehliadanie dostupného tovaru na sklade pre verejnosť. Na záver bude práca otestovaná a zhodnotená.

Klúčové slová: WPF, .NET, C#, správa tovaru

**Abstrakt v angličtine**

In this bachelor work i will create simple user friendly system designed for goods management. I will describe steps i have taken, how i selected technology and write about this implementation of this system itself. Result will be simple aplication able to add and change product and customer data, and simple web application for viewing the currently available goods in a stock. At the end, i will test them and evaluate them.

Key words: WPF, .NET, C#, inventory management

Obsah

[Úvod 1](#_Toc7760737)

[1 Aktuálny stav. 2](#_Toc7760738)

[1.1 Príchod zákazníka 2](#_Toc7760739)

[1.2 Manipulácia tovaru 2](#_Toc7760740)

[1.3 Záznamy 2](#_Toc7760741)

[2 Analýza konkurenčných riešení 3](#_Toc7760742)

[2.1 ABC Inventory 3](#_Toc7760743)

[2.2 InFlow 4](#_Toc7760744)

[2.3 Vend 6](#_Toc7760745)

[3 Výber platformy, jazyka, frameworku a databázového systému 8](#_Toc7760746)

[3.1 Platforma, jazyk 8](#_Toc7760747)

[3.1.1 QT a C++ 8](#_Toc7760748)

[3.1.2 JAVA 9](#_Toc7760749)

[3.1.3 .NET 9](#_Toc7760750)

[3.2 Windows presentation foundation a WinForms TO DO UWP 10](#_Toc7760751)

[3.3 Databáza 11](#_Toc7760752)

[3.3.1 MySQL 12](#_Toc7760753)

[3.3.2 Microsoft SQL server 12](#_Toc7760754)

[3.3.3 PostgreSQL 12](#_Toc7760755)

[4 Návrh a implementácia aplikácie 14](#_Toc7760756)

[4.1 Vyhradenie funkcionality 14](#_Toc7760757)

[4.1.1 Akcie a vlastnosti zákazníka 14](#_Toc7760758)

[4.1.2 Akcie a vlastnosti tovaru 14](#_Toc7760759)

[4.1.3 Akcie a vlastnosti záznamov 14](#_Toc7760760)

[4.2 Možnosti pridávania fotiek, tlače a identifikácie zákazníkov 15](#_Toc7760761)

[4.2.1 Možnosti a problémy identifikácie zákazníkov 15](#_Toc7760762)

[4.2.2 Možnosti pridávania fotiek 16](#_Toc7760763)

[4.2.3 Možnosti tlače 16](#_Toc7760764)

[4.3 Databáza 17](#_Toc7760765)

[4.4 Návrh okien 17](#_Toc7760766)

[4.4.1 Hlavné okno 17](#_Toc7760767)

[4.4.2 Okno užívateľa 18](#_Toc7760768)

[4.4.3 Okno detailu tovaru 19](#_Toc7760769)

[4.4.4 Modálne okná na pridávanie 19](#_Toc7760770)

[4.4.5 Popup pri zmene údajov 19](#_Toc7760771)

[4.4.6 Material Design 19](#_Toc7760772)

[4.5 Implementácia a výsledná aplikácia 20](#_Toc7760773)

[4.5.1 Rozloženie ovládacích prvkov 20](#_Toc7760774)

[4.5.3 Modálne a popup okná 23](#_Toc7760775)

[4.5.5 Obrázky a FTP pripojenie 23](#_Toc7760776)

[4.5.6 Web kamera 24](#_Toc7760777)

[4.5.7 Tlač 24](#_Toc7760778)

[5 Webové rozhranie 26](#_Toc7760779)

[5.1 Použité technológie – TO DO ASP.net mvc web 26](#_Toc7760780)

[5.1.1 O Architektúre MVC 26](#_Toc7760781)

[5.1.2 Rozdiely medzi WPF a ASP.NET 28](#_Toc7760782)

[6 Testovanie 29](#_Toc7760783)

[6.1 Virtualizácia tabuliek 29](#_Toc7760784)

[6.2 Tlač a web kamera 29](#_Toc7760785)

[7 Záver 30](#_Toc7760786)

Zoznam použitých obrázkov

[Obr. č. 1 Ukážka skladom používanej faktúry 3](#_Toc7104741)

[Obr. č. 2 Ukážka ABC Inventory okna 5](#_Toc7104742)

[Obr. č. 3 Ukážka okna aplikácie inFlow 6](#_Toc7104743)

[Obr. č. 4 Ukážka okna aplikácie Vend 8](#_Toc7104744)

[Obr. č. 5 Kompilácia a beh .NET aplikácie 10](#_Toc7104745)

[Obr. č. 6 Príklad vytvorenia a úpravy elementu 12](#_Toc7104746)

[Obr. č. 7 Ukážka štandardne vyzerajúcej aplikácie vo WinForms.[3] 12](#_Toc7104747)

[Obr. č. 8 Ukážka štandardne vyzerajúcej aplikácie vo WPF 12](#_Toc7104748)

[Obr. č. 9 Prvotná predstava o databáze 13](#_Toc7104749)

[Obr. č. 10 Fotografia mena na občianskom preukaze, ako test OCR knižníc 15](#_Toc7104750)

[Obr. č. 11 Návrh databázy 18](#_Toc7104751)

[Obr. č. 12 Logické poradie prechodov medzi stavmi 19](#_Toc7104752)

[Obr. č. 13 Kruhový zapínač 20](#_Toc7104753)

[Obr. č. 14 Kruhový zapínač, pri prechode kurzorom myši 20](#_Toc7104754)

[Obr. č. 15 TextBox s “lietajúcim” popisom 20](#_Toc7104755)

[Obr. č. 16 TextBox s “lietajúcim” popisom po zakliknutí 20](#_Toc7104756)

[Obr. č. 17 Pridanie material designu do súboru app.xaml[7] 21](#_Toc7104757)

[Obr. č. 18 Pridanie material designu a jeho využitie v samostatných oknách 21](#_Toc7104758)

[Obr. č. 19 Animácia hamburger menu hlavného okna 21](#_Toc7104759)

[Obr. č. 20 Získanie databázy z verejného servera do objektu DataSet 22](#_Toc7104760)

[Obr. č. 21 Vytvorenie tabuľky s generovaným stĺpcom v XAML 22](#_Toc7104761)

[Obr. č. 22 Úprava objektu DataSet a jeho pridanie do DataGridu 22](#_Toc7104762)

[Obr. č. 23 Úprava stĺpca vygenerovaného XAML 22](#_Toc7104763)

[Obr. č. 24 Generovanie otváracieho riadku popisu v tabuľke 23](#_Toc7104764)

[Obr. č. 25 Otváranie/Zatváranie riadku popisu, vyhľadávanie pomocou FindVisualChild 23](#_Toc7104765)

[Obr. č. 26 Príklad využitia XAML spúšťača 23](#_Toc7104766)

[Obr. č. 27 Príklad štýlovania buniek objektu dataGridu pomocou XAML 24](#_Toc7104767)

[Obr. č. 28 Ukážka Popup okna 24](#_Toc7104768)

[Obr. č. 29 Ukážka generovania názvu obrázku 25](#_Toc7104769)

[Obr. č. 30 Ukážka sťahovania a nahrávania obrázkov na server 25](#_Toc7104770)

[Obr. č. 31 Ukážka zachytenia obrázku z dostupného z web kamery pomocou EMGU CV knižnice 25](#_Toc7104771)

[Obr. č. 32 Príklad načítanie a zápis dát do PDF dokumentu pomocou knižnice FreeSpirePDF 26](#_Toc7104772)

[Obr. č. 33 Príklad tlače pdf dokumentu pomocou knižnice FreeSpirePDF 26](#_Toc7104773)

[Obr. č. 34 Vzhľadom na dôležitosť tlače faktúr, prikladám ukážku jednej vyplnenej aplikáciou 26](#_Toc7104774)

[Obr. č. 35 Ukážka webového rozhrania 27](#_Toc7104775)

[Obr. č. 36 Schéma MVC 28](#_Toc7104776)

[Obr. č. 37 Ústrižok html kódu vytvárajúceho tabuľku pohľadu, s tlačidlom otvor 28](#_Toc7104777)

[Obr. č. 38 Príklad Javascript metódy pridávajúcej nový element do stránky, a AJAX metódy komunikujúcej z kontrolérom 29](#_Toc7104778)

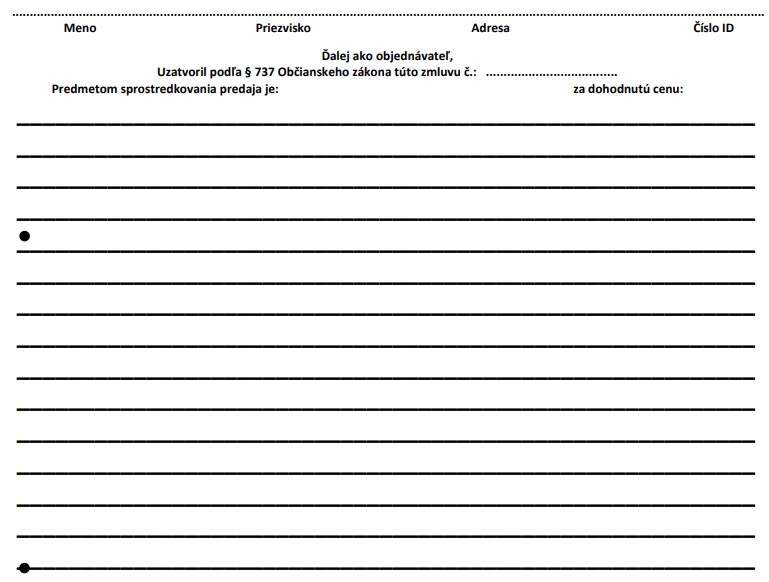
# Úvod

V súčasnej dobe sklady ktoré ešte stále používajú papierovú formu organizácie pomocou faktúr, postupne nezastaviteľne prechádzajú na organizáciu formou aplikácií, nakoľko výrazne zlepšujú produktivitu skladníka. Vývojom takejto aplikácie sa zaoberá táto bakalárska práca. Niektorým podnikom prechod trvá dlhšie, nakoľko na prácu s aplikáciami je potrebné mať zvládnuté základy práce s počítačom a tiež je potrebné všetky faktúry prepísať do informačného systému.

V prvých dvoch častiach sa zameriame na analýzu momentálneho behu skladu a na analýzu konkurenčných riešení ako InFlow, čím si vytvoríme predstavu o funkcionalite našej aplikácie. V tretej časti opíšeme teoreticky výber možných jazykov, platforiem či frameworkom ako .NET, ktoré by sa hodili pre tento systém na základe už známej potrebnej funkcionality. Nasledujúce dve časti sa zaoberajú samotným návrhom a implementáciou aplikácie aj webového rozhrania. Na záver aplikáciu testujeme, a zhodnotíme kvalitu náhrady papierového fungovania skladov s informačným systémom.

# 1 Aktuálny stav.

Analýzou momentálneho fungovania skladu hľadáme najväčšie nedostatky behu skladu, ktoré nám pomôžu pri určovaní funkcionality samostatných riešení. Sklad je organizovaný pomocou faktúr, pri čom každý zákazník má svoju špecifickú do ktorej skladník zapisuje jeho osobné údaje a tovar sním spojený.



Obr. č. 1 Ukážka skladom používanej faktúry

Fungovanie skladu sme rozdeliil na samostatné časti a ich funkcie sme opísali v nasledujúcich odsekoch.

## Príchod zákazníka

Tovar nosia na sklad postupne rôzni zákazníci, hneď pri ich príchode skladník do prázdnej faktúry zapíše meno, identifikačný kľúč a kontaktné údaje. Sú zoradené podľa kľúča, a pri opakovanom príchode zákazníka skladník ručne postupne prehľadáva faktúry až kým nenájde hľadanú.

## Manipulácia tovaru

Ak skladník hľadá tovar podľa kľúča zákazníka, opakuje rovnaký postup, len zapisuje iné údaje. Problém však nastáva, keď skladník dohľadáva tovar(napr. chce ho vrátiť). Faktúra je označená len kľúčom zákazníka nie tovaru, preto skladník musí prechádzať cez všetky faktúry hľadajúc chcený tovar, čo sa to stáva veľmi prácnym.

## Záznamy

Momentálny faktúrový systém neponúka majiteľovy skladu dohľad nad prácou skladníka, ktorá môže byť ovplyvňovaná ľudským faktorom.

Cieľom aplikácie je teda urýchlenie behu systému skladu takým spôsobom, že budeme minimalizovať čas strávený skladníkom vyhľadávaním údajov z faktúr, automatickým vypisovaním a tlačou faktúr a umožnenie jednoduchej kontroly práce skladníkov majiteľovi.

# 2 Analýza konkurenčných riešení

Cieľom analýzy konkurenčných riešení je spoznať sa so základnými dizajnovými štandardami, ako aj vybrať riešenia, ktoré riešia rovnaké alebo podobné problémy, ktoré máme aj my v našej aplikácií ako napríklad tlač faktúr, vyhľadávanie tovaru. Zhodnotíme ktoré prvky vyhovujú alebo nevyhovujú požiadavkám vyvíjanej aplikácie. Zoznam neobsahuje všetky voľne dostupné aplikácie riešiace problém inventárového manažmentu, ale vzorka dobre poukazuje na rozdiely medzi samotnými známymi riešeniami.

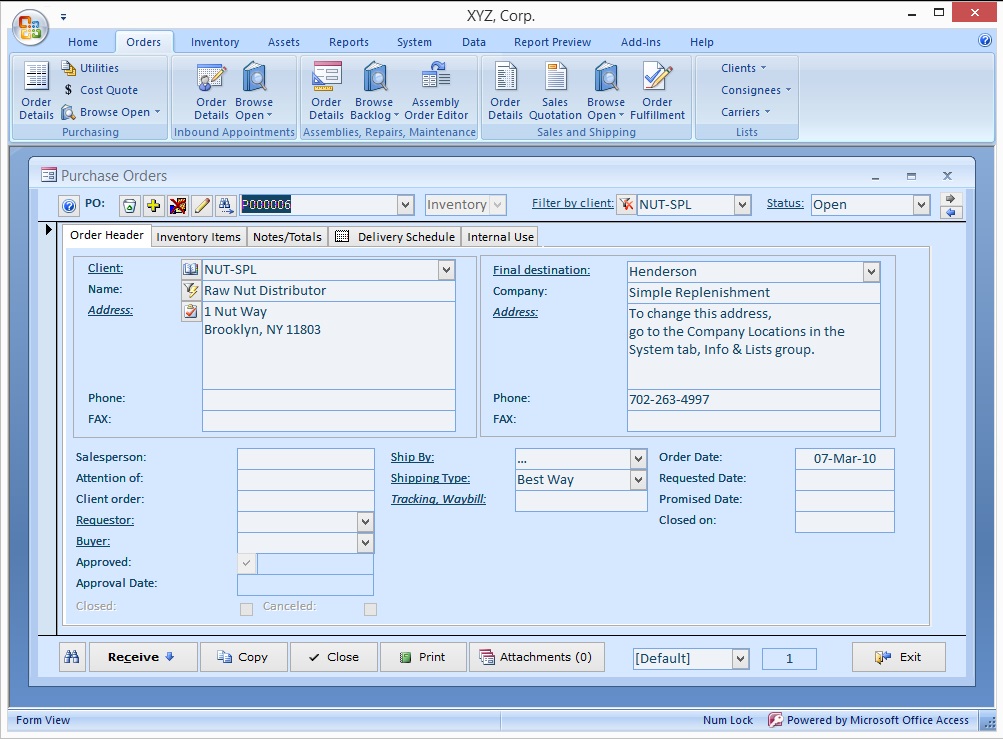
## 2.1 ABC Inventory

Jedno z najpopulárnejších riešení. Prvotný dojem aplikácie je príjemný vďaka vlažnej modrej farbe, ktorá nie je únavná pre oči pri dlhodobej práci. Druhotný dojem by som však tak nezhodnotil, aplikácia totiž ponúka mnoho skrytej funkcionality v malom obdĺžnikovom hornom menu ktoré pôsobí veľmi neprehľadne. Menu je výzorom podobné tomu známemu z Microsoft Word, avšak s dôležitým rozdielom. Jednotlivé položky (príklad. pre Word- page layout, view.. Pre ABC Inventory napr. tovar) sa nachádzajú vo viacerých záložkách. Skladník by pri prehľadávaní tovaru musel vedieť, či sa tovar nachádza na sklade alebo či je len napr. objednaný, čo je nevyhovujúce.

Zmena stavu objednávky sa vykonáva v samostatných oknách a pri každej zmene užívateľ musí doplniť údaje. Pri archivácií údaju je neprimerane zložité túto zmenu vrátiť. Pri pridaní nového tovaru alebo zmene údajov starého tovaru, z dizajnu tabuľky nie je jasné, či sú údaje už uložené alebo nie, rovnaké pri odchode z otvoreného okna, program neupozorní na neuložené zmeny, vyplnené údaje rovno miznú.

Vyhľadávanie tovaru je problémové, software neponúka zoznam všetkého tovaru, iba prehľad tovaru pre daného užívateľa. Neponúka žiadnu históriu úprav.

Z týchto vlastností je jasné že cieľom ABC Inventory bolo zahrnúť čo najväčšiu skupinu skladov či skladových riešení, ale za cenu jednoduchosti ovládania.



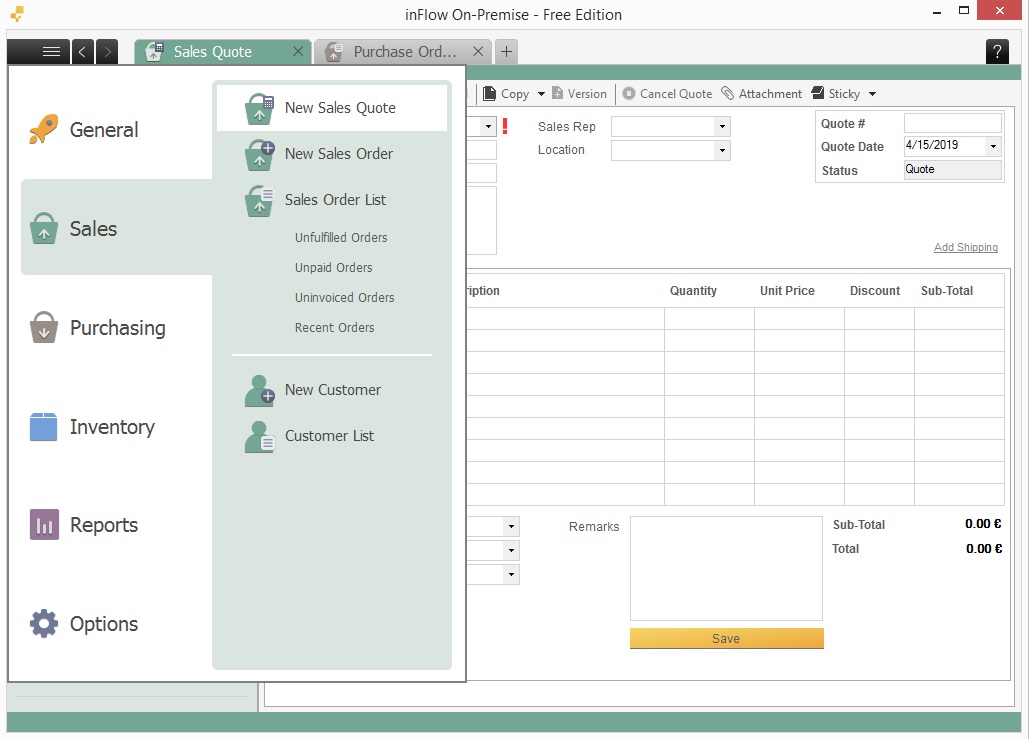
Obr. č. 2 Ukážka ABC Inventory okna

Aplikácia neponúka možnosť fotenia tovaru ani pridávania obrázkov, ale ponúka možnosť tlače. Užívateľ si vyberie objednávku, ktorú chce vytlačiť v danom okne a aplikácia automaticky doplní text do PDF dokumentu pripravenému k tlači. Na každej objednávke sa môže nachádzať len jeden tovar, čo môže spôsobiť nadmernú spotrebu papiera. Pri stlačení zapínača tlačenia aplikácia vyberie prvú aktívnu tlačiareň a tlačí, ak žiadna nie je pripojená, tlačidlo zostane stále svietiť ale pri stlačení vypíše chybu. Fakt, že tlačiť sa dá len v tomto okne môže byť pomalé, pretože užívateľ sa musí postupne preklikať.

Do našej aplikácie pravdepodobne odzrkadlíme fungovanie tlačenia, pretože funguje rýchlo, jednoznačne a jednoducho bez zásahu užívateľa. Vyhľadávanie tovaru by bolo ideálnejšie rozdeliť na dve rôzne miesta, samostatne pre špecifického zákazníka a samostatne všetok tovar s prideleným majiteľom. Musíme myslieť na to, že archivovaný tovar je možné aj znova zobraziť, či prípadne ďalej upravovať.

## 2.2 InFlow

Zadarmo riešenie postavené na Microsoft SQL servery. Pri prvom zapnutím aplikácie prebehnú prvé nastavenia, ako napríklad mena ceny. Pri vstupe vyskočí pomocník, ktorý zaberie väčšinu okna, teda nedá sa ignorovať. Táto vlastnosť môže byť dobrá pri programoch cielených pre väčšiu cieľovú skupinu. Firemné programy ako tento je obyčajne bez pomocníka, pretože bude ovládaný len osobou ktorá bude sním popredu zoznámená.



Obr. č. 3 Ukážka okna aplikácie inFlow

Pri vytváraní objednávky tovar musí byť pridaný naraz, čo znamená, že ak by skladník chcel dodatočne pridávať tovar, musel by vytvárať novú objednávku. V programe úplne absentuje možnosť archivácie tovaru ako aj jeho vyhľadávanie či tlač faktúr. Ovládacie prvky v samotných oknách sú rozložené prehľadnejšie, samotné okná aplikácie na seba obyčajne priamo nenadväzujú.

Hlavné menu aplikácie je štandardné hamburger menu, pôsobí omnoho úhľadnejšie ako horné lištové menu ABC Inventory. Nenachádza sa vo všetkých oknách, čím aplikácia uvoľňuje priestor a stáva sa prehľadnejšou.

Ponúka históriu, kreslí graf z nazbieraných údajov či uskutočnených zmien a umožňuje prehliadať zmeny údajov. Priamo k špecifickým údajom užívateľa alebo tovaru sa z tohto okna dostať nedá, teda skladník by po vyhľadaní tovaru na ktorej bola uskutočnená zmena musel tovar opätovne hľadať v prehľade.

Z dizajnu opäť nie je jasné či upravené údaje sú už uložené alebo nie. Popis tovaru sa nachádza v tabuľke vedľa ostatných údajov, avšak obyčajne býva dlhší, čo môže byť neprehľadné.

Aplikácia neponúka možnosť pridávania obrázkov. Ponúka možnosť tlače a to v akomkoľvek štádiu objednávky. Pri stlačení zapínača tlače vyskočí dialógové okno podobné tlači z programu Microsoft Windows Word o výbere tlačiarne a úpravách tlačeného dokumentu. Náš sklad má dostupnú len jednu tlačiareň, toto okno sa stáva zbytočným a tak ho nebudeme implementovať.

Hamburger menu je ideálnym úhľadným riešením aj pre našu aplikáciu, pri úpravách zákazníka alebo tovaru ho budeme taktiež schovávať pre sprehľadnenie aplikácie. Na rozdiel od ABC Inventory jednotlivé okná sú menej prepojené a ľahšie dostupné, čím sa aplikácia stáva ľahšie ovládateľnou. História tejto aplikácie má za úlohu zbierať všetky informácie o zákazníkoch a tovare, v našom prípade sledujeme len prácu skladníka a zmeny údajov ako napríklad  zmeny ceny. Z histórie by sa mohlo dať dostať priamo k tovaru ktorý bol upravený, pre jednoduchšiu zmenu späť.

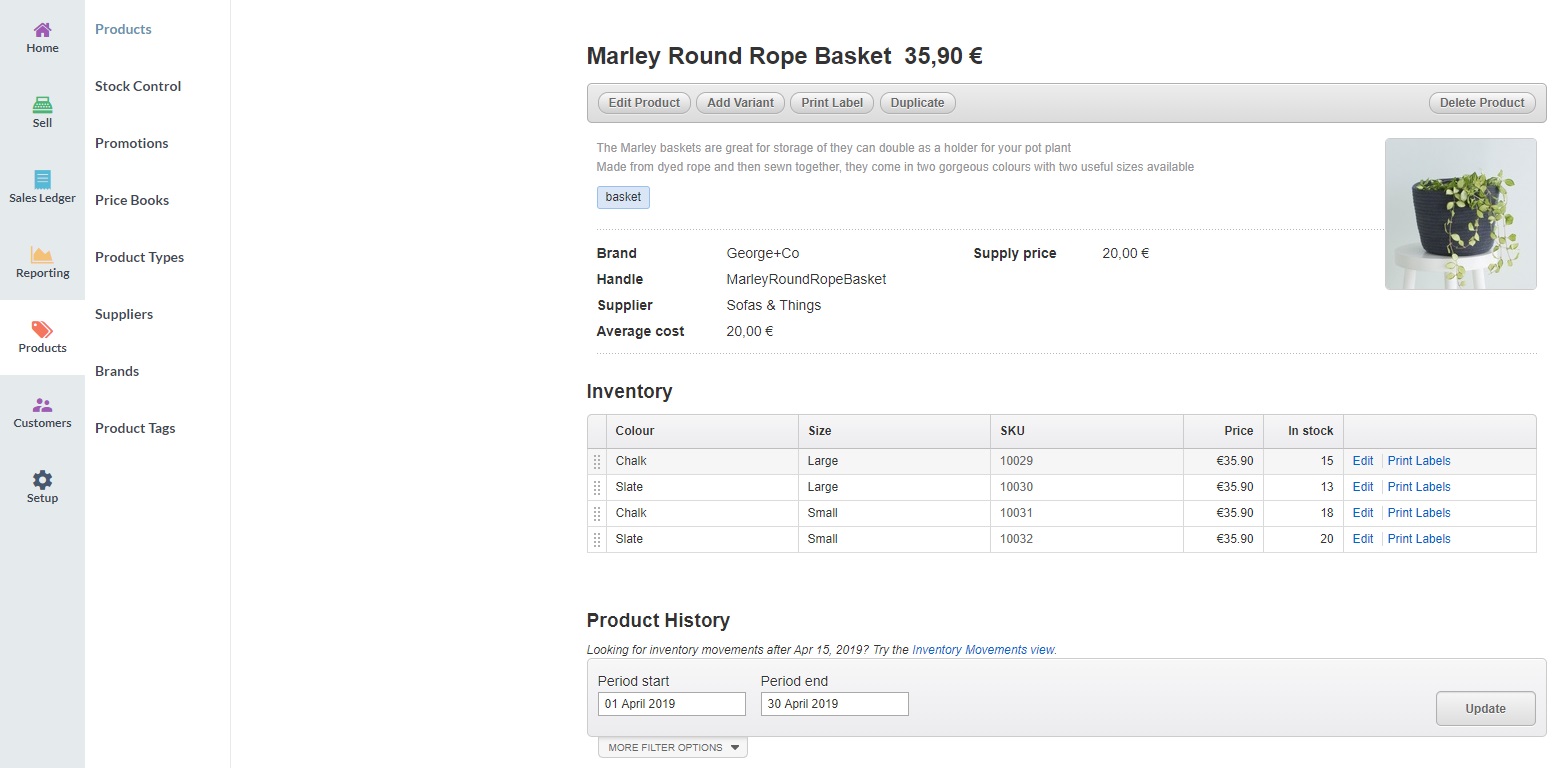
## 2.3 Vend

Prvá z uvedených systémov ktorá je web aplikáciou, čo by bolo ideálne, mohla by naraz umožňovať aj skladníkovu prácu aj webové rozhranie pre prehľad tovaru na sklade. Výhodou webovej aplikácie je kompatibilita, funguje všade kde funguje podporovaný prehliadač. Nevýhodou je že je neporovnateľne pomalšia ako ostatné konkurenčné napriek tomu že beží vo webovom prehliadači, napríklad, prechod medzi oknami tovaru a zákazníka je relatívne časovo náročná operácia, ktorá by mala byť takmer instantná.

Aplikácia neponúka možnosť fotenia tovaru, ale ponúka možnosť zmeny obrázku jednoducho spolu zo zmenami ostatných údajov. Obrázok tovaru je schovaný a otvorí sa až pri otvorení detailu tovaru. Má prehľadné hamburger menu, ktoré je aktívne vo všetkých oknách čo pomáha pri prechodoch medzi nimi.

Predaný tovar aplikácia nenávratne odloží preč z prehľadu čím uvoľňuje miesto, ale zabraňuje spätným zmenám. Umožňuje samostatný prehľad všetkého tovaru alebo tovaru na sklade, v samostatných oknách.

Výzor aplikácie je veľmi príjemný, pre prehľad tovaru môžeme odzrkadliť funkcionalitu prehliadania tovaru. Opakované hľadanie údajov pre skladníka by mohlo byť náročné, v tabuľkách v jednom z okien tovaru, napr. chýba vlastník.



Obr. č. 4 Ukážka okna aplikácie Vend

Po zhrnutí kladov a protikladov konkurenčných aplikácií, som sa inšpiroval najma po stránke dizajnu. Vadšina funkcionality v týchto aplikáciách sa nebude nachádzať v rovnakej forme v našej bakalárskej práci. Poučili sme sa rozdieloch medzi ovládaním okien pomocou spoločného menu, alebo samostatnými špecifickými prvkami daného okna.

# 3 Výber platformy, jazyka, frameworku a databázového systému

Aplikácia musí bežať na počítači skladníka, preto bude bežať len na operačnom systéme Windows. Pri výbere platformy jazyka a databázového systému myslíme na rýchlosť aplikácie potrebnú podporu ďalších zariadení (web kamera, čítanie občianskeho, tlač). Dôležité sú aj moje znalosti a inklinujem k technológiám, ktoré sú mi osobne bližšie, pretože boli vyučované na škole.

## 3.1 Platforma, jazyk

Spravil som menší prehľad momentálne najpoužívanejších platforiem ktoré sú obyčajne využívané na malé aplikácie s grafickým rozhraním. Pri výbere myslíme na to, že samotná rýchlosť jazyka či platformy je relatívne zanedbateľná pri porovnaní rýchlosti akcií ako napríklad prístup do databázy, pretože aplikácií väčšinu času zaberú operácie ako prístup do databázy alebo sťahovanie či nahrávaní súborov na vzdialený server.

### 3.1.1 QT a C++

C++ je jazyk známy svojou rýchlosťou, ale aj komplexnosťou. Napriek tomu, že je s ním perfektne možné vyvíjať aplikácie s grafickým rozhraním, jeho samotný kód je dlhší a detailnejší. Má dlhú históriu využívania v hernom priemysle, graficky najpokročilejšie hry sú obyčajne programované v tomto jazyku. Nevýhodou C++ aplikácie je že vyžadujú veľké množstvo kódu ak chceme aplikáciu vyvíjať od nuly. Nevýhodou je údržba kódu, ktorá si vyžaduje skúseného programátora. Ako bonus si dovolím spomenúť stránku StackOverflow.com, kde programátory chodia hľadať správne riešenia, má jazyk C++ najviac odpovedí (podľa stránky stackoverflow.com až 84.6%). V praxi sa obyčajne využíva pri vyvíjaní rozsiahlejších aplikácií, kde rýchlosť samotného kódu hrá väčšiu rolu.

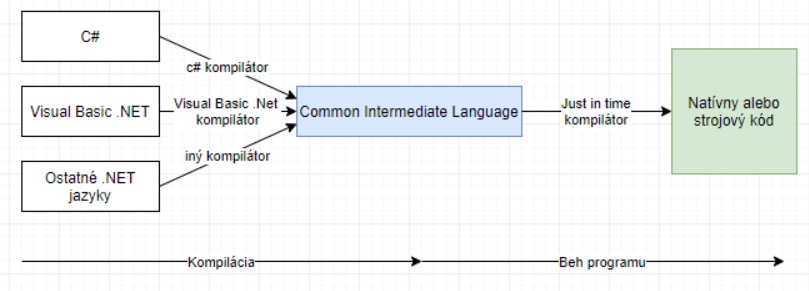
QT Je tiež multiplatformový, väčšinou je považovaný za grafickú knižnicu, čo je z časti pravda, v skutočnosti QT ponúka oveľa viac, obsahuje moduly na prácu s vláknami, skriptovanie, prácu z XML či 3D grafikou. Svojou rozsiahlou knižnicou by mohol uľahčiť prácu s web kamerou , čítaním kariet či tlačou. Na stránke stackoverflow.com má viac ako 60000+ zodpovedaných otázok (v roku 2015), QT má rozsiahlu komunitu čo môže byť výhodou. Dobrým príkladom možností QT, je aplikácia Skype. Ako mínus je často spomenutý Meta-Object Compiler(MOC), ktorý sa stará o QT rozšírenia. MOC prečíta C++ header súbory, ak nájde jeden alebo viac deklarácií obsahujúcich Q\_OBJECT a makrá. Vytvorí C++ source súbor obsahujúci meta objekty pre tieto triedy. Nepovažuje sa to ako štandardné, teda pri kompilácií QT vykonáva krok navyše. Ďalším mínusom je samotná veľkosť aplikácie, QT potrebuje na beh prostriedky väčších objemov ako konkurenčné riešenia. Teda po vykompilovaní aplikácie môže vzniknúť aplikácia ktorá je neprirodzene veľká.[1] Obyčajne využívaný pre stredne veľké a veľké systémy.

### 3.1.2 JAVA

Java ponúka tiež niekoľko zadarmo dostupných riešení, ako Swing či JavaFX. Štandardné používateľské okno pôsobí ďaleko od toho windowsového, čo by mohlo pôsobiť odpudivo na skladníka. Jeho výhodou je hlavne to, že ho vie “každý” programátor. Často je využívaný ako prvý vyučovaný jazyk, ktorý sa programátori učia, pretože má na pohľad prirodzenú syntax. Výhodou to môže byť vo väčších problematickejších problémoch, pretože obyčajne nie je problémom nájsť niekoho, kto by kód potom udržoval. Zmámym mýtom je, že jeho beh je pomalí, čo však nie je úplne pravda, tento deficit je viditeľný až pri veľkých komplexných riešeniach.[x] Využíva sa na programovanie multiplatformových riešení (ťažšie sa vytvára ale grafické prostredie na úrovni), čo je výhodou, ale naša aplikácia bude bežať len na jednom zariadení. Naše riešenie nie je multiplatformové a preto som si vybral inú možnosť.

### 3.1.3 .NET

.Net podporuje viacero jazykov ktoré si vie tento framework kompilovať do spoločného tzv. intermediate language pomocou Common Language Runtime (CLR). Touto vlastnosťou je výnimočný v porovnaní s ostatnými, nenúti využívať jeden špecifický jazyk. C# programátori môžu využívať všetok kód napísaný vo Visual Basic a opačne. Najpoužívanejšími sú C# a Visual Basic, niekedy pri vyvíjaní napríklad vnútornej logiky stránky sa používa F#, dokonca je možné využívať aj jazyk ako JAVA. C# a Visual Basic sú si schopnosťami veľmi podobné, líšia sa najmä v syntaxi. Pri kompilovaní iných aplikácií obyčajne kompilátor posiela kód podľa špecifického typu CPU. Pre 32-bitový Windows bude napríklad kompilovaný ako 486 CPU. To neznamená, že používatelia musia mať túto verziu CPU a preto aplikácie nevyužívajú plný potenciál dnešných CPU. Intermediate Language je spustený na každom CPU alebo operačnom systéme s nainštalovaným .NET frameworkom. Prikladám jednoduchú upravenú schému[2]



Obr. č. 5 Kompilácia a beh .NET aplikácie

Najväčším syntaxovým rozdielom sú takzvané kľúčové slová, napríklad variantom ukončenia C# kučeravou zátvorkou je Visual Basic kľúčové slovo End. Keďže aplikácia bude bežať len na operačnom systéme Windows, môžeme spolu s .NET využiť Windows Presentation Foundation alebo Windows Forms, ktoré pomôžu zjednodušeniu dizajnu, okná budú skladníkovi už výzorom a ovládaním známe, teda sa s ich používaním sa už nebude musieť oboznamovať.

.NET je výhodný nie len pre tvorbu aplikácii pre Windows, ktorý investuje veľké množstvo peňazí do marketingu frameworku.[x] Hlavným cieľom .NET aplikácie je využívať čistý kód a izolovať všetok ostatný do samostatných zostavení.(codeproject.com). Pod čistým kódom rozumieme také metódy, ktoré si berú využívajú hodnoty len zo vstupných parametrov, a zaujímajú nás len hodnoty ktoré vracia. Výhodou takéhoto kódu je fakt že beh metódy je závislý len na vstupných parametroch(tj. majme metódu int DajCislo(int type), môžeme o nej vždy povedať že DajCislo(1) = DajCislo(2)). Momentálne .NET funguje len pod operačným systémom Windows, avšak už existuje napríklad projekt Mono, ktorý dovoľuje beh .NET na Linuxe.

Napriek tomu, že nemám žiadne praktické skúsenosti s .NET frameworkom ani C# jazykom, vzhľadom na rozsiahle vyššie uvedené výhody .NET frameworku, som si ho vybral pretože pôsobí ako framework, s ktorým sa v budúcnosti pravdepodobne ešte stretnem.

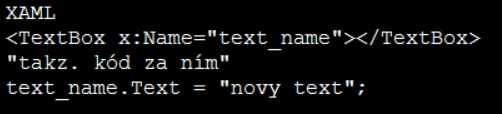
## 3.2 Windows presentation foundation, Universal Windows Platform a WinForms

Universal Windows Platform(UWP) je najnovší systém s predstavených, prvý krát pridaný až do Microsoft Windowsu 10. Využívaný je na vytváranie aplikácií bežiacich na Windows 10, Xbox One, HoloLens. Stačí aby programátor napísal kód len raz, a aplikácia bude bežať na všetkých podporovaných zariadeniach. Jeho limitáciou je fakt, že nie je možné ho využívať na starších verziách Windowsu, preto som sa rozhodoval medzi zvyšnými dvoma možnosťami.

Najväčším rozdielom medzi nimi je fakt, že WinForm je vlastne len vrstva postavená na štandardných windowsových ovládačoch(win controls). WPF nie je postavený na nich, a teda nie je na nich závislí. Môže to vyzerať ako nebadateľný rozdiel, ale v skutočnosti nie je. Ďalším rozdielom je fakt, že WPF nadväzuje na WinForms, a rozvíja ho hlavne po grafickej stránke. Neznamená to, že WinForms by nebol schopný pokročilého dizajnu ale vo WPF sa vytvára viac “po lopate”.

Dobrým príkladom je napríklad zapínadlo s obrázkom a textom. Pretože takáto kombinácia sa nenachádza v štandardných Windowsových ovládačoch WinForms to neumožňuje jednoducho, riešením by bolo implementovanie vlastného zapínača alebo využitia nejakého voľne dostupného. Vo WPF, zapínač môže obsahovať ľubovoľné prvky, pretože je to vlastne len rám (border) s obsahom (content). V tomto prípade by stačilo vytvoriť zapínač, a vytvoriť dnu ovládač textu a obrázku. Naša aplikácia má byť jednoduchá na ovládanie, a bude obsahovať množstvo tabuliek ktoré budú upravované, tieto zmeny sú jednoduchšie uskutočniteľné vo WPF.

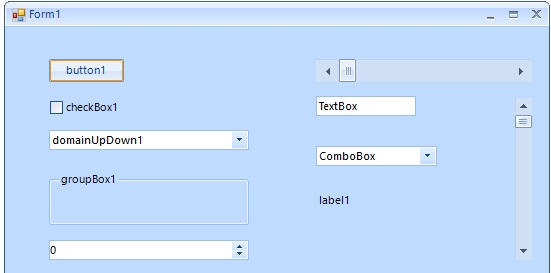
Výhodou WPF je aj špeciálny typ jazyka XAML, ktorý rozdeľuje dizajn a vnútornú stránku aplikácie, čo WinForms neponúka. Nie je zameraný na tvorbu algoritmov, skôr sa zameriava na usporiadanie a vytvorenie jednotlivých prvkov v grafickom rozhraní aplikácie. Takto vytvorené elementy môžeme ďalej upravovať v jednom z jazykov podporovaných .NET frameworkom. Uvádzam jednoduchý príklad vytvorenia a úpravy TextBoxu.



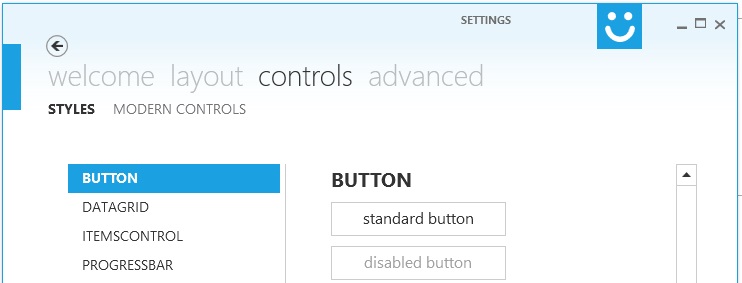
Obr. č. 6 Príklad vytvorenia a úpravy elementu

WPF podporuje vykreslenie na rozdiel od WinForms vektorovo, pri extrémnej úprave veľkostí nevzniká rušivá pixelácia.

WPF z týchto informácií pôsobí ako jednoznačný nasledovník WinForms a preto som si vybral túto technológiu. Prikladám porovnanie štandardne vyzerajúcich aplikácií oboch možností pre porovnanie. Výška obrázkov je orezaná, slúžia len ako predstava o dizajne



Obr. č. 7 Ukážka štandardne vyzerajúcej aplikácie vo WinForms.[3]



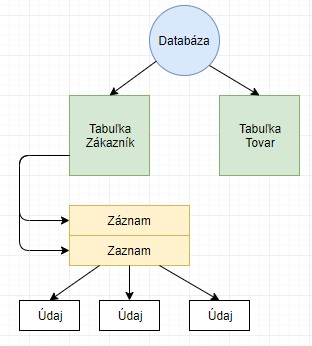
Obr. č. 8 Ukážka štandardne vyzerajúcej aplikácie vo WPF, príklad dostupný voľne na internete

### 3.3 Databáza

Databázový server je softvér alebo hardvér použitý na ukladanie a spätné zobrazovanie už vybraných údajov. Náš server bude verejný.

Naše dáta budú mať svoju štruktúru, môžeme využiť relačné databázy, ktoré zabránia duplicite niektorých dát. Ich mínusom sú v podstate len neštruktúrované dáta, ktoré nemáme. Dáta sú jednoducho uložené a vieme ich získať späť pomocou SQL queries.

Všetky z vybraných databázových systémov sú určené pre prácu s malým až veľkým množstvom dát, preto tento fakt do výberu nezahŕňam. Prikladám svoju jednoduchú predstavu o databáze pred uskutočnením samotného návrhu.



Obr. č. 9 Prvotná predstava o databáze

Výber zo štandardných populárnych riešení, ktoré som vybral na základe webu DB-engines. :

### 3.3.1 MySQL

MySQL je databázový systém sponzorovaný spoločnosťou MySQL AB, ale bola MySQL bolo kúpené taktiež Oracle Corporation. Je to open source, neponúka riešenia pre veľké projekty, nepodporuje XML ale od verzie 5.7 podporuje JSON. Pri práci je možné pracovať s viacerými storage enginmi, štandardne InnoDB alebo MyISAM. Bol vytváraný s myšlienkou podpory webových serverov a webových aplikácií napr. kuriérskych spoločností[4]. InnoDB podporuje foreign klúče a s tým bude táto funkcionalita pre našu aplikáciu dostatočnou.

### 3.3.2 Microsoft SQL server

Je relačný databázový systém vyvíjaný spoločnosťou Microsoft, je výkonný a využívaný ako aj v malých organizáciách, tak aj vo veľkých firmách. Tak ako MySQL podporuje prácu s jazykmi ako Java, PHP, C++, Delphi, ale na rozdiel od neho nepodporuje niektoré novšie jazyky ako Perl či Haskel. Neponúka na výber viacero databázových enginov ako MySQL, čím sa stáva obmedzenejší. Jeho výhodou je jednoduchšie filtrovanie dát pri vyhľadávaní dát z viacerých tabuliek. Naša aplikácia bude mať ale obmedzené množstvo tabuliek, čím sa pre nás stáva táto vlastnosť nepodstatnou. V ponuke je niekoľko edícií, zadarmo je edícia Express, ktorá je obmedzenou verziou oproti štandardnej verzií aj MySQL.

### 3.3.3 PostgreSQL

PostreSQL je plnohodnotný výkonný komunitou obľúbený open source relačný databázový systém vyvíjaný PostreSQL Global Development Group. Plne podporuje spúšťače, procedúry môžu byť programované v jazykoch ako Java, C/C++, Python a ďalšie. Výnimočnou vlastnosťou je dedičnosť. Pri vytváraní novej tabuľky umožňuje preberať dáta z inej už existujúcej tabuľky. Podporuje aj zložitejšie dátové typy. Táto funkcionalita by v našej aplikácií zostala nevyužitá.

# 4 Návrh a implementácia aplikácie

V nasledujúcich odsekov som postupne opísal návrh takej aplikácie, aby spĺňala požadovanú funkcionalitu, a potom samotnú implementácie už navrhnutej aplikácie.

## 4.1 Vyhradenie funkcionality

V tejto časti, pomocou nazbieraných údajov o chode skladu a z analýzy konkurenčných riešení prispôsobíme návrh aplikácie.

### 4.1.1 Akcie a vlastnosti zákazníka

Zákazník musí mať automaticky generovaný identifikačný kľuč ktorý ho bude ďalej reprezentovať. Skladá sa z dvoch častí, posledných dvoch cifier aktuálneho roku a najmenšieho voľného trojciferného čísla väčšieho ako 100. Musí obsahovať údaje prvé meno, druhé meno, telefónne číslo, adresa, a počet tovaru ktorý má. Užívateľ ich musí byť schopný pridávať, upravovať všetky údaje okrem identifikačného kľúča, nemusí ich však vedieť mazať. Z tabuľky zákazníka sa musíme vedieť dostať k zoznamu jeho tovaru. Z okna zákazníka skladník musí vedieť vytlačiť faktúru, ktorá bude obsahovať automaticky doplnené osobné údaje zákazníka a tovaru.

### 4.1.2 Akcie a vlastnosti tovaru

Tovar musí byť označený identifikačným číslom majiteľa, musí mať meno, veľkosť a cenu. Tovar musí vedieť prechádzať medzi jednotlivými stavmi nepredaný, predaný kartou, predaný hotovosťou, vrátený, zaplatený. Tovar môže byť archivovaný, pri čom si ponecháva jeden z pôvodných stavov, ale neukáže sa na zozname tovaru, ani na faktúre zákazníka. Tovar môže mať obrázok, ktorý môže byť odfotený priamo web kamerou, alebo pridaný z počítača na ktorom beží aplikácia. Stavy tovaru musia byť v zozname jednoznačne odlíšené. Skladník musí byť schopný zobraziť archivované údaje, či ďalej ich upravovať.

### 4.1.3 Akcie a vlastnosti záznamov

Záznamy v aplikácií slúžia len ako kontrola roboty skladníka pre šéfa skladu, alebo môžu byť použité pri hľadaní vinníka chyby. Každý záznam musí obsahovať čas zmeny, popis akcie a ak bola zmenená cena, záznam uloží starú aj novú. Z každého záznamu sa musí dať prekliknúť priamo na okno detailu tovaru pre následné úpravy. Medzi jednotlivými záznamami sa musí dať prehľadávať aj podľa dátumu a identifikačného čísla tovaru. Záznamy sa tvoria len pri úpravách tovaru, nie zákazníkov.

## 4.2 Možnosti pridávania fotiek, tlače a identifikácie zákazníkov

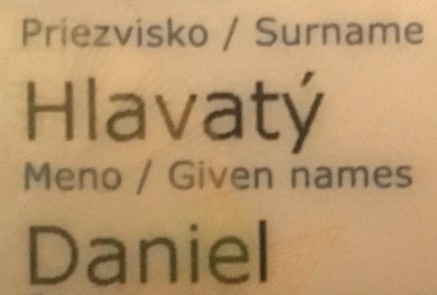
### 4.2.1 Možnosti a problémy identifikácie zákazníkov

Pri každom príchode zákazníka ako prvá prebieha jeho identifikácia. Medzi najčastejšie spôsoby, ktoré môžeme zahliadnuť v okolí, patrí identifikácia občianskym preukazom. Spôsoby implementácie sa líšia.

Najrýchlejším a najelegantnejším spôsobom je využitie kartovej čítačky ktorá číta elektronický čip na občianskom preukaze(eID). Bohužiaľ napriek tomu že eID je dnes na Slovensku štandardom, nie každý občan ho má a preto tento spôsob nie je spoľahlivý.

Miesto eID čipu by sme mohli využiť strojovo čitateľnú zónu (MRZ), čo je zóna na občianskom preukaze, kde sú údaje o držiteľovi a doklade zaznamenané vo forme opticky rozoznateľných znakov[5]. Táto zóna je čitateľná tzv. štrbinovou čítačkou dokladov. Cena takejto čítačky sa pohybuje od 400 € a vyššie. Vzhľadom na to, že je to veľká investícia pre malý sklad, táto možnosť bola zamietnutá.

Ďalšou možnosťou by bolo občiansky preukaz odfotiť, a využiť jednu z knižníc na prekladanie textu z obrázkov. Najpoužívanejšia z nich je knižnica IRONOCR alebo Tesseract od Google. Po otestovaní oboch som však zistil, že sú prekvapivo nespoľahlivé. Stačí poškriabaný alebo pošpinený preukaz, nesprávny uhol pod ktorým je fotené a výsledkom je systém, ktorý si vyžaduje veľkú pozornosť pri fotení občianskeho preukazu človekom obstarávajúcim sklad. OCR je v praxi obyčajne využívaný na zbieranie dát z faktúr, kde štruktúra dokumentu a farba pozadia je jednotná. Ako príklad uvádzam fotografiu, ktorá ľudským okom pôsobí byť nastavená ideálne, napriek tomu ani jedna z knižníc nebola schopná z nej meno správne vyčítať.



Obr. č. 10 Fotografia mena na občianskom preukaze, ako test OCR knižníc

Identifikácia telefónom môže byť tiež problémom. Takmer všetky telefóny dnes sú toho schopné, ale majú dve požiadavky. Prvou je, že zákazník musí mať nainštalovanú danú aplikáciu. Druhou je samotná registrácia, zákazník nemá priradený identifikačný kód pri prvom príchode. Cieľom práce je nájsť najjednoduchší spôsob, považoval som za rozumné nevyužiť ani túto možnosť.

Posledným a najjednoduchším variantom je identifikácia občianskym preukazom bez akýchkoľvek ďalších systémov okrem skladovej aplikácie. Každému novému zákazníkovi bude identifikačný kód priradení, pri vytvorení zákazníka v systéme. Osoba pracujúca s aplikáciou si tak môže opakovaných návštevníkov jednoducho nájsť v aplikácii podľa mena za predpokladu, že mená sú relatívne jedinečné, alebo podľa jedinečného priradeného identifikačného kódu.

Vybral som poslednú možnosť, pretože je používateľsky najjednoduchšia, osoby využívajúce sklad sa nemusia oboznamovať so žiadnymi novými zariadeniami a nevyžaduje žiadnu investíciu majiteľa do čítacích zariadení.

### 4.2.2 Možnosti pridávania fotiek

Skladník pri príchode tovaru alebo dodatočne môže chcieť pridať danú fotografiu.

Pri výbere obrázku z počítača, aplikácia by mala otvoriť štandardné Windows okno pomocou ktorého vyberie cestu k hľadanému obrázku. Aplikácia uloží jeho názov do databázy a uloží ho na FTP server. Obrázky nemôžu byť ukladané lokálne na počítači, pretože musia byť aj vo webovom rozhraní.

Protokol FTP je jeden z najstarších sieťových protokolov(kedysi jediným) využívaných v dnešnej dobe na prenos súborov, v našom prípade obrázkov. K prenosu využíva spoľahlivý TCP protokol. Dobrým porovnaním je porovnanie s protokolom HTTP, ktorý taktiež využíva pripojenie TCP. HTTP využíva len jedno pripojenie pre získanie požiadaviek aj na prenos dát a prispieva tak k lepšiemu výkonu. FTP ponúka dva, jeden dátový a jeden kontrolný.

Rýchlejším spôsobom pre beh skladu by mohla byť web kamera, ktorú by mal skladník vždy pri počítači. Knižnica EMGU CV obsahuje triedy ImageView a Capture, s ktorými je jednoduché spraviť fotografiu.

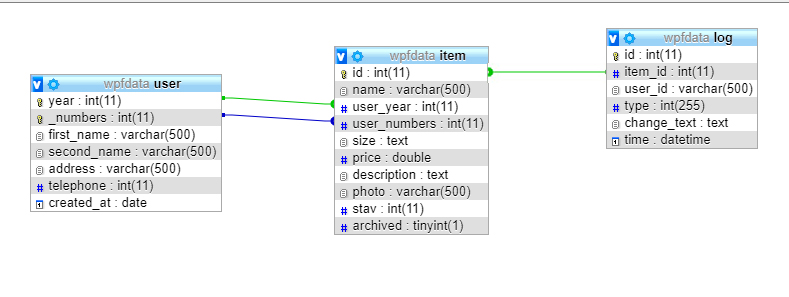
### 4.2.3 Možnosti tlače

Aplikácia má byť schopná vytlačiť faktúru zákazníkovi s jeho aktuálnym tovarom, pre jednoduchosť bez akéhokoľvek skladníkovho zásahu. Vzhľadom na štruktúru dokumentu, ktorá bola ťažko upraviteľná do .doc dokumentu preto upravujem do PDF. \* Využívam knižnicu FreeSpire.PDF ktorá je zadarmo a umožňuje zápis do pdf dokumentov podľa pozície, čo je ideálnym riešením.

Na sklade sa nachádza len jedna tlačiareň, a preto pre jednoduchosť aplikácia neobsahuje menu výberu tlačiarne. Aplikácia vyberie prvú tlačiareň pripravenú na tlač a tlačí.

## 4.3 Databáza

Prepojenie medzi tovarom a zákazníkom pomôže so zaručením priradenia tovaru majiteľovi. Samotné záznamy môžu byť vytvárané pomocou spúšťačov, alebo budú vytvárané samostatne z aplikácie. Keďže webové rozhranie musí byť schopné otvárať obrázky tovarov v prehľade, teda obrázky nie je možné ukladať lokálne na počítači. V databáze bude uložený len názov obrázku, webové rozhranie aj aplikácia si musia byť schopné dané obrázky stiahnuť z počítača FTP. V priloženom obrázku databázy vidíme všetky údaje potrebné k doteraz vyhradenej funkcionalite.



Obr. č. 11 Návrh databázy

## 4.4 Návrh okien

Všetky tabuľky musia mať comboBoxy s vyhľadávaním. Pridávanie zákazníka alebo tovaru sa vykonáva vždy v modálnych oknách, po pridaní okno informuje o úspešnosti akcie a ak prebehla úspešne automaticky ho zatvorí. Všetky ostatné operácie sa vykonávajú v samostatných oknách aplikácie. Pri úpravách detailov už vytvoreného zákazníka alebo tovaru, ak dôjde k pokusu zatvorenia okna s neuloženými zmenami, otvorí sa popup okno, ktoré informuje s touto správou a ponúka možnosť uloženia alebo zahodenia zmenených údajov.

### 4.4.1 Hlavné okno

Hlavné okno bude obsahovať hamburger menu na ľavej strane aplikácie obsahujúce zoznam všetkých prehľadov dostupných v tomto okne, teda prehľad zoznamu všetkých zákazníkov, všetkého tovaru a všetkých záznamov a pomocou neho sa medzi nimi bude dať ľubovoľne pohybovať. Toto menu bude viditeľné len v tomto okne. Pri zapnutí aplikácií bude zobrazený ako prvý zoznam užívateľov.

Zoznam všetkých zákazníkov bude tabuľka s okrúhlym zapínačom slúžiacim na pridávanie nových zákazníkov. Tabuľka bude jednoduchá s možnosťou vyhľadávania podľa identifikačného čísla alebo mena. Pri dvojitom kliku na špecifický záznam sa otvorí okno užívateľa.

Zoznam všetkého tovaru bude tabuľka, ktorá bude mať v každom riadku tri zapínače. Samostatný zapínač pre detail toho daného špecifického tovaru, samostatný zapínač pre detail toho daného špecifického zákazníka, a samostatný zapínač pre popis aktivovanie ktorého odhalí popis tovaru v novom riadku. Aplikácia sa nebude pýtať na zmeny popisu, ale bude ich ukladať pri každej zmene.

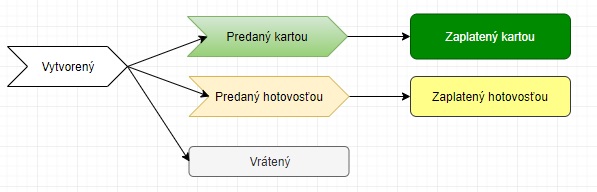
Zoznam všetkých záznamov bude tabuľka s možnosťou vybratia dátumu dňa ako filtrovanie. Tabuľka musí obsahovať stĺpec s textom, ktorý bude informovať o vykonanej zmene v databáze.

Pri vrátení sa k tomuto oknu z iného okna, alebo pri preklikávaní v hamburger menu medzi tabuľkami, tabuľka musí byť vždy aktualizovaná.

### 4.4.2 Okno užívateľa

Okno užívateľa musí obsahovať jednoduché menu pre možnosť zmeny osobných údajov zákazníka.

Musí obsahovať zoznam tovaru s menu zapínača, ktorým užívateľ môže meniť stav tovaru v logickej postupnosti. Môže upraviť popis a otvoriť detail tovaru rovnako ako v hlavnom okne, s doplnenou možnosťou archivovania. Archivovaný tovar sa nezobrazí v tabuľke ak užívateľ nerozhodne inak, a pri tlači sa do PDF dokumentu nezapisuje.



Obr. č. 12 Logické poradie prechodov medzi stavmi

Okno musí obsahovať možnosť pridávania nového tovaru. V tomto okne sa bude realizovať aj tlač, môže poskytovať náhľad na generovaný PDF dokument.

### 4.4.3 Okno detailu tovaru

Jednoduché okno s možnosťami zmeny údajov tovaru, vrátane stavu aj spätne. Obsahuje rovnaké popup okno na kontrolu uložených údajov. Detail tovaru musí podporovať pridávanie existujúcich obrázkov a pridávanie obrázkov za behu pomocou web kamery.

### 4.4.4 Modálne okná na pridávanie

Jediné okná aplikácie s tlačidlom späť, nemajú štandardné okraje známe pre Windows. Pri odchode z okna bez uložených údajov okno popup nevyhodí, po úspešnom pridaní sa okno automaticky nezatvára. Musí tiež vedieť pridávať fotografie oboma spomínanými spôsobmi.

### 4.4.5 Popup pri zmene údajov

Pri neuložených zmenách a pokuse odchodu z okna, aplikácia sa opýta či zmeny uloží. Ak používateľ chce zmeny uložiť a z nejakého dôvodu operácia nepredbehne úspešne, používateľa vráti naspäť do okna užívateľa. Ak údaje neopraví a pokúsi sa okno opustiť znova, aplikácia mu to bez opýtania dovolí. Ak zmeny uložiť nechce, bez oznamu odchádza.

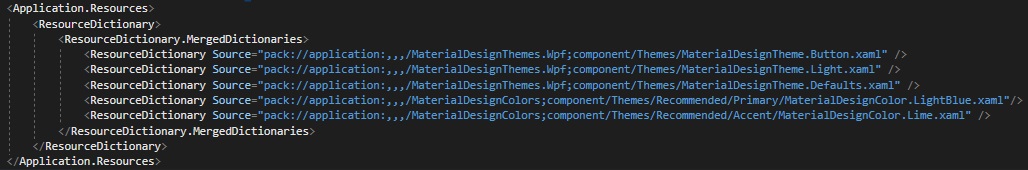
### 4.4.6 Material Design

Material design je dizajnový jazyk vyvíjaný googlom používajúci sa na dizajn stránok či aplikácií. Ušetrí veľa času so štýlovaním aplikácie a aplikácia bude pôsobiť pokročilým dizajnom s minimom námahy. Ako príklady uvádzam kruhový zapínač s popisom, a textBox s takzvaným lietajúcim popisom.

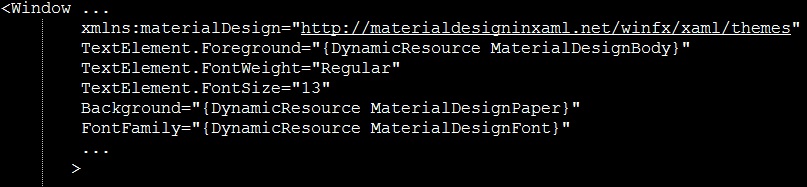
|  |  |
| --- | --- |
| Obr. č. 13 Kruhový zapínač | Obr. č. 14 Kruhový zapínač pri prechode kurzorom myši |

|  |  |
| --- | --- |
| Obr. č. 15 TextBox s “lietajúcim” popisom | Obr. č. 16 TextBox s “lietajúcim” popisom po zakliknutí |

Material design dostaneme do našej aplikácie pridaním linkov jeho komponentov vybranej témy do app.xaml súboru. Potom v každom XAML súbore, v ktorom ho chceme využívať pridáme jeho značku. Prikladám príklad použitia svetlo modrej témy.



Obr. č. 17 Pridanie material designu do súboru app.xaml[6]



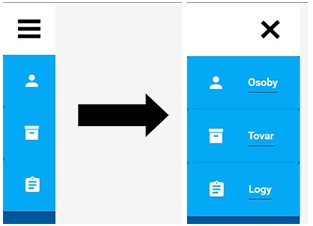
Obr. č. 18 Pridanie material designu a jeho využitie v samostatných oknách

## 4.5 Implementácia a výsledná aplikácia

Táto časť sa zaoberá samotnou implementáciou vybraných riešení, porovnáva návrh s hotovou aplikáciou.

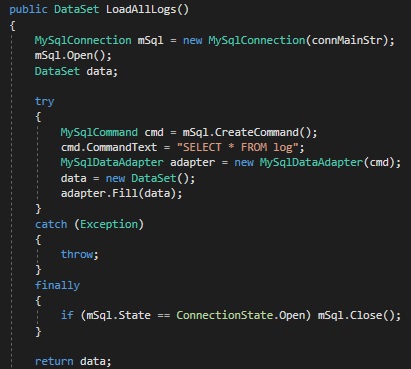
### 4.5.1 Rozloženie ovládacích prvkov

Hamburger menu hlavného okna sa nachádza vždy na ľavej strane okna, ponúka prechod medzi jednotlivými tabuľkami. WPF podporuje aj jednoduché animácie pomocou objektu storyBoard, a tak menu ide otvoriť aj zatvoriť.

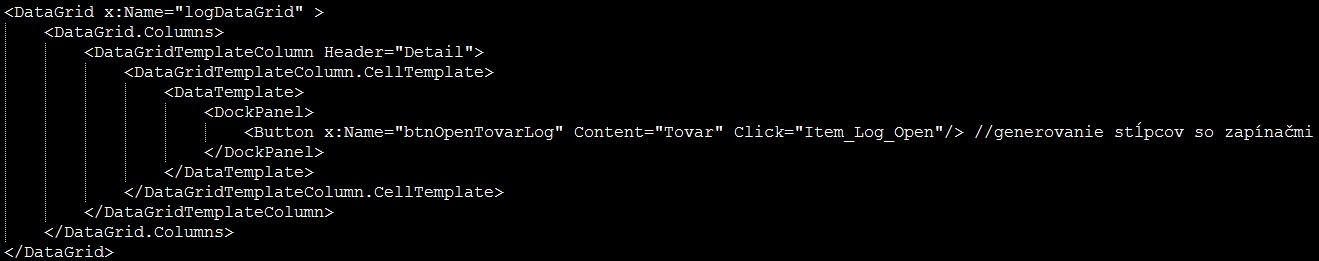


Obr. č. 19 Animácia hamburger menu hlavného okna

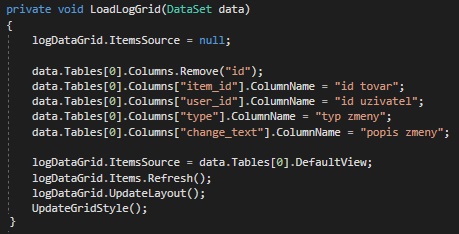
Samostatné tabuľky sú vykresľované v aplikácii pomocou objektu DataGrid, každá má svoje určité vyhľadávacie políčka formou ComboBoxu, alebo DatePickerom. Konvenciou je zobrazovanie ovládacích prvkov na pravej strane tabuľky. Údaje v nich sú vytvárané nasledovným spôsobom: pri inicializácií tabuľky sa aplikácia pripojí na verejnú databázu, odkiaľ si požadované údaje stiahne a vloží ich do objektu(alebo viacerých) DataSet, ktorý zachováva údaje v štruktúre podobnej z MySQL. DataSet potom v kóde programu upravíme na požadovanú štruktúru. DataSety je medzi sebou možné aj spájať, čo umožňuje jednoduchú úpravu dát aj pri tabuľkách skladajúcich sa z dát z viacerých príslušných MySQL tabuliek. Upravený DataSet vieme priamo pripojiť k objektu DataGrid. Úpravy ako farba riadkov, výmena poradia stĺpcov sa uskutočňuje vždy po opätovnej inicializácií gridu, alebo použitím spúšťačov. Ovládacie prvky sú generované automaticky pomocou XAML. Prikladám jednoduchý príklad vytvorenia tabuľky záznamov.



Obr. č. 20 Získanie databázy z verejného servera do objektu DataSet



Obr. č. 21 Vytvorenie tabuľky s generovaným stĺpcom v XAML

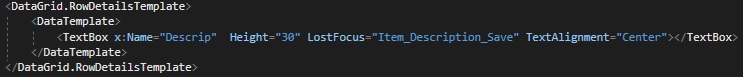


Obr. č. 22 Úprava objektu DataSet a jeho pridanie do DataGridu

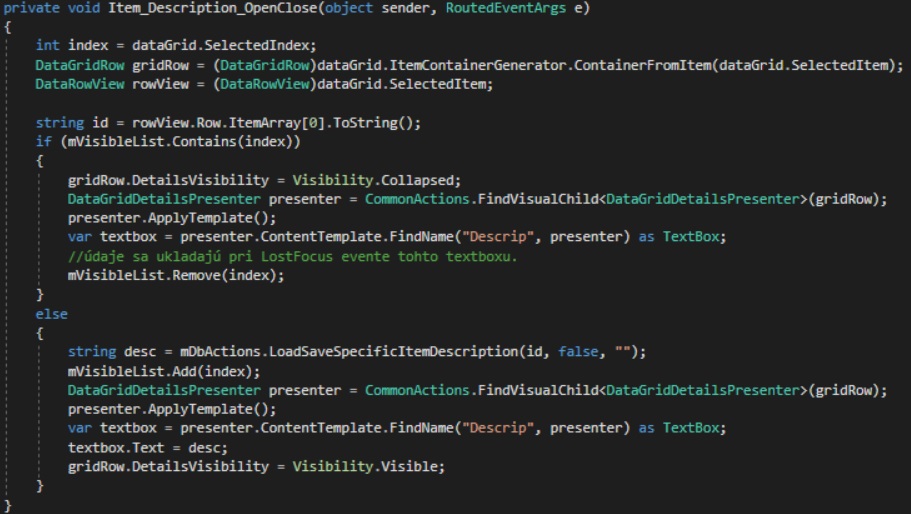


Obr. č. 23 Úprava stĺpca vygenerovaného XAML

Oproti návrhu sa zmenil systém fungovania popisu. V konkurenčných aplikáciách som narazil na problém, kde sa popis tovaru nachádzal v samostatnom stĺpci, pretože popis môže byť obyčajne dlhší, a mal by byť jednoducho opraviteľný skladníkom. Na implementáciu tohto systému som si zvolil XAML objekt RowDetailsTemplate. Po kliknutí užívateľom na príslušný zapínač, aplikácia prehľadá tzv. Visual Tree až kým nenájde objekt s daným názvom príslušný otváranému riadku. V tomto prípade hľadáme TextBox nachádzajúci sa RowDetailsTemplate objekte. Metóda FindVisualChild je metóda opísaná z internetu, ktorá postupne prehľadáva vizuálnych potomkov v danom objekte. Uvádzam príklad:



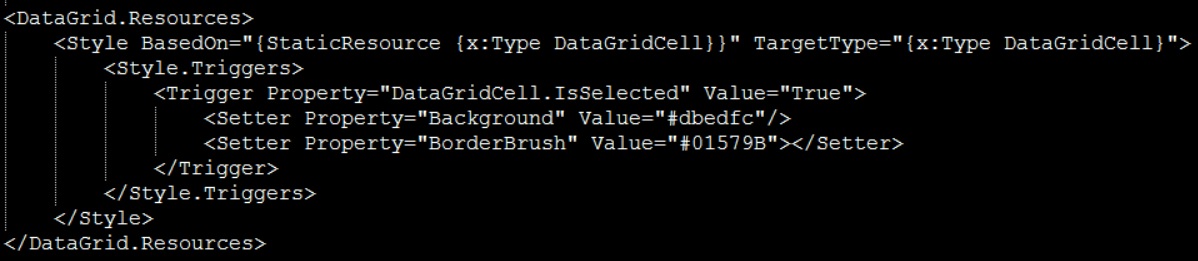
Obr. č. 24 Generovanie otváracieho riadku popisu v tabuľke



Obr. č. 25 Otváranie/Zatváranie riadku popisu, vyhľadávanie pomocou FindVisualChild

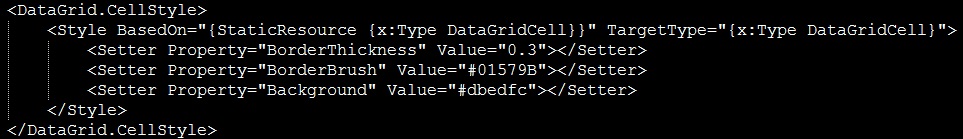
Bohužiaľ, toto riešenie som musel odstrániť a nahradiť štandardným konkurenčným riešením z dôvodu potreby otvárania jednotlivých poznámok, ktoré chce skladník vidieť spolu s ostatnými údajmi tabuľky bez potreby otvárania jednotlivých riadkov.

Pomocou XAML spúšťačov upravujem napríklad farbu označených riadkov (naša aplikácia nemá vedieť označovať riadky, preto farbu označeného som nastavil na rovnakú ako má neoznačený). Uvádzam spomínaný príklad



Obr. č. 26 Príklad využitia XAML spúšťača

K záveru, tabuľky budú roztiahnuté takmer cez celú šírku aplikácie. K úprave štýlu objektov v DataGride, môžeme opäť využiť vlastnosti XAML nasledovným spôsobom. Prikladám príklad štýlovania buniek.



Obr. č. 27 Príklad štýlovania buniek objektu DataGridu pomocou XAML

### 4.5.3 Modálne a popup okná

K implementácii modálnych okien som využil prvok DialogHost z materialDesignu, ktorý ponúka dizajn podobný modálnym oknám, zmámym z webových stránok. Po úspešnom pridaní sa okno zo sekundovým oneskorením samé zatvára využitím asynchrónnej metódy, v opačnom prípade informuje o dôvode neúspechu a zostáva otvorené.

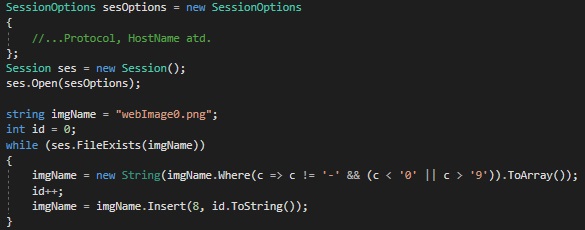
K implementácii popup okien som využil XAML objekt Popup. Na rozdiel od modálneho okna, „nevytmaví“. Riešením je schovanie, respektíve vybielenie okna na ktorom sa zobrazí. Využívam ich pri overovaní užívateľovej vedomosti o neuložených zmenených údajoch. Ak si užívateľ zvolí uložiť údaje ktoré, sú nesprávne zadané, vráti sa do pôvodného okna. Ukážka Popup okna.



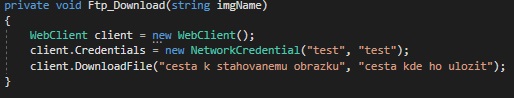
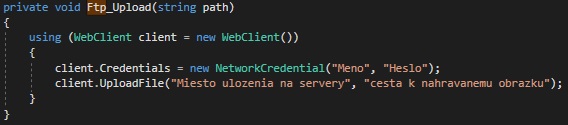
Obr. č. 28 Ukážka Popup okna

### 4.5.5 Obrázky a FTP pripojenie

Pri pridaní obrázku aplikácia vygeneruje názov obrázku s takým číslom, aké sa doposiaľ nenachádza na FTP servery pri čom sa do databázy sa ukladá len názov obrázku. Pri načítavaní aplikácia skontroluje, či hľadaný obrázok už nemá stiahnutý lokálne, ak nie stiahne si ho. Na generovanie názvu sa využíva WinSCP knižnica. Pridávam ukážku kódu nahratie, stiahnutie a generovanie názvov súboru.



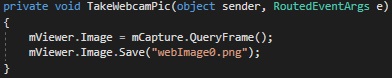
Obr. č. 29 Ukážka generovania názvu obrázku



Obr. č. 30 Ukážka sťahovania a nahrávania obrázkov na server

### 4.5.6 Web kamera

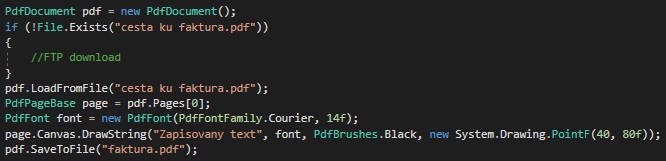
Open source EMGU CV je knižnica ktorá ponúka objekty ImageViewer a VideoCapture, ktorými môžeme robiť fotky pomocou web kamery. ImageViewer je trieda do ktorej obrázok z web kamery načítame, pomocou neho ho potom uložíme fyzicky do počítača. Trieda VideoCapture je schopná urobiť fotku z dostupnej web kamery. Ak je web kamera vypnutá tak ju zapne. Zadarmo verzia knižnice neponúka možnosť web kameru vypínať a zapínať, preto po urobení fotografie kamera zostáva zapnutá, kým sa automaticky nevypne.



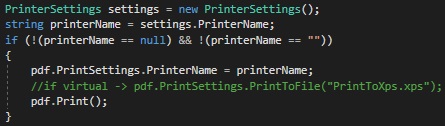
Obr. č. 31 Ukážka zachytenia obrázku dostupného z web kamery pomocou EMGU CV knižnice

### 4.5.7 Tlač

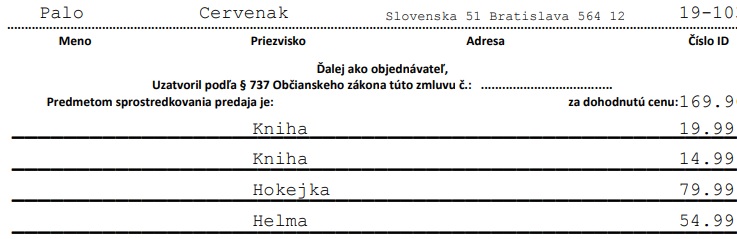
Po stlačení zapínača tlače, aplikácia najprv vytvorí PDF dokument s doplnenými údajmi zákazníka a jeho tovaru. Samotný PDF dokument faktúry sa nachádza na servery, ak ju aplikácia nemá, stiahne si ju rovnakým spôsobom ako obrázky. Využíva knižnicu FreeSpirePDF, ktorá podporuje pridávanie textu podľa pozície. Nepridáva archivované, a ak sa všetok tovar nevojde na jednu stranu, automaticky tlačí dve alebo viac. Aplikácia hľadá prvú voľnú aktívnu aplikáciu a ňou automaticky tlačí. Tlačiareň hľadá pomocou triedy PrinterSettings, ktorá obsahuje všetky tlačiarne ktoré dané zariadenie pozná, ako atribút má meno tlačiarne, a atribút ktorý značí, či je tlačiareň pripravená tlačiť. Užívateľovi oznámi len informáciu o úspechu akcie.



Obr. č. 32 Príklad načítanie a zápis dát do PDF dokumentu pomocou knižnice FreeSpirePDF



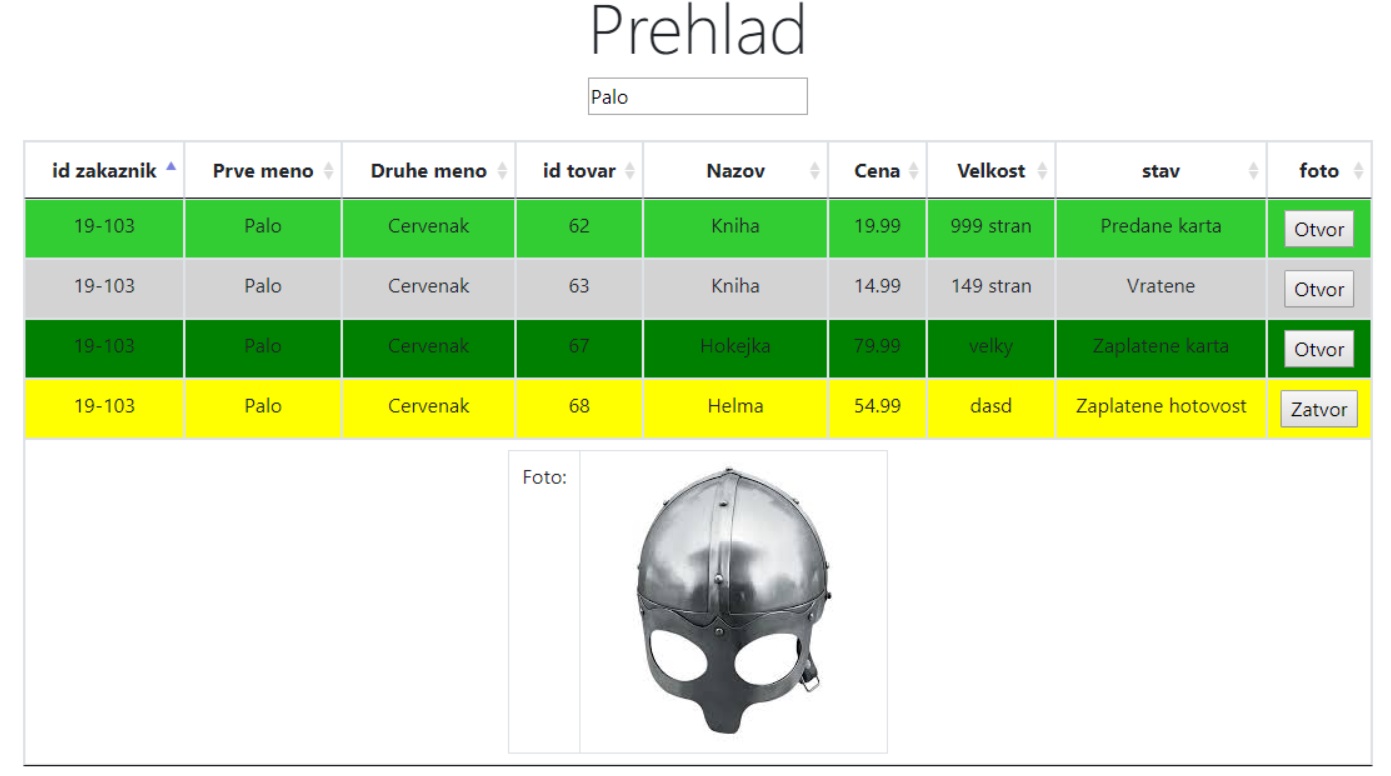
Obr. č. 33 Príklad tlače pdf dokumentu pomocou knižnice FreeSpirePDF



Obr. č. 34 Prikladám ukážku jednej vyplnenej aplikáciou

# 5 Webové rozhranie

Webové rozhranie má spĺňať jednoduchú funkciu prehliadania tovaru dostupného na sklade(nezobrazuje teda archivované), najmä pre zákazníkov. Skladá sa z jednoduchej uvítacej stránky a zo stránky obsahujúcej tabuľku dostupného tovaru, teda neukazuje archivovaný tovar. Nedovoľuje žiadnu úpravu detailov tovaru, a preto neobsahuje prihlasovanie. Vzhľad samotnej tabuľky je takmer identický tabuľkám z WPF aplikácie z rozdielom vyhľadávania ktoré funguje samostatne pre všetky stĺpce, pre jednoduchosť webové rozhranie ponúka len jedno vyhľadávacie políčko ktoré vyhľadáva vo všetkých stĺpcoch tabuľky. Prikladám ukážku.



Obr. č. 35 Ukážka webového rozhrania

## 5.1 Použité technológie – TO DO ASP.net mvc web

Značnou smerodajnou výhodou je že prislúchajúca aplikácia je napísaná taktiež pod .NET frameworku v jazyku C#, teda znamená minimum úprav. Webová aplikácia si sťahuje obrázky rovnakým spôsobom.

### 5.1.1 O Architektúre MVC

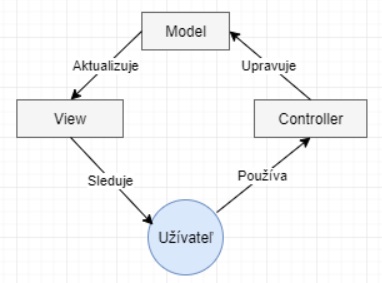
Pôvodne pochádzajúce z anglického Model View Controller, rozdeľuje aplikáciu na tieto tri časti.

- Model obsahuje všetky dáta, ktoré aplikácia pre svoj beh využíva,

- Controller (Kontrolér) je v podstate “pracujúca” časť projektu predstavujúca aplikačnú logiku, príma vstupy používateľa, komunikuje s modelom a ak je potrebné, môže ho upraviť,

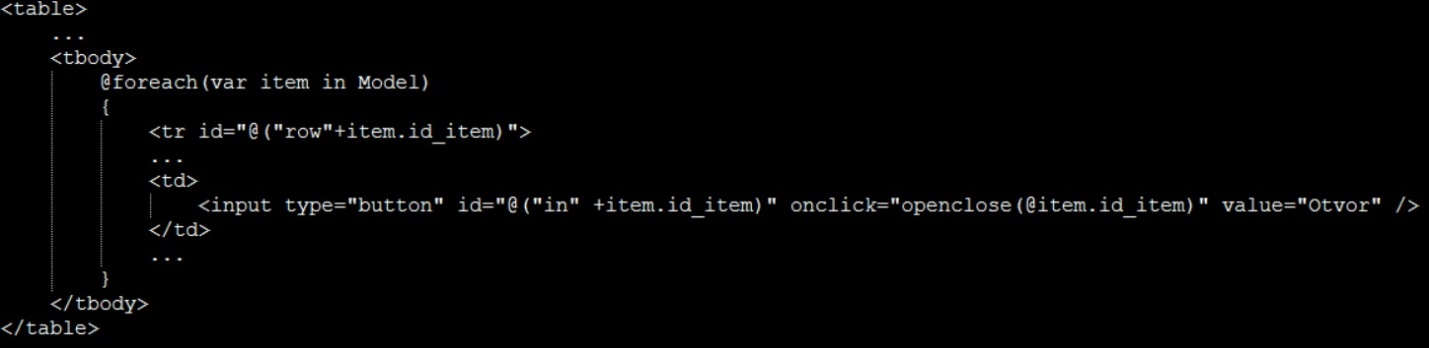
- View(Pohľad) je pasívnou časťou webovej aplikácie, pomocou tejto vrstvy sa používateľovi zobrazujú informácie, obyčajne sa skladajúca z elementov HTML, CSS a javascriptu pomocou ktorého obyčajne volá príslušný kontrolér.

Beh samotnej aplikácie by sa potom dal popísať takto – Aplikácia načíta používateľovi prvý pohľad, ak užívateľ vytvorí nejakú požiadavku pohľad ju zadá priradenému kontroléru, ktorý ju ďalej komunikuje modelu. Do pohľadu sa potom užívateľovi zobrazia požadované zmeny. Prikladám jednoduchú schému.

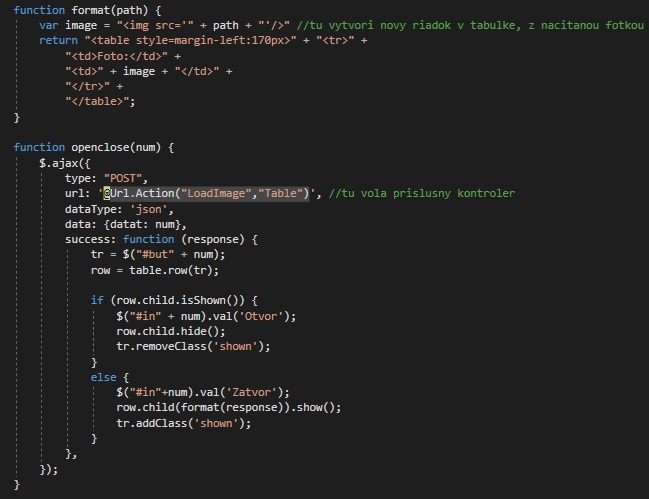


Obr. č. 36 Schéma MVC

Príkladom v našom prípade sú napríklad zapínače otvor/zatvor, ktoré sa nachádzajú v každom riadku tabuľky tovaru. Pri kliknutí používateľa na zapínaš, aplikácia si pomocou Ajaxu vypýta príslušný obrázok, ktorý má byť zobrazený v danom riadku tabuľky od príslušného kontroléra. Kontrolér si ho vypýta od modelu, a pošle mu ho naspať. Ten ho pošle ďalej do prislúchajúcemu pohľadu kde sa vykreslí nový riadok. Prikladám ústrižky potrebného kódu, okrem kontroléra, ktorý vracia relatívnu cestu k stiahnutému obrázku.



Obr. č. 37 Ústrižok html kódu vytvárajúceho tabuľku pohľadu s tlačidlom otvor



Obr. č. 38 Príklad Javascript metódy pridávajúcej nový element do stránky a AJAX metódy komunikujúcej s kontrolérom

### 5.1.2 Rozdiely medzi WPF a ASP.NET

Základným rozdielom je fakt, že WPF sa využíva na programovanie aplikácii z grafickým rozhraním, bežiacim na operačnom systéme Windows. ASP.NET je taktiež vytvorený Microsoftom avšak využíva sa na tvorbu dynamických webových aplikácií, je v podstate súčasťou .NET frameworku.

WPF oddeľuje grafiku od logiky pomocou špeciálneho XAML jazyku, kým ASP.NET je závislí na XHTML vlastnostiach ovládacích prvkov.

Podľa viacerých zdrojov (napr. differencebetween.net) WPF aplikácia načítava dlhšie jednotlivé stránky. Z analýzy konkurenčných systémov sme zistili že to nemusí byť pravidlom.

# 6 Testovanie

Fungovanie každej aplikácie pred svojím vydaním musí byť otestované. Pri testoch sa snažíme nájsť výnimočné stavy aplikácie, a vyhodnocujeme dané riešenie. Funkcionalita ktorá nie je spomenutá v ďalších bodoch bola testovaná zároveň s jej implementáciou.

## 6.1 Virtualizácia tabuliek

Pri vyfarbovaní riadkov tabuliek tovaru, som sa stretol s problémom automatickej virtualizácie tabuliek. Keď bola zapnutá, aplikácia načítavala len riadky ktoré sú práve viditeľné, a pri prechádzaní riadkami ktoré sa nenačítali na začiatku, nemali farbu. Ako prvý test teda vyskúšame vygenerovať 100 položiek tovaru. Beh aplikácie sa očividne spomalil, môj výber úpravy dát v kóde na pozadí sa neukázalo ako ideálne riešenie, ale je aspoň postačujúce.

## 6.2 Tlač a web kamera

Pri implementácií aplikácie som nemal dostupnú žiadnu virtuálnu tlačiareň, tú musíme otestovať. Rozdielom medzi tlačou na virtuálnej tlačiarni a obyčajnej je ukladanie súboru do xps súboru až potom následné tlačenie. Je prekvapivo náročné v kóde zistiť či pripojená tlačiareň je virtuálna alebo nie, a preto podmienku uloženia do xps súboru aplikácia preskočila. Ako opravu, aplikácia ukladá súbor ktorý ide tlačiť do xps súboru vždy.

Web kamera pri implementácií bola testovaná len či je schopná urobiť fotku, či je tento postup príjemný nebolo otestované. Po teste sa ukázalo, že fotiť klikom myšky môže byť náročné, užívateľ musí sledovať aj aplikáciu, aj obraz ktorý fotíte. Ako opravu by som ponúkol možnosť časovača, ktorý môže byť v budúcnosti doplnený.

# 7 Záver

Vzhľadom na to, že táto aplikácia bola mojím prvým stykom s jazykom c#, frameworkom .NET, rovnako ako zo všetkými pridanými knižnica ktoré boli potrebné na splnenie vyhradenej funkcionality. Pri práci v škole som sa naučil pracovať s objektovo orientovanými  jazykmi a tak s C# som sa spoznal pomerne rýchlo.

Mojím silným ponaučením z práce je možnosť DataBindingu v jazyku XAML, ktorú som nevyužil naplno, miesto toho som si často zvolil cestu upravovania dát v kóde na pozadí. Spočiatku to vyzeralo ako jednoduchšie riešenie, avšak pri stálom pridávaní novej funkcionality som sa stretol s častým prerábaním tabuľkových dát, čo ma stálo veľa času. Navyše pri vypnutej virtualizácii v tabuľkách aplikácia nie je taká výkonná ako by mohla byť. Ostatné dizajnové úpravy sa mi vďaka intuitivite a jednoduchosti tohto jazyka implementovali príjemne. Riešenie vyhľadávania v tabuľkách samostatnými comboBoxami by sa v budúcnosti mohli implementovať vstavanými filtrami, ktoré ponúka napr. spoločnost GrapeCity.

Funkcionalitu ktorú sme si vyhradili pomocou analýzy existujúcich konkurenčných riešení alebo analýzou momentálneho fungovania skladu sme splnili. Návrh a výsledná aplikácia sú takmer identické, s výnimkou identifikácie užívateľa pomocou ID. Kameňom úrazu sa stala identifikácia užívateľa, ale po zhodnotení všetkých variantov samotná implemenácia nebola možná.

Myslím si že výzorom moja aplikácia za konkurenčnou nezaostáva, dizajn mojej aplikácie bol silne nimi inšpirovaný. Tým že moja aplikácia je jednoduchšia a bola navrhovaná špeciálne pre už existujúci sklad, samostatné okná obsahujú menej prvkov a tým majú viacej priestoru.

V zoznamoch užívateľa a tovaru vidíme údaje potrebné na dohľadanie ostaných údajov ktoré skladník môže hľadať, čo pri starom systéme bolo časovo náročné.

V budúcnosti by sa dala aplikácia rozšíriť, napríklad pridaním uvítacieho okna výberom peňažnej meny, internetového objednávania, prípadne iných nastavení.

Zoznam použitej literatúry

[1] QT Documentation, 2019. Dostuné na internete v pôvodnom jazyku <<https://doc.qt.io/>>

[2] Microsoft, 2019. Dostupné na internete v pôvodnom jazyku <<https://docs.microsoft.com/en-us/dotnet/>>

[3] Progress, 2019, Obrázok z ukážky aplikácie je dostupný na internete <<https://www.telerik.com/products/winforms/converter-tool.aspx>>

[4] Oracle, 2019, Dostupné na internete v pôvodnom jazyku <<https://dev.mysql.com/doc/>>

[5] Ústredný portál verejnej správy, 2019 Dostupné na internete <<https://www.slovensko.sk/sk/o-portali>>

[x] Zdeňek Troníček, 2011, Učebnice jazyka Java

[6] Google, 2019, Dostupné na internete <<http://materialdesigninxaml.net/>>