Vyšší odborná škola, Střední škola,
Centrum odborné přípravy

ABSOLVENTSKÁ PRÁCE

Návrh aplikace pro evidenci a kontrolu majetku s využitím čárových kódů

Sezimovo Ústí, 2026 **Autor: Adéla Marieta Horáková**

VYŠŠÍ ODBORNÁ ŠKOLA, STŘEDNÍ ŠKOLA, CENTRUM ODBORNÉ PŘÍPRAVY

SEZIMOVO ÚSTÍ, BUDĚJOVICKÁ 421

ZADÁNÍ ABSOLVENTSKÉ PRÁCE

Student:	Adéla Marieta Horáková
Obor studia:	26-41-N/01 Elektrotechnika – mechatronické systémy
Název práce:	**Návrh aplikace pro evidenci a kontrolu majetku s využitím čárových kódů**
Anglický název práce:	**Application design for property registration and control using barcodes**

Zásady pro vypracování:

1. Popište stávající systém evidence a kontroly majetku školy.
2. Vysvětlete princip čárového kódu a funkci čtečky čárových kódů.
3. Navrhněte a naprogramujte aplikaci pro evidenci a kontrolu majetku laboratoře aplikované informatiky s využitím čárových kódů.
4. Navrhněte možné využití vytvořené aplikace pro evidenci a kontrolu majetku pomocí čárových kódů pro celou školu.
5. Absolventskou práci vypracujte problémově ve struktuře odpovídající vědecké práci.

Doporučená literatura:

[1] SHARP, John. *Microsoft Visual C# 2010. Krok za krokem.* 1. vyd. Brno: Computer Press, 2010. 696 s. ISBN 978-80-251-3147-3.

[2] BENADÍKOVÁ, Adriana. *Čárové kódy. Automatická identifikace.* 1. vyd. Praha: Grada, 1994. ISBN 80-85623-66-8.

Vedoucí práce:	Mgr. Ludmila Jůzová, VOŠ, SŠ, COP Sezimovo Ústí
Odborný konzultant práce:	Ing. Jiří Roubal, Ph.D., VOŠ, SŠ, COP Sezimovo Ústí
Oponent práce:	Ing. Luboš Řičica, VOŠ, SŠ, COP Sezimovo Ústí

Datum zadání absolventské práce: **1. 9. 2025**

Datum odevzdání absolventské práce: ~~**15. 5. 2026**~~

Mgr. Ludmila Jůzová
(vedoucí práce)

VYŠŠÍ ODBORNÁ ŠKOLA, STŘEDNÍ ŠKOLA, CENTRUM ODBORNÉ PŘÍPRAVY, Budějovická 421, Sezimovo Ústí

doc. PhDr. Mgr. Lenka Hrušková, Ph.D.
(ředitel školy)

V Sezimově Ústí dne 1. 9. 2025

Prohlášení

Prohlašuji, že jsem svou absolventskou práci vypracovala samostatně a uvedla jsem veškeré použité informační zdroje, veškerý použitý software a dodržela jsem Metodiku užití „umělé inteligence“, která je součástí metodiky tvorby absolventské práce.

Prohlašuji, že jsem v průběhu příprav a psaní této práce nepoužila nástroje „umělé inteligence“. Stvrzuji, že jsem si vědoma, že za obsah této práce plně zodpovídám.

V Sezimově Ústí dne 11. 5. 2026

Horáková

podpis

Poděkování

Děkuji především vedoucí absolventské práce Mgr. Ludmile Jůzové za podporu a pomoc při tvorbě této práce. Také děkuji panu učiteli Ing. Jiřímu Roubalovi, Ph.D. za cenné rady při psaní této práce.

Anotace

Absolventská práce se zabývá tvorbou aplikace pro evidenci a kontrolu majetku s využitím čárových kódů. Práce stručně seznamuje s inventarizací, jejími druhy. Řeší problematiku vylepování čárových kódů na majetek organizace. Charakterizuje čárové kódy a stručně pojednává o čtečkách čárových kódů. Následně popisuje samotnou tvorbu aplikace pro kontrolu a evidenci majetku. Aplikace se skládá ze dvou hlavních částí: databáze uchovávající seznamy majetku a aplikace, která napomáhá při kontrole a evidenci majetku. V práci jsou popsány programové kódy aplikace a také vzhled uživatelského rozhraní.

Klíčová slova: C#; čárový kód; čtečka čárových kódů; WPF aplikace.

Annotation

This thesis focuses on the development of an application for asset tracking and management using barcodes. The thesis provides a brief overview of inventory management and its various types. It addresses the issue of affixing barcodes to an organization's assets. It describes barcodes and briefly discusses barcode scanners. It then describes the actual development of the application for asset control and tracking. The application consists of two main parts: a database storing asset lists and an application that assists with asset control and tracking. The thesis describes the application's source code as well as the user interface design.

Keywords: C#; barcode; barcode scanner; WPF application.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Každá organizace je ze zákona č. 563/1991 Sb. povinna evidovat svůj majetek a zajistit jeho správnou kontrolu a aktualizaci v případě změny (Daně pro lidi, 2024). Evidence majetku organizaci umožňuje přehled o tom, jaký vlastní majetek, proto je velmi důležitou součástí. Tato povinnost se vztahuje i na naši školu, která musí provádět inventarizaci svého majetku v souladu s předpisy. Inventarizace je proces, při kterém dochází k fyzické kontrole stavu majetku. Tento stav se porovnává se stavem uvedeným v evidenci majetku. Tento proces bývá často velmi časově a organizačně náročný. Obzvláště ve velkých organizacích, jako je třeba naše škola, kde je majetek rozptýlen do několika budov a velkého počtu místností.

V současnosti se na naší škole eviduje majetek v softwarové aplikaci, ze které se generují seznamy majetku pro jednotlivé místnosti školy. Tyto seznamy jsou následně vytištěny a vyvěšeny v každé místnosti, kde slouží jako podklad pro inventarizaci. Každá položka majetku, přesahující hodnotou stanovenou částku, je označena svým unikátním inventárním číslem. Toto označení je však často na velmi špatně dostupném místě. Toto umístění komplikuje celý proces inventury. Inventura probíhá ve dvou fázích. Nejprve probíhá takzvaná předinventura. Při té se vytisknou soupisy majetku, které jsou následně rozděleny mezi učitele, kteří jsou odpovědní za provedení inventury. Učitelé poté se soupisy projdou učebny, které jim byly přiděleny. Poté probíhá hlavní inventura podobným způsobem, kdy majetek kontroluje ustanovená inventarizační komise. Ta slouží k hlavní kontrole majetku. Tento proces je velmi časově a fyzicky náročný, což by se dalo v současné době, kdy existují moderní technologie využitelné pro inventarizaci, urychlit a zjednodušit. Na trhu se prezentují firmy, které nabízejí vlastní softwarová řešení. Tato řešení jsou ale velmi finančně náročná a také vyžadují zaškolení všech uživatelů, kteří jej budou využívat.

Z tohoto důvodu vznikla myšlenka na vytvoření vlastní aplikace pro evidenci a kontrolu majetku, která by umožnila zjednodušení celého procesu. **Cílem** této práce je vytvořit aplikaci, pomocí které bude možné spravovat databázi majetku školy. S pomocí této databáze se bude s využitím čárových kódů kontrolovat, zda souhlasí majetek se soupisem. Aplikace usnadňuje práci při inventuře a urychluje celý proces kontrolování, zda souhlasí soupis majetku se seznamem v databázi. Součástí práce je také vytvořená databáze majetku a vygenerované čárové kódy.

Struktura této práce je následující. V kapitole 2 je popsána problematika inventarizace majetku a způsoby, kterými je možné vylepit čárové kódy. Kapitola 3 vysvětluje problematiku čárových kódů a čteček čárových kódů. V kapitole 4 je popsána tvorba vlastní aplikace a generování čárových kódů.

Kapitola 2

Inventarizace a způsob vylepení čárových kódů

V této kapitole je popsána problematika inventarizace, jaké musí splňovat podmínky a jakým způsobem musí probíhat a jaké náležitosti musí splňovat její výstupy. Dále kapitola popisuje, jakými způsoby je možné čárové kódy vytisknout nebo vyrobit, jaké lze zvolit velikosti, aby bylo možné je přečíst pomocí čtečky čárových kódů, a jakými způsoby je lze vylepit, aby byly nejlépe čitelné a nejlépe držely.

2.1 Inventarizace

Inventarizace majetku je nutná podmínka pro průkaznost účetnictví podle § 8 odst. 4 zákona č. 563/1991 Sb., o účetnictví. Inventura musí být provedena u veškerého majetku i u veškerých závazků podle § 6, § 29 a § 30 zákona č. 563/1991 Sb., o účetnictví (Daně pro lidi, 2024). Inventarizace majetku se skládá ze tří etap:

- inventura,
- srovnání zjištěného s účetním stavem,
- zjištění, zda je potřeba provést odpis, nebo vytvořit opravnou položku.

Pojmy inventura a inventarizace jsou odlišné. Inventarizace obsahuje velké množství různých prací, které jsou potřebné pro účetnictví. Inventura je jedna z částí inventarizace, při které se zjišťuje skutečný stav majetku.

Pro provedení každé inventury vznikají takzvané inventární soupisy, které musí obsahovat tyto náležitosti:

1. Určení inventarizovaného majetku a jeho množství.
2. Podpis osoby, která je za provedení inventarizace majetku odpovědná.
3. Podpis osoby, která je odpovědná za zjištění stavu inventarizovaného majetku.
4. Způsob, kterým je zjišťovaný skutečný stav.
5. Ocenění majetku k okamžiku ukončení inventury.
6. Den zahájení a den ukončení inventury.

Tento soupis musí dále obsahovat vyčíslení rozdílů mezi skutečným stavem a stavem účetním.

2.1.1 Druhy inventarizace

Rozlišuje se několik druhů inventarizace. Tyto druhy se rozdělují podle toho, kdy se inventarizace provádí nebo podle toho, jaký je její rozsah. Podle toho, kdy se provádí, se dělí na:

- pravidelné inventarizace,
- periodické – prováděné za každé účetní období,
- průběžné – prováděné průběžně během účetního období,
- mimořádné.

Podle rozsahu inventarizace se dělí na:

- úplnou – týká se všech složek,
- dílčí – týká se jen některých složek.

Dále se inventarizace dělí na fyzickou inventuru, která se provádí u fyzického majetku a na dokladovou, která se provádí u položek, které svou povahou neumožňují fyzickou inventuru. Inventura se musí uskutečnit alespoň jedenkrát ročně (MÁČE, M., 2013). Mohou nastat dva typy inventarizačních rozdílů:

- manko – skutečný stav je nižší než účetní,
- přebytek – skutečný stav je vyšší než účetní.

2.2 Způsoby vylepení čárových kódů

Čárové kódy je možné vytisknout na běžných tiskárnách, gravírovat do kovových destiček nebo na tyto destičky natisknout. Kódy, které jsou gravírované v kovových destičkách, lze přilepit nebo přišroubovat. Tento typ je ale příliš náročný na pořízení, proto se jím nebude tato práce dále zabývat. Kódy vytištěné na papír se dají nalepit pomocí lepidla, izolepy a nebo se dají vytisknout rovnou na samolepky. Tyto kódy lze také zalaminovat, aby se při případném odtržení nepoškodily. V rámci práce bylo zkoumáno, jaký způsob přilepení kódů je nejlepší na různých materiálech. Pro zkoumání byly využity tyto materiály:

- dřevo,
- plast,
- kov.

Jako první bylo zkoumáno přilepení čárových kódů na dřevěnou desku. Deska byla nejprve očištěna od nečistot, aby kódy pořádně přilnuly. Na desku byl nalepen kód, který byl vytištěný přímo na samolepce od společnosti Zebra. Dále bylo vyzkoušeno několik možností nalepení kódu vytištěného na papír. Byly vyzkoušeny čtyři způsoby přilepení kódu, který byl vytištěný na papír. První kód byl zalaminovaný do izolepy a následně přilepen pomocí lepidla. Druhý byl nalepený pouze pomocí lepidla. Další kód byl nalepený pomocí lepicí pásky. Poslední kód byl nalepen lepidlem a následně přetřen lepidlem z vrchní strany. Na obrázku 2.1 je zobrazeno nalepení kódů na dřevěnou desku. Nejlépe nalepený drží kód vytisknutý na samolepku, kód, který byl vytištěný na obyčejný papír a přetřený lepidlem i na povrchu, a kód přilepený lepicí páskou. Ostatní kódy se začaly ihned po zaschnutí lepidla odlepovat. Také bylo zkoušeno kódy odtrhnout. Odtrhnout se dají všechny kódy, ale u některých způsobů nalepení to není tak snadné. Nejsnazší na stržení je kód nalepený pomocí lepidla.

Jako další bylo zkoušeno nalepení kódů na kousek kovu. Kov byl před zkoušením očištěný pro lepší přilnutí lepidla. Na kovu byl ozkoušen kód vytištěný na speciální samolepku s plastem. Dále bylo vyzkoušeno nalepení kódu vytištěného na papír stejnými způsoby jako na dřevěnou desku. Na kovu držely dobře přilepené skoro všemi způsoby. Jediný kód, který byl nejprve zalaminovaný do lepicí pásky a následně přilepený lepidlem, na kovu nedrží a ihned se odlepuje. Ostatní kódy drží přilepeny dobře. Ukázka kovu s nalepenými kódy je na obrázku 2.2.

Jako poslední bylo zkoušeno nalepit kódy na kousek plastu. Lepení kódů bylo opět prováděno stejnými způsoby. Materiál byl před lepením očištěn od nečistot. Na plastu drží nalepené kódy všemi vyzkoušenými metodami, až na kód, který byl zalaminovaný do lepicí pásky. Tento kód se na materiál pořádně nenalepí, protože lepidlo, kterým je přilepený, pořádně nezaschne ani po dlouhé době. Ukázka nalepení kódů na plastovou desku je na obrázku 2.3.

Při zkoušení způsobů přilepení kódů bylo zjištěno, že nejlepší způsob nalepení je samolepka, se kterou se nejlépe pracuje a je nejrychleji nalepená. Dále je rychlé a snadné kód přilepit pomocí lepicí pásky. Ostatní způsoby jsou zdlouhavé z důvodu dlouhého čekání na zaschnutí lepidla. Také s kódy, na které je nutné natřít lepidlo, se špatně pracuje.

Kódy jsou na dřevěnou desku, která je zobrazena na obrázku 2.1, nalepeny následujícím způsobem:

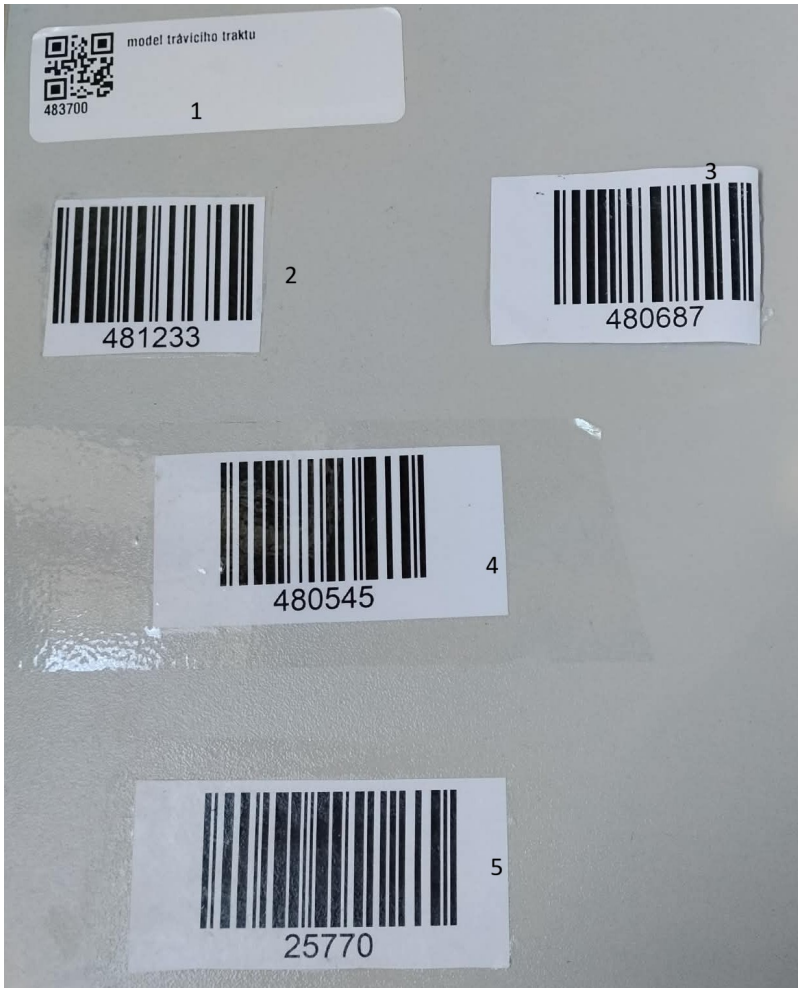
1. Kód vytištěný na samolepku.
2. Kód vytištěný na papír zalaminovaný pomocí lepicí pásky a nalepený pomocí lepidla.
3. Kód přilepený pomocí lepidla.
4. Kód přelepený lepicí páskou.
5. Kód přelepený lepidlem i z vrchní strany.

Kódy jsou na kovovou desku, která je zobrazena na obrázku 2.2, nalepeny následujícím způsobem:

1. Kód vytištěný na speciální samolepku.
2. Kód přilepený pomocí lepidla.
3. Kód přelepený lepidlem i z vrchní strany.
4. Kód přelepený lepicí páskou.
5. Kód vytištěný na papír zalaminovaný pomocí lepicí pásky a nalepený pomocí lepidla.

Kódy jsou na plastovou desku, která je zobrazena na obrázku 2.3, nalepeny následujícím způsobem:

1. Kód vytištěný na samolepku.
2. Kód přilepený pomocí lepidla.
3. Kód přelepený lepicí páskou.
4. Kód vytištěný na papír zalaminovaný pomocí lepicí pásky a nalepený pomocí lepidla.
5. Kód přelepený lepidlem i z vrchní strany.



Obrázek 2.1: Ukázka nalepení kódů na dřevěnou desku

Obrázek 2.2: Ukázka nalepení kódů na kov

Obrázek 2.3: Ukázka nalepení kódů na plast

Kapitola 3

Čárové kódy a čtečky čárových kódů

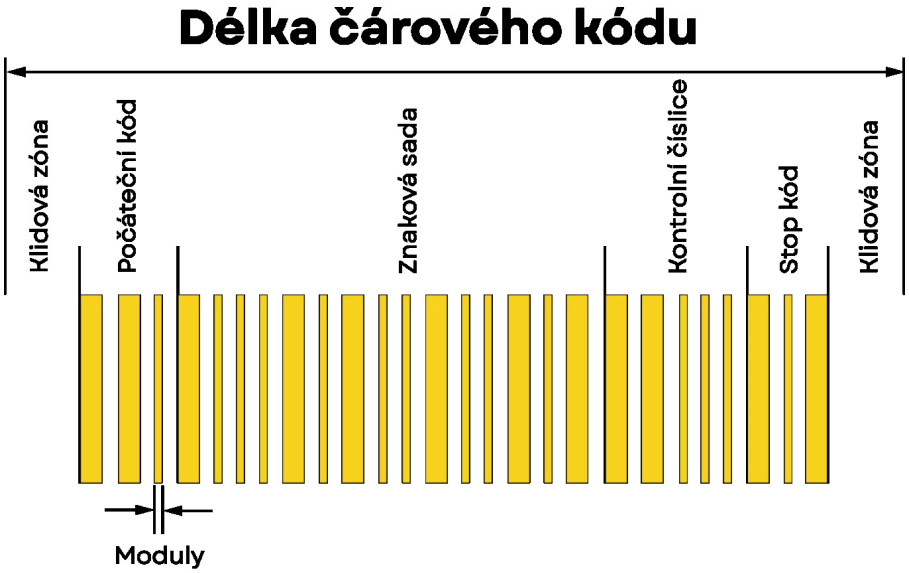
Tato kapitola se zabývá problematikou čárových kódů, které jsou v dnešní době hojně využívány pro usnadnění práce. Jakými způsoby se čárové kódy rozdělují. Jaké znaky do čárových kódů můžeme zašifrovat. Také popisuje, jakou mají kódy konstrukci a jak lze kódy vytvořit. Kapitola také popisuje čtečky čárových kódů. Jaké jsou druhy čteček čárových kódů, jak fungují a jakým způsobem snímají čárové kódy.

3.1 Čárové kódy

Existuje několik druhů čárových kódů. Kódy lze rozdělit do dvou základních skupin. Na kódy, které se využívají v obchodě, a na kódy využívané v průmyslu. Základní kritérium, podle kterého lze porovnávat čárové kódy, je kódovací tabulka jednotlivých druhů. Dále lze čárové kódy porovnávat podle znaků, které jsou v čárových kódech uvedeny. Na základě těchto znaků lze kódy rozdělit na numerické, numerické se speciálními znaky, alfanumerické a úplně alfanumerické. Dalším kritériem pro porovnávání je pevná nebo variabilní délka kódu. Kódy, které se využívají v obchodu, mají pevně danou délku. Kódy používané v průmyslu umožňují zakódovat variabilní počet znaků. Mezi kódy s pevnou délkou patří například: EAN 8 nebo EAN 13. Mezi kódy s variabilní délkou řadíme kódy: Code 39, Code 128, Codabar a Code 2/5. Dále se rozlišuje speciální typ čárového kódu, který se nazývá QR kód. Čárové kódy se podle typu dělí také na jednodimenzionální a dvoudimenzionální (Benadíková, A. et al., 1994).

3.1.1 Konstrukce čárových kódů

Všechny typy čárových kódů se skládají z různě širokých čar a mezer. Informaci v čárovém kódu nesou jak čáry, tak mezery. Každý typ kódu má specifická pravidla pro řazení čar a mezer a jejich šířku. Každý kód má svoji šifrovací tabulku. Na začátku každého kódu je pomocí specifické sekvence čar a mezer zakódovaný znak start (počáteční kód). Také na konci každého kódu je specificky zakódovaný znak stop (stop kód).



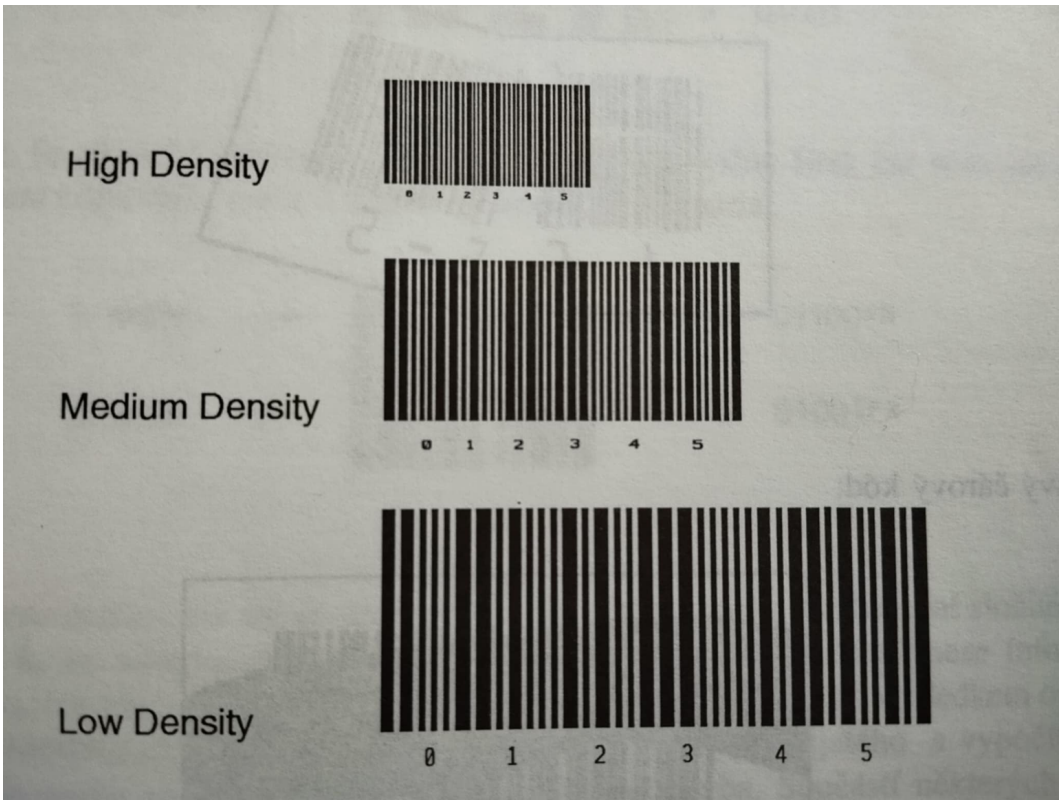
Obrázek 3.1: Konstrukce čárového kódu[1]

Na obrázku 3.1 je zobrazena konstrukce čárového kódu. Počáteční kód a stop kód zajišťují, že je kód čitelný jak zleva, tak zprava. Klidová zóna je prázdná oblast na začátku a konci kódu, která má minimálně 6,35 mm. Díky této zóně čtečka může přečíst čárový kód. Modul je šířka čáry v čárovém kódu. Kód obsahuje znakovou sadu, ve které je zakódované požadované číslo. Dále obsahuje kontrolní číslice (SMART-TEC, 2024). Pro tvorbu čárových kódů lze využít dostupné online generátory. Každý typ čárového kódu může být vytvořen v různých velikostech. Velikost čárového kódu závisí na velikosti modulu.

[1]Obrázek převzat z ⟨https://www.smart-tec.com/cs/know-how/znalost-auto-id/carove-kody⟩.

Podle hustoty čárového kódu rozlišujeme tyto skupiny:

- High Density (vysoká hustota),
- Medium Density (střední hustota),
- Low Density (nízká hustota).



Obrázek 3.2: Ukázka hustoty čárového kódu – převzato z (Benadíková, A. et al., 1994)

Na obrázku 3.2 je zobrazen stejný čárový kód s různou hustotou, hustota kódu je určena velikostí zvoleného modulu. Zobrazený kód je 2/5 Industrial, který je numerický kód s variabilní délkou. Pro úspěšné přečtení čárového kódu je také nutné dodržet kontrast. Tato hodnota je určena poměrem mezi rozdílem odrazu pozadí a odrazu čárky ku odrazu pozadí (Benadíková, A. et al., 1994). V případě jakéhokoliv poškození čárového kódu může dojít ke znemožnění jeho přečtení čtečkou čárových kódů. Čárové kódy se dají rozdělit na jednodimenzionální kódy, které jsou ukázány na obrázku 3.3, a na dvoudimenzionální kódy, které zobrazuje obrázek 3.4.

(a) Code 39

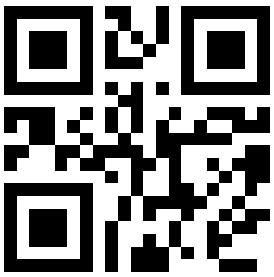
(b) Code 93

(c) Code 128

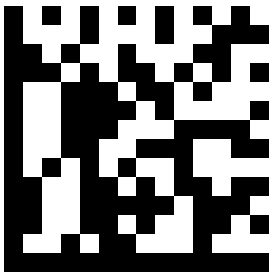
(d) EAN 8

(e) EAN 13

Obrázek 3.3: Jednodimenzionální čárové kódy[2]



(a) QR kód



(b) Datamatrix

Obrázek 3.4: Dvoudimenzionální čárové kódy[3]

[2]Obrázky vygenerovány pomocí online generátoru ⟨https://generator.dantem.net/⟩.
[3]Obrázky vygenerovány pomocí online generátoru ⟨https://barcode.tec-it.com/en/⟩.

3.2 Čtečky čárových kódů

Čtečky čárových kódů jsou relativně malá zařízení, která čtou čárové kódy. Čtečky vyzařují paprsek světla, který dopadá na čárový kód. Světlo se odráží od povrchu čárového kódu, čtečka poté analyzuje rozdíly, které způsobují tmavé a světlé pruhy. Toto odražené světlo se převádí na elektrický signál. Poté, co čtečka dekóduje data, odešle tato data do používaného počítačového systému, který umožní další použití tohoto kódu.

3.3 Typy čteček čárových kódů

Čtečky čárových kódů rozdělujeme do několika skupin:

- laserové čtečky,
- CCD čtečky,
- obrázkové čtečky,
- čtecí pera.

3.3.1 Laserové čtečky

Laserové čtečky využívají pro přečtení čárového kódu laserový paprsek. Paprsek je směřován pomocí elektromechanického systému, který pohybuje zrcátkem, nebo rotujícím hranolem. Paprsek se odrazí od čárového kódu a poté je snímám fotočidlem. Paprsek je převeden na digitální signál, tento signál zpracuje procesor čtečky, který informaci předá počítači.

3.3.2 CCD čtečky čárových kódů

CCD čtečky využívají ke zpracování odraženého světla pole fotodiod, které se označuje CCD (Charge Coupled Device). Světlo, které se odrazí od čárového kódu, se zaostří pomocí čočky a následně nasměruje do CCD snímače. U těchto čteček není potřeba hýbat s paprskem či čtečkou pro přečtení čárového kódu. Tento typ čteček je velmi jednoduchý na obsluhu.

3.3.3 Obrázkové čtečky čárových kódů

Obrázkové čtečky čárových kódů pracují s využitím CCD snímačů. Čtečka vyfotí kód, který je následně pomocí softwarového zpracování snímaného obrazu dešifrován. Výhodou těchto čteček je možnost přečíst více kódů, které vidí při jednom záběru. Tato funkce se musí zapnout v nastavení čtečky. Nevýhodou je vysoká náročnost na osvětlení a nutnost malé vzdálenosti pro přečtení kódu. Obdobně pracují čtečky v mobilních telefonech (Tůma, V., 2009).

3.3.4 Čtecí pera

Pero ke čtení čárového kódu se podobá běžnému kuličkovému peru. Z tohoto důvodu je nazýváno čtecím perem. Čtecí pero se skládá z hrotu, zdroje světla a optického systému. Nejdůležitějším parametrem je kvalita hrotu, který je hodně mechanicky namáhán při čtení čárového kódu, kterého se musí hrot pro přečtení dotýkat. K poškození může dojít například kvůli částicím prachu. Při poškození dochází ke zhoršení kvality čtecího zařízení. Hroty jsou nejčastěji vyráběny ze syntetických materiálů. Za nejkvalitnější hroty jsou považovány rubínové, které jsou nejnáročnější na výrobu. Jako zdroj světla jsou ve čtecích perech využívány LED diody. Pera mohou být jednodiodová nebo dvoudiodová.

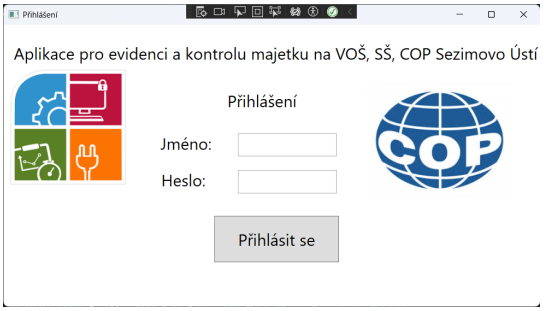
Kapitola 4

Tvorba aplikace

V této kapitole je popsána tvorba vlastní aplikace pro kontrolu a evidenci majetku. Práce byla tvořena ve vývojovém prostředí Visual Studio, pomocí programovacího jazyku C#. Toto prostředí bylo zvoleno pro svou jednoduchost. Programovací jazyk byl zvolen z důvodu jednoduchosti, dostatečného množství informací a návodů na internetu a také z důvodu předchozích zkušeností autora.

4.1 Tvorba aplikace

Tvorba aplikace začala návrhem uživatelského prostředí pro přihlášení do aplikace. Hotové uživatelské rozhraní zobrazuje obrázek 4.1.

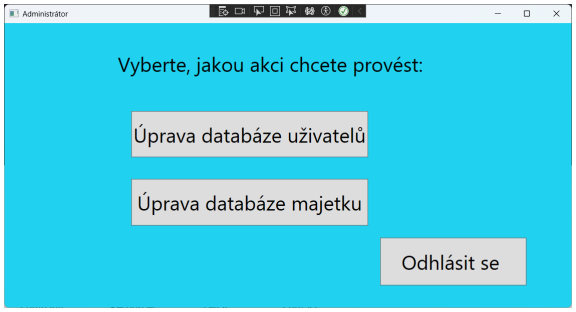


Obrázek 4.1: Výsledné uživatelské rozhraní pro přihlašování

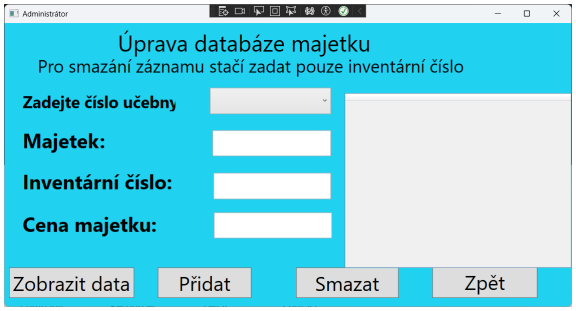
V tomto okně se nachází pět různých druhů komponentů. Obsahuje čtyři textová pole Label, jedno pole pro zadávání textu TextBox, jedno pole na heslo PasswordBox, dva boxy na obrázky Image a tlačítko Button pro přihlášení. Po vytvoření uživatelského

prostředí pro přihlášení se do aplikace přidala další dvě okna. Okno pro administrátora a okno pro běžného uživatele.

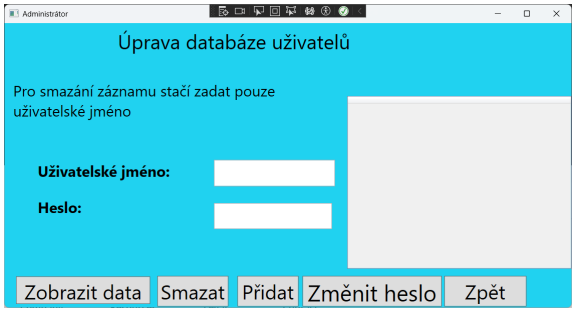
Okno pro administrátora je pojmenované Admin, okno pro běžného uživatele je pojmenované Uzivatel. Po přidání těchto oken se začalo tvořit jejich uživatelské prostředí. Uživatelské prostředí se tvořilo podobným způsobem jako u okna na přihlašování. Jediným rozdílem je, že se v těchto oknech mění zobrazení komponent.



Obrázek 4.2: Hlavní okno uživatelského prostředí pro administrátora



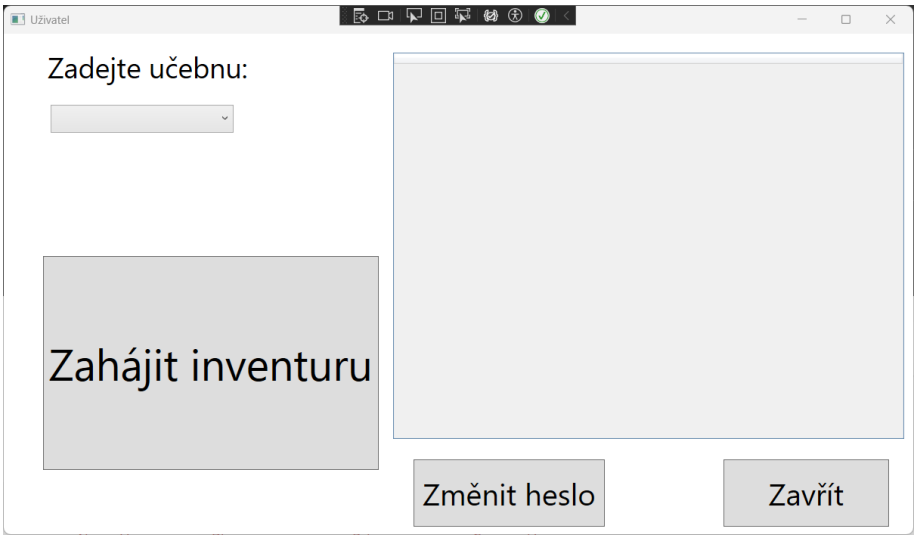
Obrázek 4.3: Okno pro úpravu databáze majetku



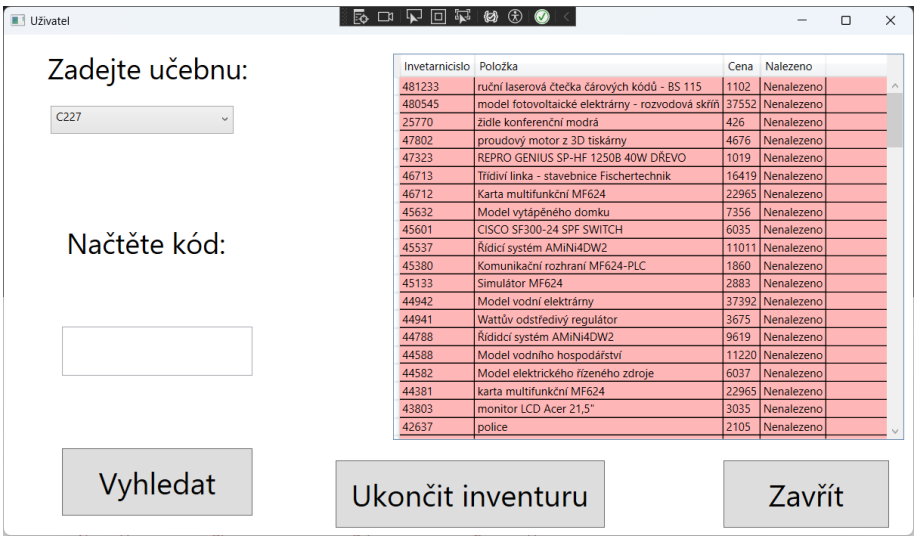
Obrázek 4.4: Okno pro úpravu databáze uživatelů

Uživatelské prostředí pro administrátora má tři různá zobrazení. Hlavní obrazovku, která je na obrázku 4.2. Tato obrazovka obsahuje dva typy komponent: tři tlačítka a Label. Obrazovku pro úpravu databáze majetku, která je na obrázku 4.3, tato obrazovka obsahuje šest Labelů, tři TextBoxy, jeden ListBox, čtyři tlačítka a jeden DataGrid pro zobrazení dat v databázi. A jako poslední obrazovku pro úpravu databáze uživatelů na obrázku 4.4. Ta obsahuje čtyři Labely, dva TextBoxy, pět tlačítek a jeden DataGrid. Po stisku tlačítka pro úpravu vybrané databáze se pomocí vlastnosti visibility u jednotlivých komponent zobrazí komponenty, které se využívají pro úpravu vybrané databáze, a naopak se skryjí nepotřebné komponenty. Tuto vlastnost nalezneme po kliknutí na vybraný komponent v oknu vlastností.

Uživatelské prostředí pro běžného uživatele, který bude vykonávat inventuru, má dvě obrazovky. Po přihlášení se zobrazí obrazovka pro volbu učebny a zahájení inventury v této učebně, která je zobrazena na obrázku 4.5, a poté okno pro uskutečnění inventury, to je ukázáno na obrázku 4.6. Přepínání mezi těmito obrazovkami funguje stejně jako u administrátora. V okně se skrývají a odkrývají komponenty.



Obrázek 4.5: Obrazovka pro zahájení inventury



Obrázek 4.6: Obrazovka pro probíhající inventuru

4.1.1 Tvorba databáze

Než se začala programovat aplikace, bylo nutné vytvořit databázi, ve které budou uloženy přihlašovací údaje a také tabulka majetku. Databáze se tvořila přímo v prostředí Visual Studio. Pro její funkčnost bylo nutné doinstalovat si balíček NuGet System.Data.SqlClient. Po nainstalování balíčku se v projektu otevře SQL Server Object Explorer.

Pomocí SQL Server Object Explorer byla vytvořena lokální databáze s názvem Inventura. V databázi byly následně vytvořeny dvě tabulky. První tabulka pro úpravu dat uživatelů. Tato tabulka se jmenuje Uzivatele. A druhá tabulka, která slouží pro úpravu dat soupisu inventury. Tato tabulka se jmenuje podle čísla učebny, pro kterou je určena. Každá učebna má tedy vlastní tabulku. Databáze je uložena v souboru Inventura.mdf, který je součástí projektu. Pro zajištění přenositelnosti aplikace je tento soubor přenášen společně s aplikací.

4.1.2 Programování aplikace

Po vytvoření databáze pro aplikaci se začalo s programováním. Jako první se programovalo okno pro přihlášení. Prvním krokem bylo vytvoření připojovacího řetězce, pomocí kterého se bude připojovat k právě vytvořené databázi.

```
        private string PripojovaciRetezec
{
   get
   {
      return @"Data Source=(LocalDB)\MSSQLLocalDB;
      Initial Catalog=Inventura;
      Integrated Security=True;
      Connect Timeout=30";
   }
}
```

Tento kód představuje vlastnost, která slouží pro připojení k databázi. Vlastnost slouží pouze pro čtení. Tato vlastnost vrací textový řetězec. Kód obsahuje důležité parametry. Parametr Data Source určuje, o jaký typ databáze se jedná. Initial Catalog určuje, jaký má databáze název. Integrated Security znamená, jaký způsob přístupu k databázi se používá, pokud je tento parametr nastaven na true, používá se Windows autentizace, pokud je false, používá se jméno a heslo. Connect Timeout určuje dobu, po kterou se aplikace pokouší připojit k databázi. Tato doba se udává v sekundách. Tento kód se nachází ve všech třech oknech aplikace. Dále se naprogramovala metoda pro přihlášení do aplikace.

```
   private void Prihlas()
   {
      string jmeno = TextBox_Jmeno.Text.Trim(); // načtení obsahu z
         TextBoxu s odstraněním mezer
      string heslo = PasswordBoxHeslo.Password.ToString().Trim(); //
         načtení obsahu z PasswordBoxu s odstraněním mezer
      string hashedPassword = HashPassword(heslo); // odeslání hesla do
         metody pro hashování

      if (string.IsNullOrEmpty(jmeno) || string.IsNullOrEmpty(heslo))
      {
         MessageBox.Show("Uživatelské jméno a heslo musí být vyplněny!");
```

```
        // ověření vyplnění údajů
    }
    else
    {
        SqlConnection pripojeni = new SqlConnection(PripojovaciRetezec);
            // Vytvoření připojení k databázi

        string prikazText = "SELECT COUNT(*) FROM Uzivatele WHERE Jmeno
            = @jmeno;"; // SQL příkaz pro databázi

        SqlCommand prik = new SqlCommand(prikazText, pripojeni);
        pripojeni.Open(); // vytvoření příkazu pro databázi
        prik.Parameters.AddWithValue("@jmeno", jmeno); // vložení
            parametru do příkazu
        int pocet = (int)prik.ExecuteScalar(); // Určení počtu záznamů
        pripojeni.Close(); // ukončení připojení

        if (pocet == 1) // ověření, že existuje záznam
        {
            string textPrikazu = "SELECT * FROM Uzivatele WHERE Jmeno =
                @jmeno;";

            SqlCommand prikaz = new SqlCommand(textPrikazu, pripojeni);
            pripojeni.Open();
            prikaz.Parameters.AddWithValue("@jmeno", jmeno);
            SqlDataReader zaznamy = prikaz.ExecuteReader(); // uložení
                výsledků z databáze
            while (zaznamy.Read() == true) // cyklus pro procházení všech
                záznamů
            {
```

```
            if (hashedPassword == zaznamy[2].ToString()) // ověření
                správnosti hesla
            {

               if (jmeno == "Admin") // zjištění, jestli se přihlašuje
                   administrátor
               {
                  Admin okno; // otevření okna pro administrátora
                  okno = new Admin();
                  okno.Show();
                  this.Close();
               }
               else
               {
                  Uzivatel okno; // otevření okna pro uživatele
                  okno = new Uzivatel();
                  okno.Show();
                  this.Close();
               }
               break;
            }
            else
            {
               MessageBox.Show("Špatné jméno nebo heslo!");
            }

         }

         pripojeni.Close();
      }
```

```
            else
            {
                MessageBox.Show("Zadané uživatelské jméno neexistuje!");
            }
        }
    }
```

Metoda Prihlas slouží pro přihlášení uživatele do aplikace. Pomocí kódu se nejprve načtou údaje zadané do políček na přihlašovací obrazovce. Zadané heslo se pomocí vlastnosti pro hashování hesel zašifruje. Následně se ověří, že uživatel zadal přihlašovací údaje. Když jsou zadány všechny údaje, vytvoří se připojení k databázi a příkaz, který chceme v databázi provést. Jako první se pomocí příkazu Count zjistí, zda databáze obsahuje záznam s daným přihlašovacím jménem. Každé uživatelské jméno je jedinečné. Po zjištění, že existuje záznam, se z databáze načte záznam se zadaným uživatelským jménem. Po načtení záznamů se pomocí cyklu projdou načtené záznamy a porovná se zadané heslo s heslem v databázi. Pokud je heslo správné, zjistí se, zda se přihlašuje administrátor, nebo běžný uživatel. Podle toho se otevře okno pro administrátora, nebo běžného uživatele. Když se heslo neshoduje, zobrazí se uživateli okno s chybovou hláškou. Pokud se nenajde záznam se zadaným jménem, zobrazí se chybová hláška.

```
    static string HashPassword(string password)
{
    using (SHA256 sha256Hash = SHA256.Create())
    {
        // Převod hesla na pole bytů
        byte[] bytes =
            sha256Hash.ComputeHash(Encoding.UTF8.GetBytes(password));

        // Převede hash na hexadecimální řetězec
        StringBuilder builder = new StringBuilder();
        foreach (byte b in bytes)
        {
            builder.Append(b.ToString("x2"));
        }
```

```
            return builder.ToString();
        }
    }
```

Metoda HashPassword ukazuje kód pro hashování hesla, které zadal uživatel. Heslo je nutné hashovat, aby ho bylo možné porovnat s heslem zadaným v databázi. Hashování se používá pro zabezpečení. Tato část kódu je využita ve všech oknech aplikace. Po naprogramování tohoto okna se programovala zbylá dvě okna. Jako první se programovalo okno pro administrátora.

Okno pro administrátora obsahuje stejný kód pro připojení k databázi jako hlavní okno aplikace. Okno dále obsahuje kódy, které se provedou po kliknutí na tlačítka, která jsou na obrazovce ihned po přihlášení. Po kliknutí na tlačítko pro úpravu databáze majetku se změní zobrazené komponenty. Ty se mění pomocí vlastnosti visibility.

```
private void ButtonUzSmazat_Click(object sender, RoutedEventArgs e)
{
    string jmeno = TextBoxUzJmeno.Text.Trim(); // načtení zadaného
        uživatelského jména bez mezer
    // na začátku a na konci
    if (string.IsNullOrEmpty(jmeno)) // zjištění, zda bylo zadáno jméno
    {
        MessageBox.Show("Uživatelské jméno musí být zadáno!"); //
            zobrazení chybové hlášky
    }
    else
    {
        SqlConnection pripojeni = new SqlConnection(PripojRetezec); //
            navázání spojení s databází
        string prikazText = "SELECT COUNT(*) FROM Uzivatele WHERE Jmeno
            = @jmeno;"; // příkaz pro databázi

        SqlCommand prik = new SqlCommand(prikazText, pripojeni);
        pripojeni.Open();
```

```
            prik.Parameters.AddWithValue("@jmeno", jmeno);
            int pocet = (int)prik.ExecuteScalar();
            pripojeni.Close();

            if (pocet == 1)
            {
               SqlConnection pripojen = new SqlConnection(PripojRetezec);

               string Prikaz = "DELETE FROM Uzivatele WHERE Jmeno=@jmeno;";
                  // příkaz pro vymazání záznamu z tabulky

               SqlCommand prikaz = new SqlCommand(Prikaz, pripojen);
               pripojen.Open();
               prikaz.Parameters.AddWithValue("@jmeno", jmeno);
               prikaz.ExecuteNonQuery(); // provedení příkazu v databázi

               pripojen.Close(); // ukončení připojení k databázi

               MessageBox.Show("Uživatel byl úspěšně smazán!"); // zobrazení
                  okna se zprávou

               Data(); // zavolání metody
            }
            else
            {
               MessageBox.Show("Zadané uživatelské jméno neexistuje!"); //
                  zobrazení chybové hlášky
            }
         }

      }
```

Metoda ButtonUzSmazat_Click ukazuje kód pro vymazání záznamu z tabulky uživatelů. Pro vymazání záznamu je nutné zadat uživatelské jméno. Na začátku kódu se načte zadané uživatelské jméno. Ověří se, zda bylo zadáno nějaké uživatelské jméno. Poté se ověří, že tabulka v databázi obsahuje záznam se zadaným uživatelským jménem. Pokud takový záznam existuje, provede se příkaz pro vymazání záznamu z tabulky. Následně se zavolá metoda Data, která zobrazí aktuální data v databázi v komponentu DataGrid. Pokud záznam se zadaným jménem neexistuje, zobrazí se chybová hláška.

```
        private void Data()
{
    SqlConnection pripojen = new SqlConnection(PripojRetezec); //
        navázání spojení s databází

    string textPrikazu = "SELECT * FROM Uzivatele;"; // definování
        příkazu pro databázi

    pripojen.Open(); // otevření připojení
    SqlDataAdapter adapter = new SqlDataAdapter(textPrikazu,
        pripojen); // vytvoření komunikace mezi aplikací a databází
    DataTable dataTable = new DataTable(); // vytvoření tabulky pro
        data z databáze
    adapter.Fill(dataTable); // naplnění tabulky daty
    adapter.Update(dataTable); // aktualizace změn
    pripojen.Close(); // ukončení připojení k databázi
    DataGrid.ItemsSource = dataTable.DefaultView; // zobrazení dat do
        komponentu DataGrid

}
```

Metoda Data načítá data z tabulky a ukládá je do proměnné dataTable, ze které se data načítají do komponentu DataGrid. Tento kód je podobný i pro zobrazení dat z tabulky pro data v tabulce, která obsahuje položky pro inventuru. Jediný rozdíl mezi těmito kódy je název tabulky v textu příkazu a jméno komponentu DataGrid. Také je rozdíl ve jméně metody. Metoda pro načtení inventárních dat se v této aplikaci jmenuje DataInv.

```
private void ButtonUzZmena_Click(object sender, RoutedEventArgs e)
{
   string jmeno = TextBoxUzJmeno.Text; // načtení zadaného jména
   string heslo = TextBoxHeslo.Text; // načtení hesla
   string hashedPassword = HashPassword(heslo); // hashování hesla
      pomocí vlastnosti HashPassword

   if (String.IsNullOrEmpty(jmeno) || String.IsNullOrEmpty(heslo)) //
      ověření, že je zadané jméno a heslo
   {
      MessageBox.Show("Jméno a heslo musí být vyplněny!"); // zobrazení
         okna s hláškou
   }
   else
   {

      SqlConnection pripojeni = new SqlConnection(PripojRetezec); //
         navázání spojení s databází
      string prikazText = "SELECT COUNT(*) FROM Uzivatele WHERE Jmeno =
         @jmeno;"; // definování příkazu pro databázi

      SqlCommand prik = new SqlCommand(prikazText, pripojeni); //
         odeslání příkazu databázi
      pripojeni.Open(); // otevření spojení s databází
      prik.Parameters.AddWithValue("@jmeno", jmeno); // vložení zadaného
         jména do příkazu
      int pocet = (int)prik.ExecuteScalar(); // provedení příkazu v
         databázi, uložení počtu záznamů do proměnné
      pripojeni.Close(); // zavření připojení

      if (pocet == 1) // ověření, že existuje záznam
```

```
        {
            SqlConnection pripojen = new SqlConnection(PripojRetezec); //
                navázání spojení s databází

            string Prikaz = "UPDATE Uzivatele SET Heslo = @heslo WHERE Jmeno
                = @jmeno;"; // definování příkazu pro databázi

            SqlCommand prikaz = new SqlCommand(Prikaz, pripojen); //
                odeslání příkazu databázi
            pripojen.Open(); // otevření spojení s databází
            prikaz.Parameters.AddWithValue("@jmeno", jmeno); // vložení
                zadaného jmnéna do příkazu
            prikaz.Parameters.AddWithValue("@heslo", hashedPassword); //
                vložení hashovaného hesla do příkazu
            prikaz.ExecuteNonQuery(); // provedení příkazu v databázi
            pripojen.Close(); // ukončení připojení k databázi

            MessageBox.Show("Heslo bylo úspěšně změněno!"); // zobrazení
                okna s hláškou

            Data(); // zavolání metody Data
        }
        else
        {
            MessageBox.Show("Zadané uživatelské jméno neexistuje"); //
                zobrazení okna s hláškou
        }
    }
}
```

Kód metody ButtonUzZmena_Click se provede po stisknutí tlačítka pro změnu hesla uživatele. Kód načte zadané uživatelské jméno a heslo zadané administrátorem. Načtené

heslo odešle do metody HashPassword, kde se zadané heslo hashuje z důvodu bezpečnosti. Ověří se, že bylo zadané uživatelské jméno i heslo. Pokud ne, zobrazí se okno s hláškou. Pokud bylo zadáno oboje, ověří se, že existuje záznam se zadaným uživatelským jménem. Po ověření, že záznam existuje, se provede záznam pro úpravu záznamu v tabulce. Nakonec se zaktualizuje obsah DataGridu v metodě Data. V okně pro běžného uživatele najdeme stejný kód pro změnu hesla. Jediným rozdílem je, že v okně pro uživatele se nevolá metoda Data.

```
    private void ButtonUzPridat_Click(object sender, RoutedEventArgs e)
{
   string jmeno = TextBoxUzJmeno.Text.Trim();
   string heslo = TextBoxHeslo.Text.Trim();
   string hashedPassword = HashPassword(heslo);
   if (String.IsNullOrEmpty(jmeno) || String.IsNullOrEmpty(heslo))
   {
      MessageBox.Show("Jméno nebo heslo musí být vyplněny");
   }
   else
   {
      SqlConnection pripojen = new SqlConnection(PripojRetezec);

      string PrikazJmeno = "SELECT Jmeno FROM Uzivatele WHERE
          Jmeno=@jmeno;";

      SqlCommand prikazJmeno = new SqlCommand(PrikazJmeno, pripojen);
      pripojen.Open();
      prikazJmeno.Parameters.AddWithValue("@jmeno", jmeno);
      SqlDataReader zaznamy = prikazJmeno.ExecuteReader();
      while (zaznamy.Read() == true)
      {

         if (jmeno == zaznamy[0].ToString())
         {
```

```
            MessageBox.Show("Zadané uživatelské jméno již existuje!");
        }
        break;
    }
    pripojen.Close();

    pripojen.Open();
    string Prikaz = "INSERT INTO Uzivatele (Jmeno, Heslo) VALUES
        (@jmeno,@heslo);";

    SqlCommand prikaz = new SqlCommand(Prikaz, pripojen);

    prikaz.Parameters.AddWithValue("@jmeno", jmeno);
    prikaz.Parameters.AddWithValue("@heslo", hashedPassword);
    prikaz.ExecuteNonQuery();

    MessageBox.Show("Uživatel byl úspěšně přidán!");
    Data();
    pripojen.Close();
  }
}
```

Obslužná metoda pro tlačítko ButtonUzPridat_Click je naprogramována pro přidání záznamu do tabulky. Na začátku se načte zadané jméno a heslo. Heslo se odešle do metody HashPassword, kde se heslo hashuje. Ověří se, že bylo zadáno uživatelské jméno i heslo. Pokud ne, zobrazí se hláška. Pokud je oboje zadáno, aplikace se připojí k databázi a provede se příkaz pro přidání záznamu do databáze. Po provedení příkazu v databázi se zobrazí okno s hláškou, že byl uživatel úspěšně přidán. Následně se v metodě Data nahraje do DataGridu nový obsah tabulky.

```
  private void ButtonPridat_Click(object sender, RoutedEventArgs e)
  {
    string majetek = TextBoxMajetek.Text.Trim(); // načtení názvu
```

```
    majetku
string invetar = TextBoxInv.Text.Trim(); // načtení inventárního
    čísla
string ucebna = ComboBox.Text.Trim(); // načtení čísla učebny
string cena = TextBoxCena.Text.Trim(); // načtení ceny majetku
int InventarniCislo; // vytvoření číselné proměnné pro inventární
    číslo
int CenaMajetek; // vytvoření číselné proměnné pro cenu
bool cenaUspech = int.TryParse(cena, out CenaMajetek); //
    převedení načtené ceny do číselné proměnné
bool uspech = int.TryParse(invetar, out InventarniCislo); //
    převedení načteného inventárního čísla do číselné proměnné
if (string.IsNullOrEmpty(majetek) || string.IsNullOrEmpty(invetar)
|| string.IsNullOrEmpty(ucebna) || string.IsNullOrEmpty(cena)) //
    ověření, že jsou všechna pole vyplněna
{
   MessageBox.Show("Všechna pole musí být vyplněna!"); // zobrazení
       hlášky
}
else
{
   if (uspech == true && cenaUspech== true) // ověření, že se
       povedlo převedení na číselné proměnné
   {
      SqlConnection pripojen = new SqlConnection(PripojRetezec);

      string Prikaz = "INSERT INTO " + ucebna + " (Invetarni_cislo,
          Položka, Cena) VALUES (@invertar,@polozka, @cena);"; //
          příkaz pro vložení do databáze

      SqlCommand prikaz = new SqlCommand(Prikaz, pripojen);
```

```
            pripojen.Open();
            prikaz.Parameters.AddWithValue("@invertar", InventarniCislo);
            prikaz.Parameters.AddWithValue("@polozka", majetek);
            prikaz.Parameters.AddWithValue("@cena", CenaMajetek);
            prikaz.ExecuteNonQuery();
            pripojen.Close();
            MessageBox.Show("Položka byla úspěšně přidána!");
        }
        else
        {
            MessageBox.Show("Inventární číslo nebo cena nesmí obsahovat
                písmena!");
        }

        DataInv(); // zavolání vlastnosti

    }
}
```

Metoda ButtonPridat_Click je pro tlačítko pro přidání záznamu do tabulky se soupisem majetku. Po stisku tlačítka se načtou zadané informace ze všech políček. Po načtení zadaných dat z políčka pro cenu majetku a jeho inventárního čísla je nutné tyto údaje převést z datového typu string, který je pro text, na typ int, který je pro čísla. Převedení se provede pomocí metody TryParse, která zjistí, zda zadané číslo neobsahuje nějaké písmeno. Kód zjistí, jestli byla vyplněna všechna pole. Pokud není nějaké z polí vyplněno, zobrazí se okno s hláškou. Jinak se ověří, že ani jedno ze zadaných čísel neobsahuje písmena. Pokud písmena neobsahují, provede se kód pro přidání záznamu do tabulky, jinak se zobrazí okno s hláškou. Nakonec se zavolá metoda DataInv, která zaktualizuje data v DataGridu.

```
private void ButtonSmazat_Click(object sender, RoutedEventArgs e)
{
    string majetek = TextBoxMajetek.Text.Trim();
```

```
string inventar = TextBoxInv.Text.Trim();
string ucebna = ComboBox.Text.Trim();
int InventarniCislo;
bool uspech = int.TryParse(inventar, out InventarniCislo);
if (string.IsNullOrEmpty(inventar) || string.IsNullOrEmpty(ucebna))
{
   MessageBox.Show("Číslo učebny musí být vyplněno!");
}
else
{
   if (uspech == true)
   {
      SqlConnection pripojen = new SqlConnection(PripojRetezec);

      string Prikaz = "DELETE FROM " + ucebna + " WHERE
         Invetarni_cislo=@inventar;";

      SqlCommand prikaz = new SqlCommand(Prikaz, pripojen);
      pripojen.Open();
      prikaz.Parameters.AddWithValue("@inventar", InventarniCislo);
      prikaz.ExecuteNonQuery();

      pripojen.Close();
      MessageBox.Show("Položka byla úspěšně smazána!");
   }
   else
   {
      MessageBox.Show("Inventární číslo nesmí obsahovat písmena!");
   }
   DataInv();
}
```

```
}
```

Metoda ButtonSmazat_Click ukazuje, co má nastat po stisku tlačítka pro vymazání záznamu z tabulky se soupisem inventury. Na začátku se načtou potřebné údaje. Pro vymazání záznamu je nutné zadání učebny a inventárního čísla. Inventární číslo je opět nutné převést z datového typu string na int. To je provedeno stejným způsobem jako u předchozí části kódu. Po ověření, že je zadáno vše potřebné a číslo bylo úspěšně převedeno, se provede kód pro odstranění záznamu z tabulky.

```
        private void ButtonOdhlasit_Click(object sender,
            RoutedEventArgs e)
{
    MainWindow okno;
    okno = new MainWindow();
    okno.Show();
    this.Close();
}
```

Kód metody ButtonOdhlasit_Click je pro tlačítko na odhlášení z okna pro administrátora. Po stisknutí tohoto tlačítka se zavře okno pro administrátora a znovu se otevře okno pro přihlášení do aplikace. Po naprogramování okna pro administrátora se programovalo poslední okno, které je pro běžného uživatele, který bude provádět inventuru.

Okno pro běžného uživatele obsahuje stejný kód pro připojení k databázi jako hlavní okno a okno pro administrátora. Dále obsahuje stejnou metodu pro hashování hesel jako ostatní okna. V programu se dále nachází kódy pro akce, které se provádí po stisku tlačítek, která okno obsahuje.

```
private void ButtonZahaj_Click(object sender, RoutedEventArgs e)
{
    string ucebna = ComboBoxCislo.Text.Trim().ToUpper(); // načtení
        čísla učebny s odstraněním
    // mezer a zvětšení písmen

    List<int> inventarnicisla = new List<int>(); // vytvoření kolekce
```

```
        pro čísla
    if (string.IsNullOrEmpty(ucebna)) // ověření, že byla zadána učebna
    {
        MessageBox.Show("Učebna musí být zadána!"); // zobrazení hlášky
    }
    else
    {
        SqlConnection pripojeni = new SqlConnection(PripojRetezec); //
            vytvoření spojení s databází
        string textPrikazu = "SELECT Invetarni_cislo FROM " + ucebna; //
            definování příkazu

        SqlCommand prikaz = new SqlCommand(textPrikazu, pripojeni); //
            odeslání příkazu databázi
        pripojeni.Open(); // otevření připojení
        SqlDataReader zaznamy = prikaz.ExecuteReader(); // provedení
            příkazu v databázi

        while(zaznamy.Read() == true) // cyklus pro přečtení všech
            záznamů
        {
            inventarnicisla.Add(Convert.ToInt32(zaznamy[0])); // uložení
                záznamu do seznamu
        }
        pripojeni.Close(); // ukončení připojení
        foreach(int i in inventarnicisla) // cyklus pro projetí záznamů
            záznam po záznamu
        {
            string Prikaz = "UPDATE " + ucebna + " SET Nalezeno =
                @nalezeno WHERE Invetarni_cislo = @cislo;"; // definování
                příkazu
```

```
            SqlCommand command = new SqlCommand(Prikaz, pripojeni); //
               odeslání příkazu databázi
            pripojeni.Open(); // otevření připojení k databázi
            command.Parameters.AddWithValue("@nalezeno", "Nenalezeno");
               // zadání parametru do příkazu
            command.Parameters.AddWithValue("@cislo", i); // zadání
               parametru do příkazu
            command.ExecuteNonQuery(); // provedení příkazu v databázi
            pripojeni.Close(); // ukončení připojení
        }

        DataInventura(); // zavolání vlastnosti
        ButtonZahaj.Visibility = Visibility.Hidden; // změna vlastnosti
           visibility u komponentu
        LabelKod.Visibility = Visibility.Visible;
        textBoxKod.Visibility = Visibility.Visible;
        ButtonHledej.Visibility = Visibility.Visible;
        ButtonZmenaHesla.Visibility = Visibility.Hidden;
        ButtonUkoncit.Visibility = Visibility.Visible;

    }
}
```

Metoda ButtonZahaj_Click se provede po stisku tlačítka pro zahájení inventury. Po stisku tlačítka se načte zadané číslo učebny. Vytvoří se kolekce čísel pro uložení všech inventárních čísel z tabulky. Ověří se, že bylo uživatelem zadáno číslo učebny. Pokud ne, zobrazí se chybová hláška. Pokud ano, provede se kód pro upravení dat v tabulce tak, aby bylo u všech položek inventury napsáno „nenalezeno“. Dále se zavolá metoda, která naplní DataGrid daty z tabulky. Nakonec se změní vlastnost visibility u požadovaných komponent. Po zahájení inventury se začnou vyhledávat položky podle čísel. Vyhledávání probíhá v metodě Hledej. Tuto metodu poté volá políčko pro zadání čísla, když se stiskne klávesa Enter, nebo když se stiskne tlačítko pro vyhledávání.

```
    private void Hledej(string cislo)
{
    string ucebna = ComboBoxCislo.Text.Trim().ToUpper(); // načtení
       zadané učebny
    int InvCislo; // vytvoření číselné proměnné
    bool uspech = int.TryParse(cislo, out InvCislo); // převedení
       string na int
    List<int> invetraniCisla = new List<int>(); // vytvoření kolekce
       čísel
    SqlConnection pripojeni = new SqlConnection(PripojRetezec); //
       vytvoření připojení k databázi

    string textPrikazu = "SELECT Invetarni_cislo FROM " + ucebna; //
       definování příkazu pro databázi

    SqlCommand prikaz = new SqlCommand(textPrikazu, pripojeni); //
       odeslání příkazu databázi
    pripojeni.Open(); // otevření připojení
    SqlDataReader zaznamy = prikaz.ExecuteReader(); // uložení záznamů
       do proměnné

    if (string.IsNullOrEmpty(cislo)) // ověření vyplnění čísla
        MessageBox.Show("Inventrání číslo musí být vyplněno!"); //
           zobrazení okna s hláškou
    else
    {
        if (uspech == true) // ověření, že se povedlo převést string na
           int
        {
            while (zaznamy.Read() == true) // cyklus pro projetí záznamů
            {
```

```
            invetraniCisla.Add(Convert.ToInt32(zaznamy[0])); //
                uložení inventárních čísel do kolekce
        }
        pripojeni.Close(); // ukončení připojení

        if (invetraniCisla.Contains(InvCislo)) // zjištění, zda
            záznamy z tabulky
                                                // obsahují načtené
                                                    inventární číslo
        {
            string Prikaz = "UPDATE " + ucebna + " SET Nalezeno =
                @nalezeno WHERE Invetarni_cislo = @cislo;";
            // definování příkazu pro databázi
            SqlCommand command = new SqlCommand(Prikaz, pripojeni);
                // odeslání příkazu databázi
            pripojeni.Open(); // otevření připojení
            command.Parameters.AddWithValue("@nalezeno", "Nalezeno");
                // vložení parametru do příkazu
            command.Parameters.AddWithValue("@cislo", InvCislo); //
                vložení parametru do příkazu
            command.ExecuteNonQuery(); // provedení příkazu v databázi
            pripojeni.Close(); // ukončení připojení
        }
        else
        {
            string Prikaz = "INSERT INTO " + ucebna +
                "(Invetarni_cislo, Nalezeno) VALUES
                (@Icislo,@Nalez);"; // definování příkazu pro databázi

            SqlCommand command = new SqlCommand(Prikaz, pripojeni);
                // odeslání příkazu do databáze
```

```
                pripojeni.Open(); // otevření připojení
                command.Parameters.AddWithValue("@Nalez", "Přebývá"); //
                    vložení parametru do příkazu
                command.Parameters.AddWithValue("@Icislo", InvCislo); //
                    vložení parametru do příkazu
                command.ExecuteNonQuery(); // provedení příkazu
                pripojeni.Close(); // ukončení připojení
            }

        }
        else
        {
            MessageBox.Show("Inventární číslo nesmí obsahovat
                písmena!"); // zobrazení okna s hláškou
        }
        textBoxKod.Text = ""; // vymazání čísla z políčka pro zadání
        DataInventura(); // zavolání metody
    }
}


        private void ButtonHledej_Click(object sender, RoutedEventArgs e)
       {
           string[] cisla = textBoxKod.Text.Split(new[] { '\n', '\r',
               ',', ';' },
               StringSplitOptions.RemoveEmptyEntries); // rozdělení
                   zadaných čísel
           foreach (string c in cisla) // projetí každého zadaného kódu
           {
               Hledej(c.Trim()); // zavolání metody pro každý kód
           }
```

```
            textBoxKod.Text = ""; // vymazání textu z políčka
                    }
```

Metoda Hledej se volá pro vyhledání zadaných inventárních čísel v databázi. Po načtení čísel se v případě zadání více čísel provede kód pro oddělení zadaných čísel. Zadaná čísla se uloží do kolekce čísel. Pokud tabulka obsahuje zadané číslo, změní se obsah v sloupečku Nalezeno na „nalezeno“. Pokud tabulka zadané číslo neobsahuje, změní se obsah políčka sloupce Nalezeno na „přebývá“", protože daná položka nepatří do zvolené učebny. Celý řádek je zbarvený červeně, když je položka nenalezena, když je položka nalezena, je celý řádek podbarvený zeleně. Když je ve sloupečku Nalezeno „přebývá“, je celý řádek podbarvený oranžově. Metoda ButtonHledej_Click si načte všechny kódy, které byly uživetelem načteny pomocí čtečky čárových kódů. Kódy jsou v metodě od sebe odděleny a po jednom odesílány do metody Hledej. Po načtení všech inventárních čísel pomocí čtečky čárových kódů se ukončí inventura a její výsledek se uloží do excelovské tabulky. Pro ukončení inventury se použije následující kód metody ButtonUkoncit_Click.

```
 private void ButtonUkoncit_Click(object sender, RoutedEventArgs e)
{
   string ucebn = ComboBoxCislo.Text.Trim().ToUpper(); // načtení zadané
       učebny
   if (string.IsNullOrEmpty(ucebn)) // ověření, že byla zadána učebna
   {
      MessageBox.Show("Učebna musí být zadána!"); // zobrazení okna s
          hláškou
   }
   else
   {
      DataTable dataTable = new DataTable(); // vytvoření datové tabulky
      using (SqlConnection conn = new SqlConnection(PripojRetezec)) //
          připojovací řetězec
      {
         conn.Open();
         string query = "SELECT * FROM " + ucebn;
         using (SqlDataAdapter adapter = new SqlDataAdapter(query, conn))
```

```
        {
            adapter.Fill(dataTable); // načtení dat z databáze do
                DataTable
        }
    }
    // Vytvoření nového sešitu
    Spire.Xls.Workbook workbook = new Spire.Xls.Workbook();
    // Vymazání výchozích listů a vytvoření nového
    workbook.Worksheets.Clear();
    Spire.Xls.Worksheet sheet = workbook.Worksheets.Add("Inventura");

    // Vložení dat od řádku 1, sloupce 1, včetně hlaviček sloupců
    sheet.InsertDataTable(dataTable, true, 1, 1);

// Zápis dat do Excelu včetně názvů sloupců od řádku 1, sloupce 1
    sheet.InsertDataTable(dataTable, true, 1, 1);

    // Zvýraznění hlavičky (tučné písmo + barva pozadí)
    sheet.Rows[0].Style.Font.IsBold = true;
    sheet.Rows[0].Style.Font.Size = 14;
    sheet.Rows[0].Style.HorizontalAlignment =
        HorizontalAlignType.Center;
    sheet.Rows[0].Style.Color = System.Drawing.Color.LightGray;

    // Formátování datových řádků
    for (int i = 1; i < sheet.Rows.Count(); i++)
    {
        CellRange dataRow = sheet.Rows[i];
        dataRow.Style.Font.Size = 12;
        dataRow.Style.HorizontalAlignment = HorizontalAlignType.Left;
    }
```

```
// Nastavení názvu písma
sheet.AllocatedRange.Style.Font.FontName = "Arial";

// Nastavení ohraničení
sheet.AllocatedRange.BorderAround(LineStyleType.Thin,
   System.Drawing.Color.Black);
sheet.AllocatedRange.BorderInside(LineStyleType.Medium,
   System.Drawing.Color.Black);

// Automatické přizpůsobení šířky sloupců
sheet.AllocatedRange.AutoFitColumns();

SaveFileDialog dialog = new SaveFileDialog(); // Inicializace
   dialogového okna pro výběr
                                          //umístění a názvu
                                             ukládaného souboru
dialog.Filter = "Excel|*.xlsx"; // definování přípony souboru,
   který se bude ukládat
if (dialog.ShowDialog() == true) // ověření, že se zobrazilo okno
   pro uložení
{
   if (File.Exists(dialog.FileName)) // ověření, jestli existuje
      záznam se zadaným jménem
   {
      File.Delete(dialog.FileName); // smazání souboru se zvoleným
         jménem
   }
   else
   {
      workbook.SaveToFile(dialog.FileName); // uložení tabulky do
```

```
                souboru
            }
            workbook.SaveToFile(dialog.FileName); // uložení tabulky do
                souboru
        }
        MessageBox.Show("Inventura byla ukončena");
        ButtonUkoncit.Visibility = Visibility.Hidden;
        textBoxKod.Visibility = Visibility.Hidden;
        ButtonZahaj.Visibility = Visibility.Visible;
        ButtonHledej.Visibility = Visibility.Hidden;
        LabelKod.Visibility = Visibility.Hidden;
        DataGridInv.ItemsSource = null;
        ComboBoxCislo.Text = "";

    }
}
```

Pro export dat právě ukončené inventury z databáze do Excelu bylo nutné do projektu přidat balíček NuGet. Název tohoto balíčku je Spire.XLS. Po stisku tlačítka pro ukončení inventury se ověří, že je zadané číslo učebny. Poté se načtou data z tabulky a nahrají se do proměnné pro tabulku. Následně se vytvoří nový sešit v Excelu. Poté se definuje vzhled vytvářené tabulky. Po definování tabulky se inicializuje dialog pro uložení souboru, ve kterém se volí umístění a název ukládaného souboru. Pokud existuje záznam se zvoleným jménem, tento soubor se vymaže a následně vytvoří nový. Nakonec se zobrazí okno s hláškou a znovu se zobrazí komponenty pro zahájení inventury.

Pro zobrazení tabulky načtené z databáze se v aplikaci využívá prvek DataGrid. Pro změnu barvy pozadí v DataGridu je nutné upravit XAML kód tohoto komponentu následujícím způsobem:

```
<DataGrid x:Name="DataGridInv" Grid.Column="2"
    d:ItemsSource="{d:SampleData ItemCount=5}" Margin="91,20,10,101"
    Grid.RowSpan="4" Grid.ColumnSpan="2">
<DataGrid.RowStyle>
```

```
<Style TargetType="DataGridRow">
<Style.Triggers>
<DataTrigger Binding="{Binding [Nalezeno]}" Value="Nalezeno">
<Setter Property="Background" Value="LightGreen"/>
</DataTrigger>
<DataTrigger Binding="{Binding [Nalezeno]}" Value="Nenalezeno">
<Setter Property="Background" Value="#FFB6B6"/>
</DataTrigger>
<DataTrigger Binding="{Binding [Nalezeno]}" Value="Přebývá">
<Setter Property="Background" Value="#FFF3B680"/>
</DataTrigger>
</Style.Triggers>
</Style>
</DataGrid.RowStyle>
</DataGrid>
```

Tento kód definuje komponent DataGrid, který se používá pro zobrazení dat z tabulky v databázi. Pro změnu pozadí řádků se využívají DataTrigerry, které mění barvu pozadí na základě hodnoty v sloupci Nalezeno.

4.2 Nasazení aplikace

Pro spuštění aplikace na jiném PC se přenese celá složka s aplikací. Aplikace se spouští pomocí souboru Inventura.exe, který se nachází ve zkopírované složce. Pro provedení inventury se po spuštění aplikace, přihlásí se uživatel, vybere si číslo učebny, pro kterou chce inventuru udělat, a tuto inventuru zahájí. Po záhájení inventury obejde celou vybranou učebnu se čtečkou čárových kódů. Po načtení všech kódů je vyhledá v databázi. Po vyhledání ukončí inventuru a její výsledek si uloží do excelovského souboru. Tento proces bude oproti současnému stavu inventury na škole ulehčením a urychlením práce pro učitele, kteří inventuru provádějí. V současnosti se majetek školy eviduje pomocí softwarové aplikace, ze které se vytiskne seznam majetku každé učebny. Majetek je označený svým unikátním číslem. Toto číslo je na majetku napsáno pomocí lihové fixy. Často je

majetek takto označen na špatně přístupném místě. Inventura probíhá ve dvou vlnách. Napřed probíhá takzvaná předinventura. Předinventura probíhá tak, že se vytisknou seznamy majetku pro jednotlivé učebny. Tyto seznamy se rozdělí mezi učitele, kteří inventuru provádějí. Učitelé poté s těmito seznamy procházejí učebny, které jim byly přiřazeny, a hledají příslušná inventární čísla na majetku. Tato práce je velmi zdlouhavá a ztížená tím, že jsou inventární čísla většinou na velmi špatně viditelných a přístupných místech. Poté následuje hlavní inventura, která probíhá stejným způsobem jako předinventura. Tento způsob inventury je velmi zdlouhavý, složitý a spotřebuje se při něm zbytečně velké množství papíru. Je tedy velmi neekologický.

4.3 Testování aplikace

Z důvodu přechodu školy na nový způsob inventury nebyly vylepeny čárové kódy v učebně aplikované informatiky. Aplikace byla z tohoto důvodu zkoušena pouze pomocí kódů, které byly vygenerovány na základě poskytnutých invetárních čísel. Tyto kódy byly následně vytištěny. Vytištěné kódy byly pomocí čtečky načítány přímo z papírů do aplikace. Byla ověřena funkčnost aplikace. Aplikace umožňuje také využití čtečky, která dokáže načíst více kódů najednou. Nebo je možné načíst všechny kódy a poté všechny kódy vyhledat v databázi. Během testování bylo zjištěno, že aplikace správně zpracovává načtené kódy a umožňuje jejich následné vyhledání v databázi pomocí připojení aplikace do databáze a následného použití příkazů, které se využívají pro práci s databází. Nebyly zjištěny žádné závažné chyby, které by bránily jejímu používání. Testování potvrdilo, že aplikace je schopna pracovat jak s jednotlivě načtenými kódy, tak s více kódy načtenými současně. Aplikace kódy, které byly načteny najednou, rozdělí na jednotlivé kódy, které uloží do proměnné a jeden po druhém projde v databázi. Tyto kódy jsou nejčastěji oddělené mezerníkem.

Kapitola 5

Závěr

Cílem této absolventské práce bylo navrhnout aplikaci pro evidenci a kontrolu majetku s využitím čtečky čárových kódů. Tento cíl práce byl splněn. V rámci práce byla vytvořena plně funkční aplikace, která umožňuje evidovat majetek a také ho kontrolovat.

V rámci práce byla jako první zpracovaná teoretická část, která se zabývá problematikou inventarizace, čárových kódu a jejich čteček. Dále byly řešeny různé možnosti nalepení čárových kódů na různé materiály, aby bylo zjištěno, jaký způsob bude pro daný materiál nejlepší.

Vlastní aplikace pro evidenci a kontrolu majetku byla tvořena v programovacím jazyce C# s využitím technologie WPF. Tato technologie umožňuje vytvoření moderního uživatelského prostředí. Data, se kterými aplikace pracuje, se zde zobrazují pomocí komponenty DataGrid. To zajišťuje, že jsou data přehledná pro uživatele, který s nimi pracuje. Pro velmi snadnou orientaci v probíhající inventuře byla využita podmíněná logika. Ta umožňuje rychlé rozpoznání chybějícího majetku. Aplikace také umožňuje export dat z provedené inventury do tabulky v Excelu pro jejich archivování. Pomocí aplikace lze také spravovat databázi uživatelů, kteří mají do aplikace přístup. To je umožněno pomocí administrátorského přístupu. Administrátor také může spravovat veškerý majetek.

Aplikace byla také otestována, aby byla ověřena její plná funkčnost. Toto testování potvrdilo, že aplikace správně zpracovává načtené kódy. Aplikace umožňuje načtení více kódů najednou a jejich následné ověření v databázi. Testování také potvrdilo správnou funkčnost části programu, která je určena pro administrátora.

Přínosem této práce je zjednodušení a zrychlení procesu inventury majetku. Také snížení možných chyb, které můžou při inventuře nastat. Aplikace celkově zefektivňuje celý proces inventury ve škole.

Do budoucnosti lze aplikaci rozšířit o napojení na centrální databázi, která se využívá

ve škole. Také by se daly rozšířit funkce aplikace. To by zajistilo její větší použitelnost a univerzálnost pro využití v reálném provozu instituce.

Literatura

Benadíková, A., Mada, Š. a Weinlich, S. (1994), *Čárové kódy: automatická identifikace*, Grada. ISBN 80-85623-66-8.

Daně pro lidi (2024), Inventarizace [online]. [cit. 2026-03-12], ⟨https://www.daneprolidi.cz/clanek/3-inventarizace.htm⟩.

Máče, M. (2013), *Účetnictví a finanční řízení*, Grada. ISBN 978-80-247-8385-7.

SMART-TEC (2024), Čárové kódy [online]. [cit. 2026-01-23], ⟨https://www.smart-tec.com/cs/know-how/znalost-auto-id/carove-kody⟩.

Tůma, V. (2009), Čárové kódy, (Diplomová práce), VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ, Brno.

Příloha A

Obsah přiloženého CD/DVD

K této práci je přiloženo CD/DVD s následující adresářovou strukturou.

- `Absolventská práce v LaTeX2e`
- `Složka s aplikací`
- `Soubor s databází`
- `Skripty pro vytvoření databáze`
- `Horakova_AP_2025_2026.pdf` – absolventská práce ve formátu PDF

Příloha B

Použitý software

LaTeX 2$_\varepsilon$ ⟨http://www.miktex.org/⟩

Visual Studio ⟨https://visualstudio.microsoft.com/⟩

WinEdt 11.2 ⟨http://www.winedt.com/⟩

Software z výše uvedeného seznamu je buď volně dostupný, nebo jeho licenci toho času vlastní Vyšší odborná škola, Střední škola, Centrum odborné přípravy, Sezimovo Ústí, Budějovická 421, kde autor toho času studoval a vytvořil tuto práci.

Příloha C

Časový plán absolventské práce

Činnost	**Časová náročnost**	**Termín ukončení**	**Splněno**
tvorba databáze předmětů v laboratoři aplikované informatiky	2 týdny	15.09.2025	listopad 2025
tvorba čárových kódů	2 týdny	1.10.2025	prosinec 2025
tvorba aplikace pro kontrolu majetku	2 měsíce	4.12.2025	únor 2026
testování aplikace	3 týdny	10.01.2026	únor 2026
umisťování čárových kódů	1 měsíc	1.02.2026	—
AP: kapitola Úvod	2 týdny	16.02.2026	duben 2026
AP: kompletní text	6 měsíců	30.04.2026	květen 2026