Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

Systém řízení skladu

2020

Jiří Zapletal

Vedoucí práce: doc. RNDr. Michal Krupka, Ph.D.

Studijní obor: Aplikovaná informatika, prezenční forma

Bibliografické údaje

Autor:	Jiří Zapletal
Název práce:	Systém řízení skladu
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2020
Studijní obor:	Aplikovaná informatika, prezenční forma
Vedoucí práce:	doc. RNDr. Michal Krupka, Ph.D.
Počet stran:	40
Přílohy:	1 CD/DVD
Jazyk práce:	český

Bibliograhic info

Author:	Jiří Zapletal
Title:	Warehouse management system
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2020
Study field:	Applied Computer Science, full-time form
Supervisor:	doc. RNDr. Michal Krupka, Ph.D.
Page count:	40
Supplements:	1 CD/DVD
Thesis language:	Czech

Anotace

Cílem práce je vytvoření modulu pro správu skladového hospodářství (WMS), který bude vyvinut pro interní program Cutter Information System (CIS) vyvinutý firmou Cutter Systems. Modul se bude skládat ze dvou základních subsystémů (sklad výrobků a sklad materiálu). První subsystém bude obsahovat všechny informace o výrobcích vyrobených firmou Cutter Systems (umístění ve skladech, prodaných, reklamovaných, servisovaných kusech, atd.). Druhý subsystém se zaměří na nakoupený materiál pro výrobu (dodavatelé, prodejní a nákupní ceny, přehled skladových zásob). Oba systémy nabídnou možnost vytvářet nové výrobky a materiál, provádět skladové pohyby, u výrobků navíc i operace. Textová část práce bude obsahovat popis použitých technologií, uživatelskou a programátorskou dokumentaci.

Synopsis

The aim of this thesis is creating a module for warehouse management (WMS), which will be developed for the internal program Cutter Information System (CIS) developed by CUTTER Systems. The module will consist of two basic subsystems (product warehouse and material warehouse). The first subsystem will contain all information about the products manufactured by the company (storage, sold, claimed, or serviced pieces etc.). The second subsystem will focus on the material purchased for production (suppliers, sales and purchase prices, stock overview). Both systems will offer the possibility to create new products and material, perform stock movements, and operations with individual products. The text part of the thesis will contain a description of the technologies used, user and programming documentation.

Klíčová slova: systém řízení skladu; cutter information system; wms; cis

Keywords: warehouse management system; cutter information system; wms; cis

Rád bych poděkoval panu doc. RNDr. Michalu Krupkovi, Ph.D. za tipy, rady a vedení této práce, panu Ing. Martinu Řezáčovi ze společnosti Cutter Systems spol. s r.o. za možnost tvorby práce s tímto tématem a dále pak rodině a přátelům za pomoc a podporu po celou dobu studia.

Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

datum odevzdání práce podpis autora

Obsah

Seznam obrázků

Seznam zdrojových kódů

1 Úvod

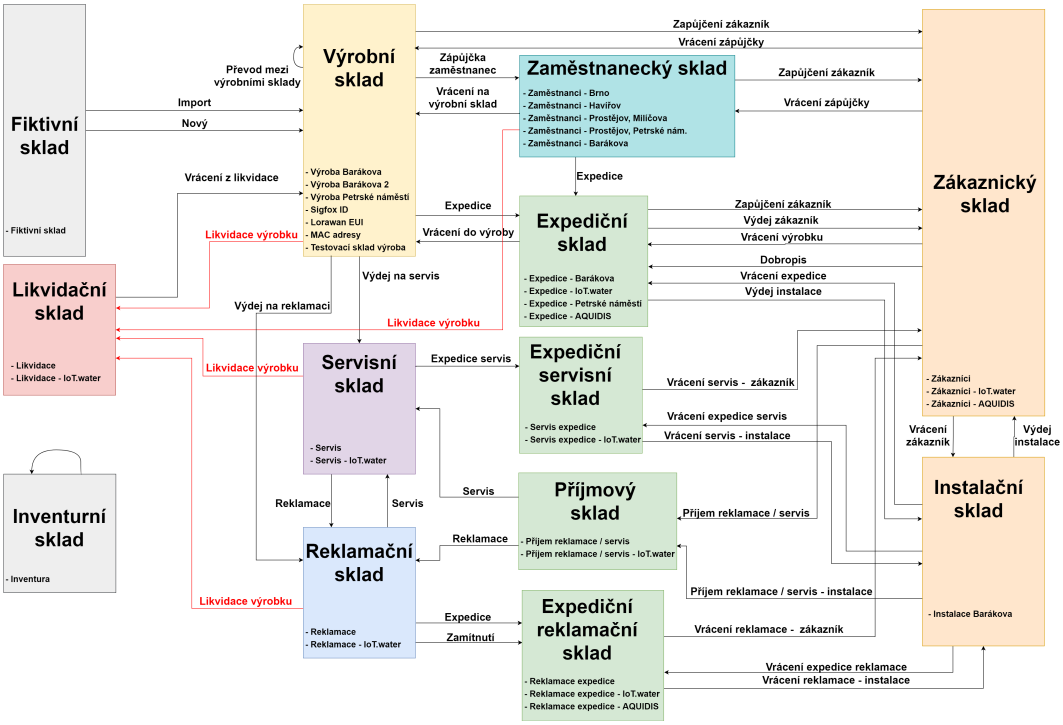
Každá výrobní firma si potřebuje zaznamenávat co nejvíce informací o vyrobených kusech a výrobních postupech. Dříve se tyto informace zapisovaly do papírů a existovaly obrovské místnosti, kde se papíry uchovávaly po několik let. V současné době již existují profesionální firmy, které se na tuto problematiku zaměřují a prodávají své programy. Mezi nejznámější patří Helios nebo Abra. Jelikož se tyto společnosti živí prodejem svých systémů, účtují si za ně obrovské částky v řádech milionů. Proto se firma Cutter Systems spol. s r.o. rozhodla, že si takový skladový systém vytvoří sama, s výhodou naprosté flexibility, co se týče požadavků na software.

V době tohoto rozhodnutí jsem ve firmě pracoval přibližně půl roku a zcela jsem se proto zhostil tohoto nového projektu. Nejdříve bylo potřeba navrhnout a vyvinout modul erp, který by sloužil jako jádro pro ostatní moduly a nabízel by interní přehledy firmy o lidech, jejich pozici v zaměstnání, cizí firmy a jejich vztah k naší firmě.

Po tomto začátku se mohl začít vyvíjet modul správy skladů. Cílem tohoto modulu měla být primárně možnost sledovat počty vyrobených výrobků firmou a jejich umístění. Výrobky se definovaly jako položky, které mají unikátní sériové číslo, název, typ, kategorii, nadřazený výrobek a další podružné informace. Dále bylo potřeba definovat sklady, přes které budou výrobky za svou existenci putovat. Sklady tedy jsou: výrobní, expediční, zákaznický, reklamační, servisní, likvidační a další, jak je vidět na obrázku 1. Jsou zde také definovány povolené pohyby výrobků mezi sklady. Příkladem jsou: vytvoření, expedice, vydání zákazníkovi, příjem, servis, reklamace, likvidace a další.

Modul byl vytvořen a nasazen do ostrého provozu, kde je dnes nedílnou součástí firmy. Během používání systému přicházely požadavky na různé úpravy či přidávání nových funkcí. Dále se firma rozhodla připojit ke stávajícímu skladu výrobků také sklad materiálu. Požadavkem bylo si udržovat také informace o materiálu neboli součástkách, které jsou potřebné pro vyhotovení finálních výrobků. Základem bylo mít několik skladů, na které půjde materiál přijmout a následně pak z něj vydat. Připojily se k tomu informace o ceně, dodavatelích a další. Nyní sklad materiálu umí generovat statistiky odepisování součástek na konkrétní zakázky, které pak slouží jako podklady pro ekonomické oddělení firmy. V současné době se oba skladové subsystémy plně používají, ale požadavky na úpravy také přibývají.

V této práci se čtenář seznámí nejdříve s použitými technologiemi v systému, dále bude následovat uživatelská dokumentace, která se zaměřuje na používání systému a programátorská dokumentace popíše systém jak byl implementován vývojářem. Na konci jsou navíc vyhlídky systému do budoucna.



Obrázek 1: Typy skladů

2 Přehled technologií

Tato kapitola obsahuje přehled všech použitých technologií v této práci. U každé technologie je její stručný popis a použití. Celá práce je psána v programovacím jazyce C# a dotazovacím databázovém jazyce sql. Vývojová prostředí byla použita Microsoft Visual Studio a PgAdmin.

2.1 C Sharp

Vývojáři pro svoji práci používají programovací jazyk ve kterém napíšou program, který je poté spustitelný v nějakém systému. Programovací jazyky se dělí do více úrovní. Nižší úrovně obsahují jazyky, které slouží pro vývoj programů přímo pro procesory, nebo se v nich píšou aplikace u kterých je nutná co nejvyšší rychlost. Vyšší programovací jazyky se specializují na tvorbu složitějších komplexních systému, přičemž jsou pro programátora pohodlnější. Například pokud bychom chtěli poslat e-mail z námi vytvořeného programu, tak ve vyšším programovacím jazyce bychom tuto funkci napsali v řádu minut, v nižším programovacím jazyce by to trvalo minimálně hodiny.

Pro tvorbu tohoto systému byl zvolen jazyk C#, který v dnešní době společně s jazykem java patří mezi nejpoužívanější vyšší programovací jazyky. Jazyk java měl oproti jazyku C# obrovskou výhodu v tom, že se programy v něm napsané daly spustit na jakémkoliv operačním systému, přičemž programy v jazyce C# šly spustit pouze na operačním systému Windows. Společnost Microsoft se ale velice posunula vpřed, když vydala novou verzi, která je jíž multiplatformní, tudíž aplikace jsou spustitelné na jakýchkoliv operačních systémech. Tento krok posunul jazyk C# na první místo, jelikož nemá oproti jazyku java žádnou nevýhodu, ale dost výhod, například v psaní desktopových aplikací, protože se v něm pohodlně vytváří grafické uživatelské rozhraní.

2.1.1 .NET Framework

Pro možnost spuštění a fungování programu je potřeba aby existovalo dané běhové prostředí, kde se program spustí. .Net Framework je základní běhové prostředí pro programy vyvinuté v jazyce C#. Je to také první běhové prostředí ze všech technologií .NET. Toto prostředí ještě není multiplatformní. Běžně se používá pro vývoj grafických částí systémů, které jsou pak běžně užívány na operačních systémech Windows.

2.1.2 .NET Core

Novější open-source běhové prostředí pro aplikace psané na platformě .NET. Je multiplatformní, tudíž je spustitelný na libovolném počítači s libovolným operačním systémem. Převážně se používá pro aplikace, kde není potřeba grafické uživatelské rozhraní. Takové programy běží na pozadí a běžný uživatel o nich ani

neví, nebo jsou umístěny na výkonných centrálních počítačích, které se nazývají servery.

2.1.3 .NET Standard

Specifikace .NET, která je implementována ve všech rozhraních platformy .NET, tudíž knihovny závislé na .NET Standardu je možné použít v projektech závislých jak na .NET Frameworku tak na .NET Coru a i dalších.

2.2 SQLite

Každý větší systém je potřeba konfigurovat. Konfigurační data je potřeba nějak ukládat. Pro tento systém je zvoleno ukládání konfigurace do relační databáze SQLite, která uchovává své data v jednom souboru. Výhodou ukládání do takové databáze je nemožnost běžného čtení dat, je potřeba k tomu mít program na prohlížení a úpravu.

2.3 PostgreSQL

Pro ukládání všech dat v systému (kromě konfiguračních) je zvolena open-source relační databáze. Tato databáze je velice používaná a připravena pro ukládání velkých dat.

2.4 Entity Framework

Data z databáze se dají prohlížet přes vhodnou aplikaci. Pro spojení programu s databází je potřeba mít v programu databázový ovladač. V jazyce C# se pro propojení s PostgreSQL používá ovladač Npgsql. Abychom si nemuseli všechny dotazy do databáze psát ručně v jazyku sql, používá se navíc objektový databázový mapper s názvem Entity framework. Díky tomuto mapperu můžeme psát dotazy v jazyku C# a i přesto nad nimi mít stoprocentní kontrolu. Dále podporuje dotazy LINQ, které umožňují vytvářet složité databázové dotazy.

2.5 Cutter information system

Cutter information system se zkratkou CIS je systém vyvinutý firmou Cutter Systems spol. s r.o. V softwarovém odvětví této firmy je to hlavní produkt. Systém je plně pluginovatelný, tudíž na jeho základě lze vytvořit jakoukoliv aplikaci řešící jakýkoliv druh problému. Je postavený na architektuře server-klient, tudíž v produkci běží serverová část a klientských částí může být několik, které se připojují k serveru přes firmou vytvořený protokol ALCP, který využívá internetové přípojení s protokolem TCP/IP.

Jádro systému poskytuje několik funkcí, které již nemusí vývojář implementovat na úrovni modulů. První nejdůležitější funkcionalitou je správa uživatelů.

Každý uživatel má své uživatelské jméno a heslo pro možnost připojení do systému. Uživatelů může být neomezené množství a pro lepší orientaci se dají dělit do skupin. Pro možnosti omezení přístupu k částem systému existují uživatelské role. Každý uživatel může mít přidělených více rolí. Každá role má pak definována práva na jednotlivé funkce nebo části systému.

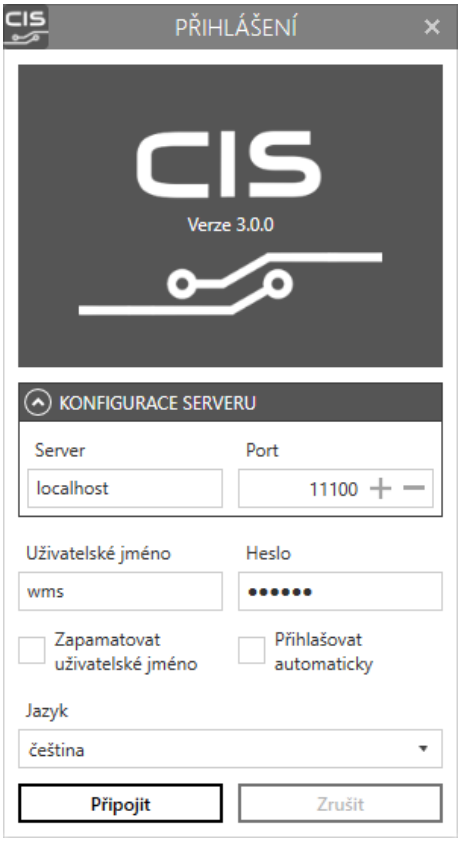
Dále nám jádro poskytuje připojení do konfiguračního souboru pro možnost vyčítání a ukládání konfigurací, základní komunikaci s databází přes entity framework a možnost registrovat rest api dotazy, jelikož souběžně se serverem může běžet součást, která poskytuje serverové rest api, sloužící pro komunikaci přes http. V neposlední řadě nám nabízí komunikaci s hardwarovými zařízeními vyrobených firmou přes firemní Cutter protokol.

3 Uživatelská dokumentace

Tato kapitola popisuje jak se systém ovládá. Důležité je, že systém tvoří dvě aplikace. Serverová část běží jako konzolová aplikace nebo systémová služba, která poskytuje data klientům a dále vše loguje, tudíž není dále pro běžného uživatele důležitá. Klientská část je spustitelná desktopová grafická aplikace pro Windows, která bude v této kapitole popsána.

3.1 Spuštění a přihlášení

Po spuštění klientské aplikace se zobrazí přihlašovací okno. Aplikace se musí připojit k serveru, který nemusí nutně běžet na stejném počítači, tudíž je nutné zadat adresu serveru, port na kterém poslouchá a svoje přihlašovací údaje. Výchozí uživatelské jméno je *wms* s heslem *wmswms*.

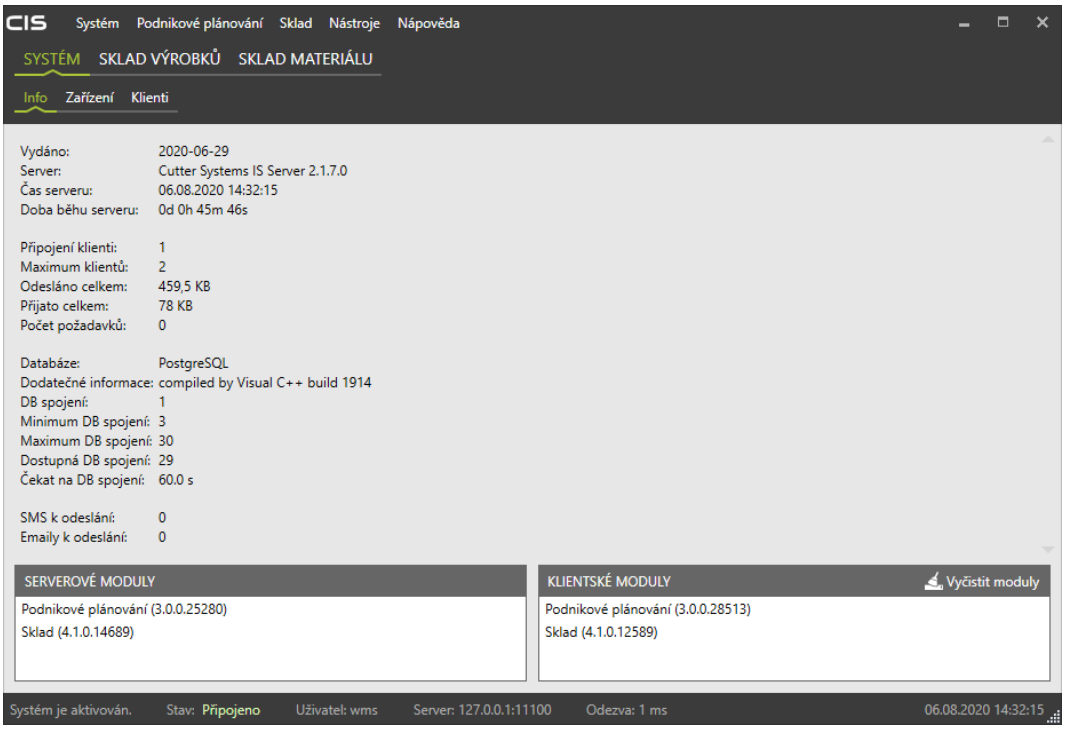


Obrázek 2: Přihlášení do aplikace

Po stisknutí tlačítka Připojit se nejdříve ověří správnost údajů pro přihlášení a dále se ověřují aktualizace klientské části. Pokud se zjistí, že existuje novější verze aplikace, automaticky se aktualizace stáhnou a aplikace se spustí.

3.2 Úvodní pohled

Po přihlášení do systému se zobrazí hlavní okno aplikace. Zde je možné spouštět a vykonávat veškerou funkcionalitu, kterou systém nabízí.



Obrázek 3: Úvodní pohled

3.2.1 Menu

3.2.1.1 Systém

Tato položka je z jádra systému. Obsahuje základní informace o aktuálně přihlášeném uživateli, správu uživatelů, rolí a jednotlivá práva. Dále je zde pokročilé nastavování serveru a zobrazení serverového logu. V poslední řadě jsou zde položky pro odhlášení klienta a vypnutí aplikace.

3.2.1.2 Podnikové plánování

Tato položka je přidána prvním modulem, který je potřebný pro další moduly. Jsou zde číselníky firem, jejich oddělení a pracoviště. Dále jsou zde evidovány osoby a pracovní pozice osob v jednotlivých firmách. Poslední položkou zde jsou Práva na firmy. Zde je možné omezit jednotlivým uživatelům viditelnost firem.

3.2.1.3 Sklad

Nejdůležitější položka v menu. Obsahuje všechnu funkcionalitu, která je součástí této práce. Popis této položky je v kapitole **3.3.**

3.2.1.4 Nástroje

Zde jsou dvě pokročilé funkce, které jsou hojně využívány v jiných aplikacích postavených na tomto systému. Jedná se o programy komunikující s firemním hardwarem.

3.2.1.5 Nápověda

Nápověda obsahuje všechny podrobnější informace o systému, licencích, zabezpečení a novinkách v jednotlivých modulech, jelikož moduly se pořád vyvíjí a aplikují se do nich nové funkcionality.

3.2.2 Záložky

3.2.2.1 Systém

První hlavní záložkou je Systém, kterou poskytuje jádro systému. Obsahuje tři podzáložky.

1. Info - přehled základních informací o serveru, datum vydání, doba funkčnosti, počet připojených klientů, přenesených dat a informace o databázi. Ve spodní části je zobrazen přehled modulů,
2. Zařízení - výpis připojených hardwarových zařízení vyrobených firmou (v této aplikaci žádné nevyužíváme),
3. Klienti - tabulka všech připojených klientů k serveru, pokud je aktuálně přihlášený uživatel administrátor, může klienty odpojit nebo jim i vypnout počítač.

3.2.2.2 Sklad výrobků a sklad materiálu

Přehledy základních informací o systému, který je předmětem této práce. Popis těchto záložek je v kapitole **3.3.**

3.3 Modul WMS

3.3.1 Číselníky

Přehled základních dat pro správnou funkcionalitu celé aplikace. Tabulky jsou k dispozici z menu Sklad a podmenu Číselníky. Úprava těchto dat je možná jen s oprávněním a měl by ji provádět administrátor nebo správce systému, jelikož po změnách může dojít k jinému chování, které nemusí být žádoucí.

3.3.1.1 Typy skladů

Tabulka pro přehled všech typů skladů. Každý typ může obsahovat více skladů. Například existují tři výrobní sklady.

3.3.1.2 Kategorie zboží

Pro lepší přehled a třídění všech výrobků, materiálů a jejich typů jsou k dispozici Kategorie. Každý výrobek je nějakého daného typu a každý typ je nějaké konkrétní kategorie. Vytvoření kategorií bylo původně zamýšleno tak, že každá konkrétní kategorie bude reprezentovat skupinu zařízení se stejnou funkcionalitou lišících se pouze v některých částech či verzích. Uživatelé často systém používají jinak než bylo plánováno vývojáři a tudíž nyní existuje kategorie s názvem Výrobky CUTTER, která obsahuje mnoho typů, které nemají nic společného.

3.3.1.3 Typy zboží

Nejdůležitější rozdělení všech výrobků a materiálů je podle daných typů. Na produkční databázi existuje již přes tisíc šest set typů.

3.3.1.4 Typy operací

U všech výrobků je potřeba evidovat různé úkony, které se s nimi provedly. Prvním z těchto úkonů jsou Operace. Operace se vždy provádí s konkrétními výrobky, přičemž jich může být v dané operaci i více. Příkladem může být: osazení, oživení, test, finální test, oprava a další.

3.3.1.5 Typy skladových pohybů

Výrobky i materiál je potřeba evidovat. U materiálu používáme pouze příjem, výdej a inventuru. U výrobků je to ale složitější. Zde se musí pozorovat každý krok provedený s tímto kusem. Začíná to skladovým pohybem s názvem Vytvoření nového výrobku a dále to pokračuje expedicí a vydání zákazníkovi. Dále je možné, že zákazník pošle kusy na reklamaci nebo servis a výrobek je tedy nutné přijmout, zlikvidovat nebo opravit případně dále znovu expedovat a vydat zákazníkovi. Těchto procesů může být i více, ale každý pohyb musí být v systému evidovaný a také dohledatelný.

3.3.1.6 Parametry

Při vývoji systému není vždy dopředu jasné, co za konkrétní data se bude ukládat, proto vznikly Parametry. Parametry je možné přidávat ke konkrétním datům, které jsou: výrobky, materiál, operace a skladové pohyby. Systém umožní přidat různé parametry těm datům, které jsou daného typu, pro který jsou definovány.

3.3.2 Zakázky

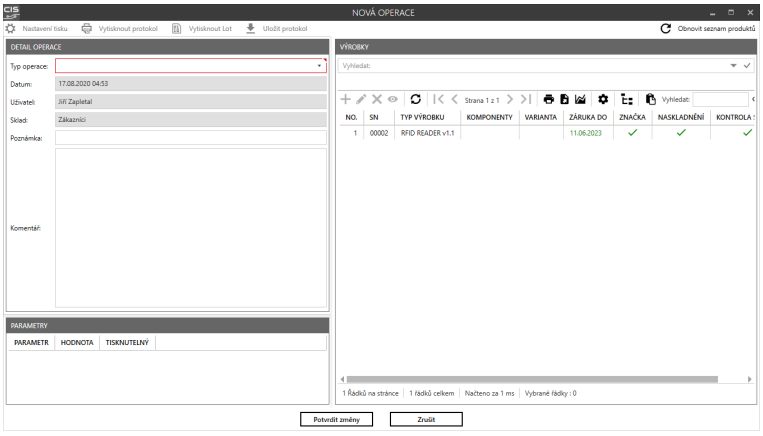
Systém bylo třeba provázat i s účetním programem, který firma používá. V daném programu jsou základní data ukládána ve formě zakázek. Tyto zakázky si tento systém stahuje přes rest api, který program nabízí. Při výdeji materiálu je tedy vždy potřeba zadat konkrétní zakázku na kterou se materiál odepíše. Data jsou pak dále exportována pro ekonomické oddělení firmy, které je dále zpracovává a má tudíž přehlednější provázání obou systém.

3.3.3 Sklady

Nejpodstatnější část v systému a návrhu správného fungování správy skladů jsou sklady. Jak již víme, každý sklad je nějakého daného typu, aby šlo s výrobky a materiálem provádět příslušné skladové pohyby. Pro správné fungování je potřeba mít minimálně jeden sklad od každého typu. Jelikož je systém v produkčním prostředí velice rozsáhlý a poskytuje pohledy i zákazníkům, existuje tedy až šest skladů jednoho daného typu. Sklady můžeme rozdělit na dva základní druhy: sklady výrobků a sklady materiálů. Toto rozdělení je přísně striktní a není možné, aby se materiál objevil na skladě výrobků nebo naopak.

3.3.4 Skladové operace

Skladové operace jsou jedním ze dvou základních úkonů, které lze s výrobky provádět. Operace lze přidávat nebo upravovat. Každá operace musí obsahovat alespoň jeden výrobek a musí být nějakého typu. Nakonec lze přidat poznámku nebo komentář.



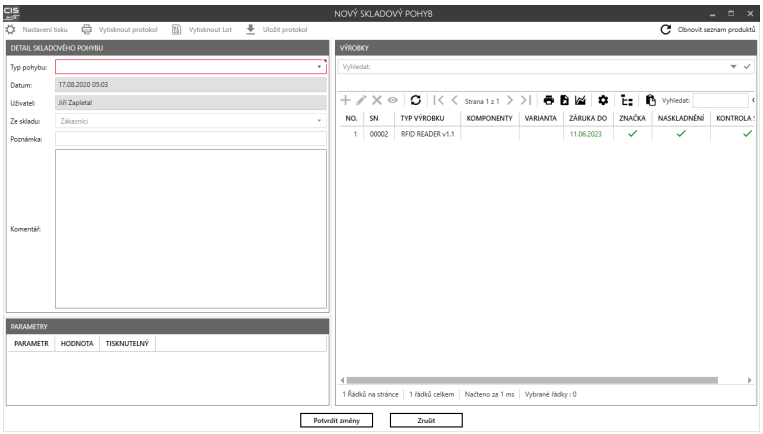
Obrázek 4: Nová operace

Existují dvě speciální operace, které do okna přidají další pole pro vkládání dat. Jsou to operace s názvy Nastavení záruky a Prodloužení záruky. Jejich název přesně popisuje, jaký mají účel. Operaci Nastavení záruky lze kdykoliv ručně vytvořit, ale také se vytváří automaticky při výdeji výrobků zákazníkovi. S

druhou operací je to podobně, ta se sama vytváří při opětovném vydání výrobku zákazníkovi a doba se prodlouží o čas, který výrobek strávil mimo sklad zákazníci.

3.3.5 Skladové pohyby

Druhým hlavním úkonem je provádění Skladových pohybů. Jejich provedením se změní aktuální umístění výrobků nebo materiálů. Tato funkce je nejpoužívanější v systému. Každý pohyb musí být daného typu a musí být proveden z jednoho skladu do druhého skladu. Podle výběru typu se zobrazí seznam možných skladů, na které lze převádět. Dále se zobrazují vstupní políčka, která je potřeba vyplnit.



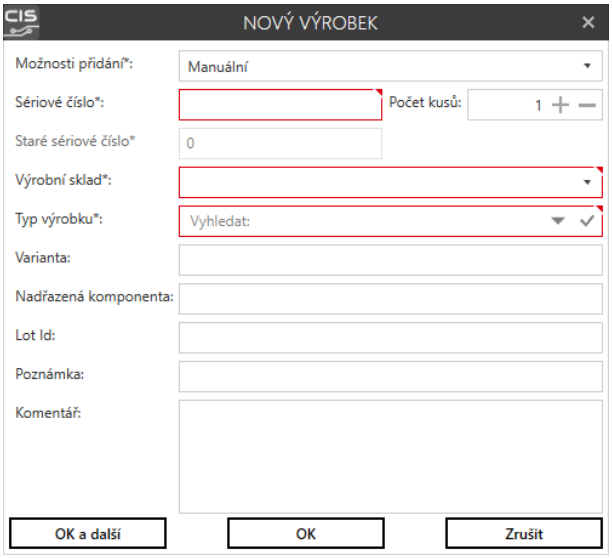
Obrázek 5: Nový skladový pohyb

3.3.6 Výrobky

Výrobky jsou jednou ze dvou komodit, se kterými se v systému pracuje. Každý výrobek má své unikátní sériové číslo, podle kterého lze výrobek jednoznačně najít. Dále výrobek obsahuje mnoho dalších potřebných informací. Musí být nějakého konkrétního typu, aby se dal správně zařadit. Důležitou podmínkou je, že každý výrobek se musí vždy nacházet na nějakém jednom skladu. Výrobky se také dají takzvaně řetězit neboli vytvořit z nich stromovou strukturu tak, že jde výrobku nastavit rodič. Pokud má výrobek nastaveného rodiče, říká se mu potom komponenta. Nelze ale vytvořit cyklickou závislost.

3.3.6.1 Přidání výrobku

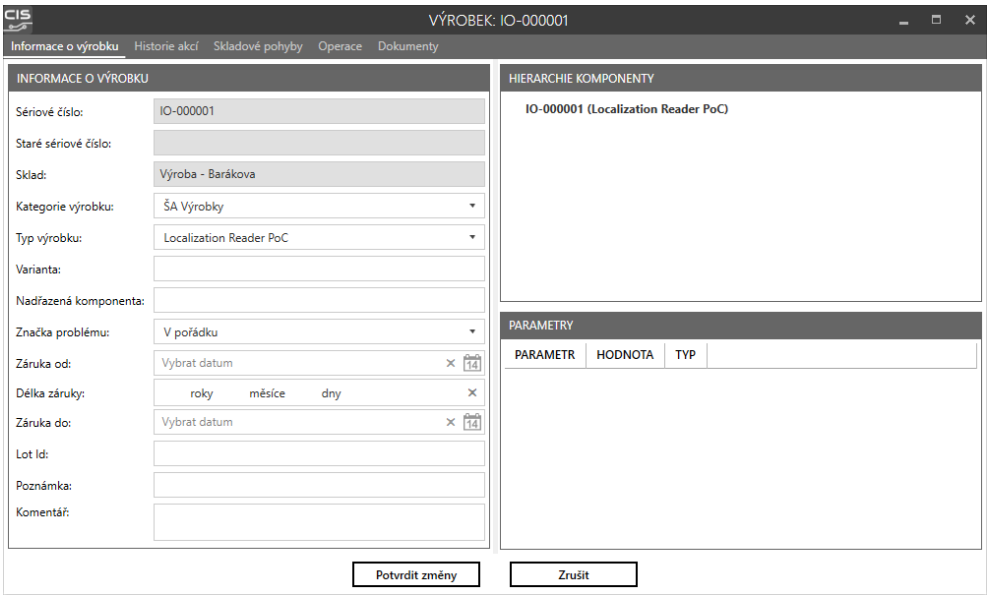
První důležitou funkcí systému je přidávání výrobků. Při přidávání se musí zvolit sériové číslo, případně řada sériových čísel pro vytvoření více výrobků zároveň. Musí se také zvolit příslušný typ výrobku a sklad, na který bude výrobek po vytvoření přesunut. Dále je možné přiřadit k výrobku další informace, jako jsou lot-id, poznámka nebo komentář.



Obrázek 6: Vytvoření výrobku

3.3.6.2 Úprava výrobku

Během života výrobku je někdy potřeba jej upravovat. Nelze upravit jeho sériové číslo, co však jde je kategorie a typ, nadřazený výrobek, poznámka, komentář a také, jestli typ nemá obecnou chybu. Nově je v systému volba nastavení záruky.



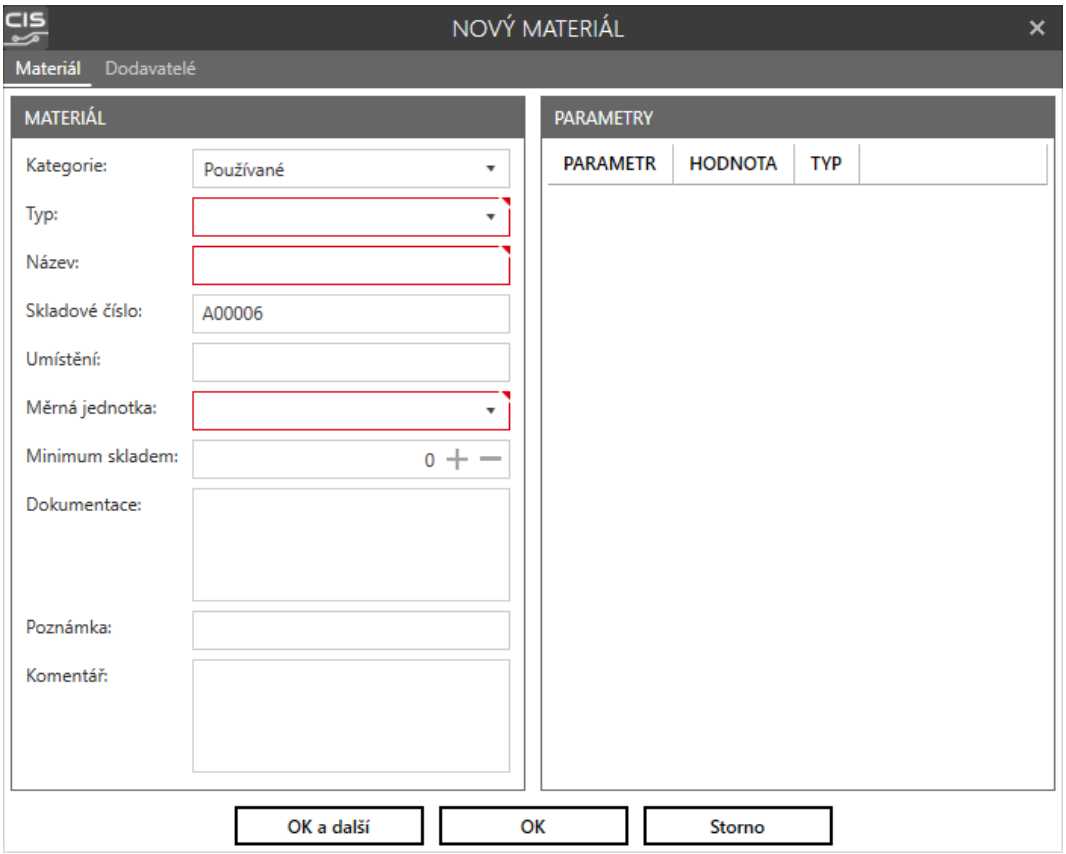
Obrázek 7: Úprava výrobku

3.3.7 Materiál

Práce s materiálem se zásadně liší od práce s výrobky. Kategorie i typy se vytváří ve stejných tabulkách, ale konkrétní materiál již nemá jednoznačné sériové číslo, ale označujeme jen druhy materiálu a jejich počty. Například rezistor 100 ohmů není označen každý zvlášť, ale třeba dvacet kusů.

3.3.7.1 Přidání materiálu

Při přidávání nového materiálu je potřeba vyplnit typ a název. Automaticky se generuje skladové číslo, které se poté tiskne a lepí na sáčky či krabičky, aby šel materiál fyzicky vyhledat, případně je k dispozici pole umístění kde se zapisuje například číslo police. Další důležitá věc jsou dodavatelé, aby se při nedostatku materiálu mohlo rychle materiál doobjednat od stejných dodavatelů, o kterých víme, že jej prodávají.

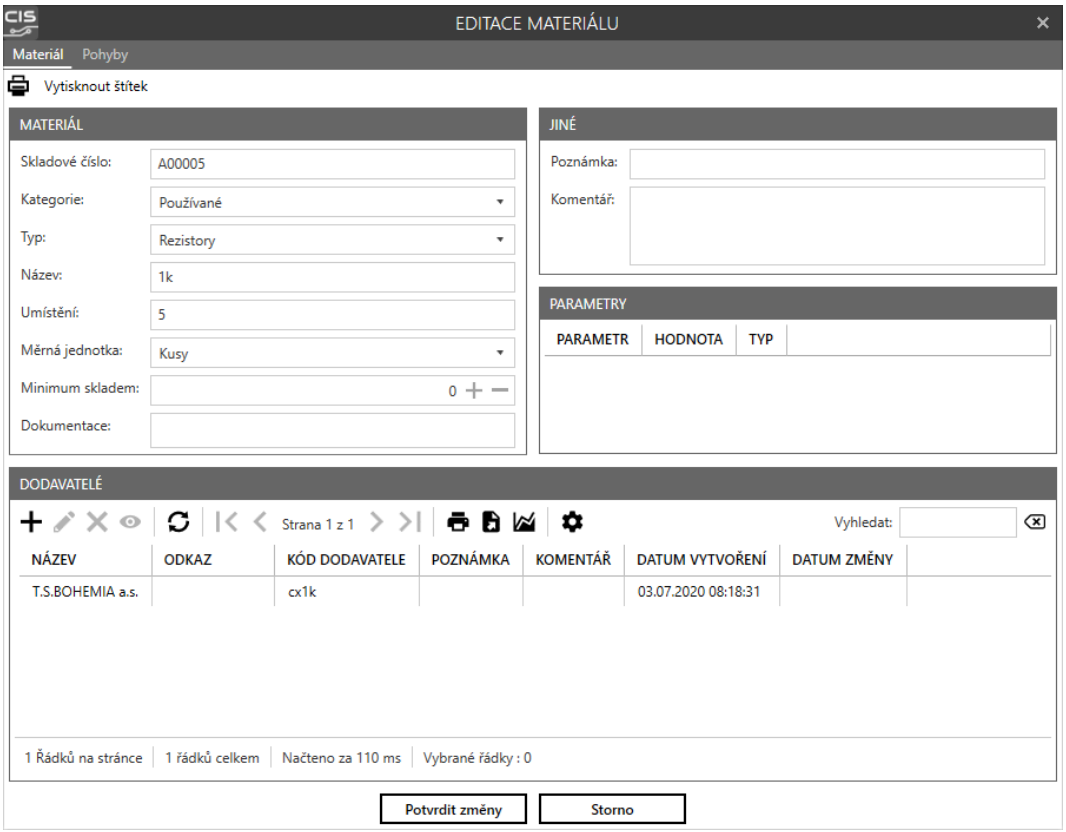


Obrázek 8: Vytvoření materiálu

3.3.7.2 Úprava materiálu

Editace materiálu se nepoužívá tak často, ale tato funkcionalita existuje. Pokud uživatel zavítá do tohoto okna, potřebuje si pravděpodobně změnit umístění,

nebo přidat poznámku nebo komentář.



Obrázek 9: Úprava materiálu

3.3.8 Výchozí pohledy - Výrobky

3.3.8.1 Stavy skladů

Zobrazen celkový počet a ceny na jednotlivých skladech.

3.3.8.2 Stavy výrobků na skladech

Zobrazen celkový počet a ceny na jednotlivých skladech, přičemž jsou výrobky ještě rozděleny podle kategorií a typů

3.3.8.3 Historie akcí

Přehled všech provedených skladových operací a skladových pohybů nad všemi výrobky nad všemi sklady za celou dobu používání systému. Na produkci tabulka obsahuje přes čtyři sta čtyřicet tisíc záznamů.

3.3.8.4 Všechny výrobky

Výpis všech výrobků v systému. Toto je nejpoužívanější pohled, na který se aplikují různé filtry pro vyhledávání požadovaných výrobků.

3.3.9 Výchozí pohledy - Materiál

3.3.9.1 Pohyby

Přehled všech provedených skladových pohybů s materiálem.

3.3.9.2 Přehled materiálu

Tabulka, která zobrazuje všechen materiál v systému.

3.3.9.3 Dostupný materiál

Zde je zobrazen materiál s daným skladem pokud byl někdy proveden pohyb s materiálem nad daným skladem. Pokud tedy nebude v systému žádný pohyb s materiálem, bude tato tabulka prázdná.

3.3.10 Správa skladu

Tato položka je v menu s názvem Správa skladu, která nabízí další podpoložky.

3.3.10.1 Odstranit výrobek

S tímto programem pracují i obyčejní lidé, kteří čas od času udělají chybu. Touto chybou může být vytvoření nového výrobku nebo jeho vytvoření se špatným sériovým číslem. Aby se v této situaci nemuseli zaměstnanci obracet na mě jako vývojáře a správce systému, abych musel provádět ruční zásah nad surovými daty v databázi, vznikla tato funkčnost. Pokud byl výrobek vytvořen a nebyla u něj provedena žádná operace ani žádný skladový pohyb je možné ho touto funkcí smazat.

3.3.10.2 Sloučit transakce

Ze skladových pohybů typu výdej zákazníkovi se dále navíc tiskne i protokol, který slouží jako výdejka a přikládá se zákazníkovi k přehledu a kontrole. Pokud si zákazník požaduje odebrat naráz více výrobků, které nebyly v systému vydány zároveň, je potřeba před tiskem provést sloučení těchto výdejek a k tomu slouží tato funkčnost.

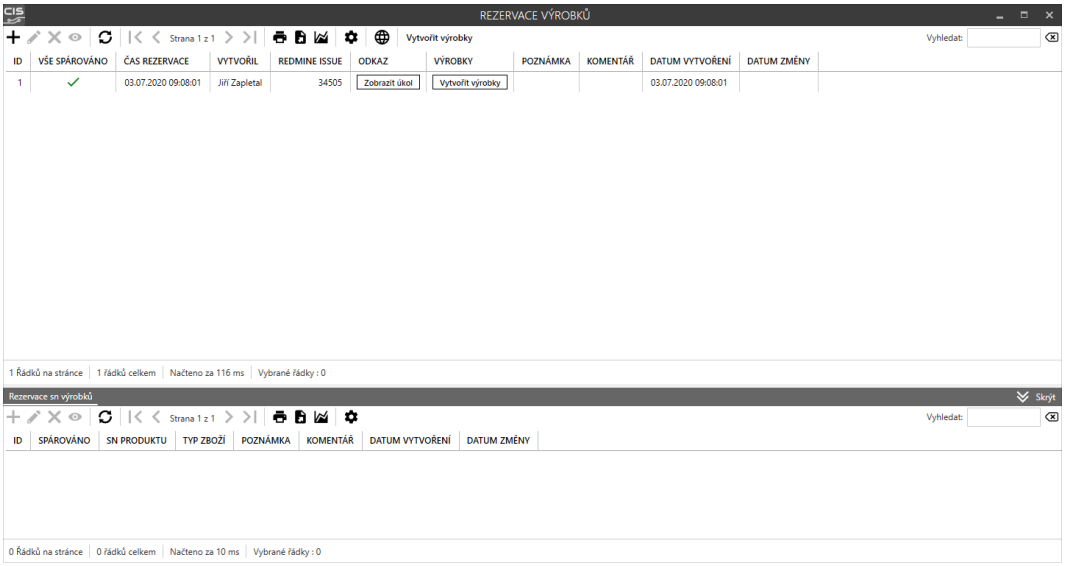
3.3.10.3 Expedice výrobků

Tato funkcionalita vznikla již před delší dobou a nyní se nějakou dobu nepoužívá. Plánuje se ale funkcionalitu upravit a znovu začít používat. Současná funkčnost

lze využít pro složitější vydávání výrobků zákazníkovi. Některé výrobky jsou na výdejce nové, některé jsou repasované a ostatní nahrazují vadné kusy.

3.3.10.4 Rezervace výrobků

Toto vzniklo v nedávné době. Firma se rozhodla některé části finálních výrobků nechávat vyrábět externí firmy. Části výrobků které nejsou vyrobeny u nás jsou v systému vedené jako samostatné subkomponenty a mají své sériové číslo. Toto číslo si přiřazují při výrobě, tudíž by mohl vzniknout konflikt, kdyby někdo na firmě chtěl použít konkrétní sériové číslo, které bude přiřazeno výrobku, který se nevyrábí u nás. Pokud jsou konkrétní sériová čísla zarezervována, Tak při pokusu o vytvoření takového výrobku klasickým způsobem systém uživatele upozorní na existující rezervaci.



Obrázek 10: Rezervace výrobků

3.3.11 Nastavení

3.3.11.1 Tisk protokolů

Jak již bylo zmíněno, z provedených skladových pohybů se někdy následně generuje protokol. Pro různé typy výrobků je potřeba tisknout jiná data na výsledný protokol. Tyto informace se nastavují přesně v tomto okně.

3.3.11.2 Práva na sklady

Tento systém využívají i naši největší zákazníci. Není žádoucí, aby viděli všechny naše výrobky, tudíž jsou vytvořeny práva na viditelnost výrobků pouze na konkrétních skladech. Tato funkcionalita je velice důležitá, ale přináší i velkou obtíž.

Většina zobrazovaných dat v tabulkách musí brát ohled na tyto práva a kvůli tomuto se systém zpomaluje a načítání může být zdlouhavé.

3.3.11.3 Výchozí osoba pro transakce

Při provádění některých skladových pohybů se musí vyplnit osoba, která s výrobkem pracuje. Pokud bude toto pole zaškrtnuté, automatiky se do pole osoba vyplní jméno aktuálně přihlášené osoby.

4 Programátorská dokumentace

V této kapitole čtenáři přiblížím programovou část celého systému, architekturu, napojení na podpůrné knihovny, připojení do databáze.

4.1 Popis adresáře

Zde je zobrazena a popsána kořenová složka projektu.

- Core - obsahuje knihovny a spustitelné přístupové body systému, které poskytuje jádro firemního programu,
- Core/Server - serverová část programu s přístupovým bodem *ISExecutable.exe*
- Core/Client - klientská část programu s přístupovým bodem *ISClientWPF.exe*
- Database - zde je soubor *Readme.md* kde je podrobně popsáno, jak zprovoznit databázi (na výběr jsou tři možnosti),
- Debug - sestavený projekt, včetně závislých knihoven připravený ke spuštění,
- Module - vývojová složka obou modulů včetně závislých knihoven,
- Module/ERP - zdrojové kódy prvního modulu,
- Module/Libs - knihovny vyvíjené nebo upravované firmou, uložené na lokálním nuget serveru a pro možnost použití mimo firmu stažené do této složky,
- Module/WMS - zdrojové kódy hlavního modulu,
- Module/ERP.sln - soubor řešení spustitelný programem Visual Studio pro možnost vývoje systému.

4.2 Popis modulů

Každý modul se nachází v cestě *Module/jméno modulu*. Primárně obsahuje tři další podsložky, neboli tři knihovny. Knihovna *Shared* obsahuje třídy potřebné jak v klientské, tak v serverové části. V klientské části je nejdůležitější třída s názvem *ClientPlugin.cs* implementující rozhraní *IClientPlugin* a *IReleaseNotesProvider*, v serverové je to třída s názvem *ServerPlugin.cs* implementující rozhraní *IServerModule*.

4.2.1 IClientPlugin

Při startu klientské části programu jádro reflexí vyhledává třídy, které implementují rozhraní *IClientPlugin*. Z takových tříd potom vytvoří instance, inicializuje je a zařadí si je mezi moduly. Následuje popis správné implementace rozhraní.

- CodeName - vývojářem zvolené kódové slovo, pod kterým je modul označen,
- Code - číselně označený modul, mělo by to vycházet z CodeName,
- PluginName - jméno modulu, jak se bude zobrazovat v klientské části mezi inicializovanými moduly,
- Description - krátký popis modulu, je poté zobrazen v informacích o systému
- Author - autor modulu
- Version - verze modulu,
- Client - důležitá třída pro práci, pomocí ní můžeme komunikovat se serverem, nebo přistupovat vzdáleně k databázi,
- IsLicenced - označení, zda-li je modul licencovaný,
- RightsLabels - slouží k zaregistrování práv,
- RightsDesc - zobrazuje podrobné popisy k registrovaným právům,
- Start - metoda, která se volá jako první, zde se inicializují naše další třídy,
- Stop - metoda volaná při ukončování aplikace,
- AfterStart - metoda volaná po všech startovacích metodách ve všech modulech,
- HandleLocalMessage - metoda, přes kterou mohou komunikovat klientské moduly mezi sebou, nebo případně externí zařízení,
- Activated - metoda volané po proběhlé aktivaci modulu,
- ProccessData - metoda je vyvolána, pokud server posílá nějaký požadavek bez předchozí žádosti, obvykle využíváno tak, že server všem klientům zahlásí nějakou zprávu.

4.2.2 IReleaseNotesProvider

Pokud inicializovaný klientský modul implementuje toto rozhraní, očekává se, že v třída poskytuje v datovém nosiči *ReleaseNotes* seznam novinek, které poté jádro zobrazuje klientům.

4.2.3 IServerModule

Při startu serverové části programu jádro reflexí vyhledává třídy, které implementují rozhraní *IServerModule*. Z takových tříd potom vytvoří instance, inicializuje je a zařadí si je mezi moduly. Následuje popis správné implementace rozhraní.

- CodeName - vývojářem zvolené kódové slovo, pod kterým je modul označen (ideálně stejné jako klientský modul),
- Code - číselně označený modul, mělo by to vycházet z CodeName (ideálně stejné jako klientský modul),
- Name - jméno modulu, jak se bude zobrazovat v klientské části mezi inicializovanými moduly (ideálně stejné jako klientský modul),
- Description - krátký popis modulu, je poté zobrazen v informacích o systému
- Author - autor modulu
- PluginDependencies - některé moduly mohou být závislé na jiných, pokud do závislostí modulu dáme CodeName jiného modulu, docílíme toho, že závislý modul se inicializuje jako první,
- Version - verze modulu,
- ISServer - důležitá třída pro práci, pomocí ní se registrují entity do entity frameworku, registrují dotazy přes rest api, otevírá se přímý kontext do databáze, přistupuje do konfiguračního souboru a další
- IsLicenced - označení, zda-li je modul licencovaný (ideálně stejné jako klientský modul),
- ClientHandlers - registrace dotazů od klientů,
- DeviceHandlers - registrace dotazů od zařízení,
- HTTPHandlers - registrace dotazů získaných přes http požadavky (dnes již nevyužíváno, použití novější implementace - rest api),
- Init - metoda, která se volá jako první, zde se inicializují naše další třídy,
- Stop - metoda volaná při ukončování aplikace,
- AfterInit - metoda volaná po všech startovacích metodách ve všech modulech,

4.3 Tabulková komponenta

Systém je primárně využívaný ke vkládání dat a k zobrazování a vyhledávání dat z databáze. Pro zobrazování slouží obecná tabulková komponenta *CutterGrid*. Tuto komponentu je k dispozici ve firemní knihovně s názvem *CutterGridWpf*. Pro použití je potřeba vytvořit datového manažera, který slouží pro generování. Vytvořením třídy, která dědí generickou třídu *CutterGridManagerEF* a propojením s konkrétní instancí tabulkové komponenty nám vznikne požadovaný pohled. Generika se v takových případech používá třída, která je přímo mapována přes entity framework na databázovou tabulku nebo databázový pohled. Manager nám nabízí možnost vyhledávat ve všech datech, vyhledávat v jednotlivých sloupcích, řadit sloupce, líné načítání dat metodou stránkování, případně vytvořit závislé pohledy.

4.4 Konfigurační soubor serveru

Pro konfiguraci serveru se používá SqLite databáze, která se ukládá do adresáře serveru do souboru s názvem *Config.dat*, tabulky *t_values*. Tohle je jediná konfigurace, která se v systému používá, tudíž obsahuje mnoho položek. Následuje popis těch nejdůležitějších.

- ListenerPort - port, na kterém server poslouchá,
- DBHost - adresa serveru na kterém běží databáze,
- DBPort - port na kterém databázový host poslouchá,
- Database - název databáze,
- DBUser - jméno databázového uživatele,
- DBPassword - heslo k databázovému uživateli,
- RestEnabled - pokud se nastaví na jedna, při startu se spustí i serverové rest api,
- RestPort - port na kterém poslouchá serverové rest api,
- WMS2_WarehouseDocumentsPath - cesta k adresáři ve kterém jsou uloženy protokoly a další soubory k výrobkům,
- WMS2_edit_material_transaction - počet dní po které lze editovat materiál ve skladových pohybech,

4.5 Sdílené modulové knihovny

Sdílené modulové knihovny se používají k propojení tříd použitých na serveru i na klientovi, jsou tedy závislé na .net standard. Obecně používané třídy tedy jsou:

1. VersionManager - slouží pro zápis aktuálních verzí klientských i serverových modulů,
2. Utils - statické metody pro konverzi nebo kontrolu dat,
3. Requests - čísla požadavků z klienta na server, aby šlo jednoznačně určit, co klient požaduje,
4. Interfaces - rozhraní pro implementaci konkrétního chování v daném kontextu,
5. Enums - enumy používané ve sdílených třídách,
6. DataStructures - datové struktury pro výměnu dat mezi klientem a serverem,
7. Entities - databázové entity mapující se na databázové tabulky, na serveru používané pro složitější dotazy do databáze, na klientovi se používají pro zobrazování dat v tabulkové komponentě,
8. Exceptions - chyby, které se vyvolají na serveru a přenesou se serializací až na klienta.

4.6 Serverový modul WMS

Vstupní bod serverové části je třída ServerPlugin implementující rozhraní IServerModule. V této třídě probíhá mapování entitních tříd na databázové tabulky přes entity framework a dále se zde inicializují tři nejdůležitější části.

4.6.1 Klientské dotazy

Pokud požaduje klientské část systému něco oznámit serveru, nebo jej požádat o data, musí poslat žádost. Systém obsahuje mnoho takových žádostí, některé jsou triviální, za některými se skrývá velká výpočetní síla. Z důvodu orientaci v kódu a jeho čitelnosti vznikli takzvaní manažeři žádostí. *RequestServerApi* je abstraktní třída, kterou manažeři musí dědit. Obsahuje základní princip logování žádostí a metody pro zpracování triviálních žádostí. Konkrétní třídy, které žádosti zpracovávají pak jsou: InitActionRequests, SaveActionRequests, OtherActionRequests, ReservationProductsRequests, OtherReservationProductsRequests, SaveProductsRequests, OtherProductsRequests a OtherMaterialRequests. Jména těchto tříd by měly jednoznačně určovat, jaké žádosti se v nich zpracovávají.

4.6.2 Databázový manažer

Většina práce, která se na serveru vykonává potřebuje přístup do databáze. K tomu slouží třída *DatabaseManager*, která dědí *DatabaseProvider*.

```
1  public abstract class DatabaseProvider : IDisposable
2  {
3     protected DatabaseProvider(IServer server)
4     {
5        Server = server;
6     }
7
8     protected IServer Server { get; private set; }
9
10    protected void Log(LogLevel level, string message) => Server.Log(
          level, $"[{GetType().Name}] {message}");
11
12    public virtual void Dispose()
13    {
14       Server = null;
15    }
16
17    public DbContext GetDbContext() => Server.GetEFContext();
18
19    public async Task<List<T>> SqlQuery<T>(string sqlName, params
          object[] parameters) where T : class
20    {
21       await using var ctx = GetDbContext();
22       return await SqlQuery<T>(ctx, sqlName, parameters);
23    }
24
25    public Task<List<T>> SqlQuery<T>(DbContext ctx, string sqlName,
          params object[] parameters) where T : class
26    {
27       var sql = Server.GetQueryText(sqlName);
28       return ctx.Set<T>().FromSqlRaw(sql, parameters).ToListAsync();
29    }
30 }
```

Zdrojový kód 1: Třída DatabaseProvider

DatabaseManager dále obsahuje jednotlivé providery pro práci s danými entitami nebo skupinami entit. Každý tento provider také dědí *DatabaseProvider*. Příkladem to jsou: ProductDatabaseProvider, MaterialDatabaseProvider, TransactionDatabaseProvider, OperationDatabaseProvider, ParameterDatabaseProvider a jiné.

4.6.3 Rest api

Třetí nejpodstatnější částí serverového modulu je zpracovávání dotazů přes rest api. Api využívají malé aplikace vyvinuté firmou, které slouží pro testování výrobků a následného zapisování dat z provedených testů do systému. Třída pro tuto práci nese název *RestApiManager*, kde se požadavky registrují a následně i zpracovávají. V jádru serveru je implementovaný rest api server přes knihovnu Nancy.

4.6.4 Ostatní části serveru

1. CheckMaterializedView - v databázi používáme jeden materializovaný pohled. Ten se liší od klasických pohledů tím, že své výstupní záznamy má uložené v tabulce. Při takovém použití je potřeba po změně v datových tabulkách provést přegenerování materializovaného pohledu. K tomu slouží CheckMVManager, který v pravidelných časových intervalech zjišťuje ze systémových tabulek, zda-li byla provedena změna a případně zahájí přegenerování,

2. ContractsSynchronization - třída, která se přes rest api dotazuje do účetního programu na zakázky a ty pak synchronizuje do našeho systému pro odepisování materiálu na tyto zakázky

3. Queries/wms.dbq - soubor obsahující složitější databázové dotazy v jazyku sql, které ještě nebyly převedeny na dotazy do jazyka C#, nebo není žádoucí je převádět, kvůli časové náročnosti zpracování těchto dotazů.

4.7 Klientský modul WMS

Vstupním bodem pro klientskou část modulu je třída ClientPlugin implementující rozhraní IClientPlugin. V inicializační metodě se vytváří dvě záložky (sklad výrobků a sklad materiálu) a položky v menu s názvem Sklad. Dále se zde registrují zobrazovatelná podokna aplikace.

4.7.1 Volání dotazů na server

Jak již víme, na serveru jsou registrované žádosti od klienta, tudíž existuje koncept volání těchto serverových dotazů. Pro toto volání existuje statický singleton, což je jediná instance třídy RequestManager, který poskytuje skupiny žádostí. Každá taková skupina je samostatná třída, která dědí z třídy RequestClientApi, která obsahuje logování žádostí a délku žádostí. Dále nabízí dvě metody pro volání primitivních žádostí na server, které se na serveru zpracovávají také pomocí primitivních metod.

4.7.2 ViewModels

Klientská část pro zobrazování oken je psána ve stylu MVVM. Datovou části oken tedy jsou takzvané view modely. Každý model dědí ze třídy WmsViewModel, která poskytuje instance tříd ClientPlugin pro práci, kterou nabízí jádro, RequestManager pro zasílání dotazů na server, RightManager pro práci s načtenými právy na sklady pro aktuálního uživatele. Výměna dat do gui a zpět probíhá pomocí bindování na property. Klikání myší na tlačítka se přenáší do modelu přes commandy, které vyvolají příslušnou metodu, která požadavek obstará. Pokud se jedná o potvrzovací tlačítko, tak si metoda načte probindovaná data z propert, vytvoří z nich datové struktury a zašle je na server, který vykoná danou práci.

4.7.3 Views

Podokna zobrazované z hlavního okna aplikace. Každé takové okno má svého nositele dat (view model). Okna jsou tvořeny z user control, nebo primitivních komponent, jako jsou tlačítka boxy pro text, výběrové komponenty, záložky a další. Celé vykreslování grafiky používá Mahhaps Metro design. Důležitou poznámkou je, že při používání stylu MVVM je obecná poučka: V code behind, což je součást okna psána v C# je co nejméně kódu. Vše co souvisí se zobrazením má být psáno v xaml a vše co souvisí s daty má být ve view modelu.

4.7.4 PrintProtocol

Klientská část systému nabízí možnost tisknout data tiskárnou. Konkrétní výstupní protokoly mi generují samostatné třídy, které dědí systémovou třídu DocumentPaginator. V implementaci těchto tříd je potřeba implementovat několik propert (počet stran, velikost stránky a jestli se má tisknout ještě další strana) a metodu GetPage, kterou si již volá samotný ovladač tiskárny. Jelikož je žádoucí mít na každé straně v zápatí počet stran, musí se celý dokument předgenerovat hned na začátku a vyplnit zástupnými znaky pro data, která v tu chvíli neznáme. Při opakovaném volání metody pro zisk aktuální stránky se strana nevytváří, ale pouze vyzvedne z fronty stránek a zástupné znaky se nahradí reálnými daty. Poté je stránka vytištěna.

4.7.5 GridManagers

Tato složka obsahuje třídy, které dědí CutterGridManagera a jsou potřeba pro tabulkovou komponentu. Těchto managerů je velice mnoho, protože musí být každá databázová tabulka napojena právě na jednoho, aby se daly záznamy upravovat. Všichni tito manageři dědí generickou bázovou třídu WmsGridManagerEF, jejíž zdrojový kód je zde k dispozici 2.

Tato bázová třída obsahuje funkčnosti pro přegenerování dat, pokud se změnily práva na sklady, serverový zámek na právě upravovaný záznam, RequestMa-

nagera pro zasílání dotazů na server a obecnou metodu pro zachytávání chyb při úpravě záznamů.

4.8 Popis databáze

Databáze pro uchovávání všech dat, jak již bylo řečeno, je použita PostgreSQL. Jádro systému má své vlastní schéma s názvem CIS. Každý modul má své schéma, tudíž existují další dvě s názvem erp a wms. Schéma system je globální a je využívané pro definici obecných funkcí. Všechny databázové tabulky mají svou ekvivalentní vytvořenou třídu v entitách. Popis jednotlivých tabulek by měl být pochopitelný z názvu, dále je pak dohledatelný v databázových scriptech. Při tvorbě databázových funkcí a pohledů pro sklad materiálu a výpočty jeho ceny jsem čerpal z literatury číslo 3.

```
1  public abstract class WmsGridManagerEF<T> : ErpGridManagerEF<T>
       where T : class
2      {
3          protected WmsGridManagerEF(IRightManager rightManager)
4          {
5              RightManager = rightManager;
6              rightManager.WarehouseRightsChanged +=
                   OnWarehouseRightChanged;
7          }
8
9          public RequestManager RequestManager { get; }
10         protected IRightManager RightManager { get; }
11
12         public override void HandleError(GridAction action, Exception
               ex)
13         {
14             if (!(ex.InnerException is DBException dbEx))
15             {
16                 base.HandleError(action, ex);
17                 return;
18             }
19             if (action == GridAction.Delete && dbEx.Error == DBError.
                   ForeignKeyViolaion)
20                 WindowService.ShowError("Záznam nelze odstranit, protože
                       je použitý v jiné tabulce.", "Chyba");
21             else if (action == GridAction.Add && dbEx.Error == DBError.
                   UniqueViolation)
22                 WindowService.ShowError("Záznam nelze přidat, protože
                       nemá unikátní identifikátor \n (Záznam s tímto jmé
                       nem již existuje).", "Chyba");
23             else base.HandleError(action, ex);
24         }
25
26         public Task<LockState> ServerLock(Func<string, LockState>
               serverFunc, T item)
27         {
28             return Task.Run(() => serverFunc($"{nameof(T)};{item.
                   GetHashCode()}"));
29         }
30
31         protected virtual async void OnWarehouseRightChanged()
32         {
33             await RefreshItems();
34         }
35     }
```

Zdrojový kód 2: Třída WmsGridManagerEF

5 Budoucnost aplikace

Za necelé tři roky práce na tomto systému se neodvažuji říct, kterým směrem se bude systém posouvat. Požadavků na úpravy a implementace nových funkčností přibývá každým dnem několik, ale hlavně z mnoha stran. Ekonomické oddělení by chtělo přidat jinou funkčnost než třeba obchodní oddělení, ale výroba chce také něco jiného. Z tohoto důvodu budu muset požadovat, aby mi byl přidělen člověk, který bude koordinovat a rozřazovat požadavky od uživatelů, abych měl více času na vývoj a programování. V současné době si všechny plány řídím sám. Můj osobní požadavek bude převedení celkového systému na .net core, případně na nový plánovaný .net 5. Co se ale týče nových funkcí, ty nechám na novém koordinátorovi, jen vím, že budou.

Závěr

Cílem této práce bylo vyvinout a popsat systém řízení skladu. Systém je plně funkční a využívá se ve společnosti Cutter Systems spol. s r.o. Je používaný ve všech odděleních, tudíž vytvoření systému se zdařilo. Fungování firmy bez tohoto systému si nyní nedokážu představit. Dále systém pomohl rozvoji CISu, jelikož pro zprovoznění různých funkcionalit bylo zapotřebí rozvinout i jádro firemního systému.

Při práci jsem se velice přiučil v databázových návrzích systémů a vytváření složitých dotazů. V jazyce C# jsem se také zdokonalil, jelikož je celý systém psán v nejnovější verzi jazyka s použitím nejnovějších technologií. Co se týče architektonického návrhu, mohly by být části více rozděleny, aby mohly fungovat samostatněji a zvýšila by se přehlednost kódu. V době vytváření systému jsem ale neznal mnoho návrhových vzorů, tudíž je systém takto napsaný.

Conclusions

The aim of this work was to develop and describe a warehouse management system. The system is fully functional and is used in the company Cutter Systems spol. s r.o. It is used in all departments, so the creation of the system was successful. I can't imagine how a company would work without this system. Furthermore, the system helped the development of CIS, because to put various functionalities into operation, it was necessary to develop the core of the company system.

During this work, I learned a lot in database system design and creating complex queries. I also improved in C# because the whole system is written in the latest version of the language using the latest technologies. In terms of architectural design, parts could be more divided to work more independently and increase code clarity. At the time of creating the system, I did not know many design patterns, so the system is written this way.

A Obsah přiloženého CD

`bin/`
: Kompletně sestavený celý projekt připravený ke spuštění.

`doc/`
: Text práce ve formátu PDF, vytvořený s použitím daného stylu pro závěrečné práce, včetně všech příloh.

`install/`
: Instalátory aplikací, scripty pro zprovoznění databáze.

`src/`
: Kompletní zdrojové texty systému.

`readme.md`
: Instrukce pro zprovoznění celého systému.

Bibliografie

[1] IAN GRIFFITHS Programming C sharp 8.0: Build Cloud, Web, and Desktop Applications.

[2] JON P SMITH Entity Framework Core in Action.

[3] ECOMDASH The Ultimate Guide To Inventory Management for Multi-channel Retailers, Dostupné z: https://www.ecomdash.com/inventory-guide-multichannel-retailing/principles-inventory-management/.

[4] HANS-JÜRGEN SCHÖNIG, Mastering PostgreSQL 11: Expert techniques to build scalable, reliable, and fault-tolerant database applications.