Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁRSKA PRÁCA

Systém evidencie dochádzky študentov s použitím RFID



2024

Ján Hyben

Vedúci práce:
Mgr. Radek Janoštík, Ph.D.

Študijný program: Informační technologie,
prezenční forma

Bibliografické údaje

Autor:	Ján Hyben
Názov práce:	Systém evidencie dochádzky študentov s použitím RFID
Typ práce:	bakalárska práca
Pracovisko:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2024
Študijný program:	Informační technologie, prezenční forma
Vedúci práce:	Mgr. Radek Janoštík, Ph.D.
Počet strán:	42
Prílohy:	elektronické dáta v online úložisku katedry; RFID USB čítačka
Jazyk práce:	slovenský

Bibliographic info

Author:	Ján Hyben
Title:	Student attendance recording system using RFID
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2024
Study program:	Information Technologies, full-time form
Supervisor:	Mgr. Radek Janoštík, Ph.D.
Page count:	42
Supplements:	electronic data in the department's online repository; RFID USB reader
Thesis language:	Slovak

Anotácia

Práca sa zaoberá návrhom a implementáciou okennej aplikácie pre systém Windows v jazyku C#, ktorá slúži na zaznamenávanie dochádzky študentov na hodiny. Využitie technológie RFID umožňuje vyučujúcim jednoducho zaznamenávať dochádzku študentov pomocou ich RFID identifikačných kariet. Súčasťou práce je aj správne rozpoznávanie a spracovanie údajov z RFID čítačky, komunikácia s identifikačnými kartami, efektívne spravovanie databázy a podpora súborov formátu CSV. Aplikácia vyučujúcim poskytuje prehľady dochádzky. Výsledkom je plne funkčná okenná aplikácia na správu a evidenciu dochádzky študentov pomocou technológie RFID.

Synopsis

The thesis deals with the design and implementation of a window application for Windows in C# language, which is used to record student attendance. The use of RFID technology allows teachers to easily record students' attendance using their RFID identification cards. The work also includes proper recognition and processing of data from the RFID reader, communication with identification cards, efficient database management, and support for CSV file format. The application provides teachers with attendance reports. The result is a fully functional window application for managing and recording student attendance using RFID technology.

Kľúčové slová: RFID; záverečná práca; okenná aplikácia; Windows; C#

Keywords: RFID; thesis; window application; Windows; C#

Rád by som poďakoval svojmu vedúcemu práce Mgr. Radku Janoštíkovi, PhD. za cenné rady a usmernenia počas písania mojej záverečnej práce a programovania aplikácie. Moje vďaka patrí aj mojej rodine, priateľke a priateľom za ich podporu a povzbudenie počas celého môjho štúdia.

Odovzdaním tohto textu jeho autor/ka čestne vyhlasuje, že celú prácu vrátane príloh vypracoval/a samostatne a za použitia iba zdrojov spomínaných v texte práce a uvedených v zozname literatúry.

Obsah

1 Úvod **8**
1.1 Motivácia . . . 8

2 Existujúce riešenia **9**
2.1 Netechnologické riešenia . . . 9
2.2 Technologické riešenia . . . 10
2.3 Zhrnutie . . . 11

3 Použité technológie **12**
3.1 C# . . . 12
3.2 .NET . . . 12
3.3 WPF . . . 13
3.4 XAML . . . 13
3.5 Entity Framework . . . 13
3.6 SQLite . . . 13
3.7 Externé knižnice . . . 14
3.8 RFID . . . 14

4 Návrh riešenia **16**
4.1 Architektúra MVVM . . . 16
4.2 Databáza SQLite . . . 17

5 Programové riešenie **19**
5.1 Zabezpečenie hesla . . . 19
5.1.1 Hashovanie hesla . . . 19
5.1.2 Hashovací nástroj BCrypt . . . 20
5.1.3 Neoprávnené získanie hesla . . . 20
5.2 Prihlasovanie . . . 21
5.2.1 Overenie RFID karty užívateľa . . . 22
5.3 Registrácia . . . 23
5.4 Práca s RFID čítačkou . . . 24
5.5 Dashboard . . . 25
5.5.1 Pridanie nového predmetu . . . 25
5.5.2 Môj profil . . . 27
5.5.3 Odhlásenie a reštart . . . 28
5.5.4 Moje predmety . . . 28
5.6 Detail predmetu . . . 29
5.6.1 Editácia predmetu . . . 30
5.6.2 Vymazanie predmetu . . . 30
5.6.3 Pridanie študentov pomocou CSV . . . 30
5.6.4 Pridanie študentov ručne . . . 31
5.6.5 Editácia študenta . . . 31
5.7 Zápis dochádzky . . . 32

5.8 Vizualizácia a export dochádzky 33
5.8.1 Export vo formáte CSV 33
5.9 Validácia zadaných údajov 34
5.9.1 Tvar užívateľského mena a hesla 34
5.9.2 Tvar ostatných vstupných hodnôt 34

6 Výsledky **35**
6.1 Kvalita riešenia 35
6.2 Potenciálne rozšírenia aplikácie 36

Záver **37**

Conclusions **38**

A Čítačka RFID **39**

B Obsah elektronických dát **40**

Literatúra **41**

Zoznam obrázkov

1 Moodle modul „*Docházka*“ z pohľadu učiteľa. [1] 11
2 Pasívny RFID transpondér, takzvaný "*tag*". [16] 15
3 Bezkontaktná komunikácia RFID. [17] 15
4 Diagram architektúry Model-View-ViewModel. [18] 17
5 Databázové tabuľky a ich atribúty. 17
6 Absence Manager - uvítacie okno aplikácie s prihlasovacím formulárom. 21
7 Absence Manager - výzva na priloženie RFID karty pri prihlásení. 22
8 Absence Manager - okno s registračným formulárom. 23
9 Transpondér zaliaty do plastu RFID karty pod silným osvetlením. 24
10 Absence Manager - záložka *Add New Class* zasadená v *Dashboarde*. 26
11 Absence Manager - záložka *My Profile* zasadená v *Dashboarde*. . . 27
12 Absence Manager - záložka *My Classes* zasadená v *Dashboarde*. . 28
13 Absence Manager - okno *Class Detail*. 29
14 Absence Manager - okno zaznamenávania dochádzky. 32
15 Absence Manager - zobrazenie tabuľky dochádzky študentov. . . . 33
16 Čítacie zariadenie RFID použité počas vývoja aplikácie. 39

1 Úvod

V dnešnej digitalizovanej dobe, kedy technologický pokrok preniká do každého aspektu nášho života, je nevyhnutné implementovať nové technológie aj v oblasti vzdelávania. Významným benefitom moderných technologických nástrojov je ich schopnosť zvyšovať efektívnosť, spoľahlivosť a presnosť procesov, čo je neoceniteľné najmä v akademickom prostredí. Katedra informatiky predstavuje ideálne prostredie pre testovanie a zdokonaľovanie nových technologických aplikácií. Práve tu je možné overiť funkčnosť nových riešení, odstrániť prípadné nedostatky a optimalizovať ich pre ďalšie použitie v edukačných procesoch. V tejto časti predstavím problematiku a motiváciu, ktorou sa zaoberá moja bakalárska práca.

1.1 Motivácia

Efektívna správa dochádzky študentov je jedným z kľúčových aspektov akademického prostredia. Je nevyhnutné, aby vyučujúci mali k dispozícii presné a spoľahlivé údaje o účasti študentov na prednáškach a cvičeniach. V súčasnosti používané metódy často spoliehajú na uloženie spravovaných dát vo fyzickej forme a neposkytujú úroveň jednoznačnosti a bezchybnosti záznamov, ktorá by zabezpečila efektívnu správu dochádzky a spravodlivé hodnotenie študentov.

Cieľom mojej bakalárskej práce je navrhnúť a vyvinúť aplikáciu, ktorá by tieto problémy riešila. Rozhodol som sa tak preto, aby som zjednotil a zefektívnil spôsob, akým vyučujúci spravujú dochádzku svojich študentov. Rovnako ma zaujíma, ako by sa aplikácia tohto typu navrhovala a vyvíjala a ako by som mohol využiť vedomosti a schopnosti, ktoré som nadobudol počas štúdia.

Pri návrhu a implementácii aplikácie kladiem dôraz na jednoduchosť, prenositeľnosť a vysokú mieru možného prispôsobenia konkrétnym potrebám užívateľov rôzneho prostredia. Výsledný systém poskytne vyučujúcim jednoduché, intuitívne a vizuálne príjemné rozhranie pre správu dochádzky, s možnosťou efektívneho zápisu prítomností a absencií u jednotlivých študentov.

2 Existujúce riešenia

Počas môjho viacročného štúdia na univerzite som sa stretol s viacerými spôsobmi zaznamenávania dochádzky študentov do hodín. Bolo to potrebné takmer výlučne iba na také hodiny, na ktorých účasť je pre študentov prezenčných študijných programov povinná – teda cvičenia a semináre v nižších ročníkoch štúdia.

2.1 Netechnologické riešenia

Najčastejším riešením, s ktorým som sa stretol počas rokov, bol záznam dochádzky na dochádzkový hárok. Je to hárok, ktorý obsahuje zoznam všetkých študentov zapísaných na daný predmet v riadkoch spolu s jednotlivými dátumami výuky daného semestra v stĺpcoch. Tento proces prebiehal viacerými formami, ktoré by sa dali zhrnúť do dvoch kategórií:

a) vyučujúci dá hárok cirkulovať medzi študentov, ktorí sa k svojmu menu a pod príslušný dátum výuky vlastnoručne podpíšu; na konci výuky je hárok odovzdaný späť do rúk vyučujúceho

b) vyučujúci si hárok ponechá a nahlas číta mená študentov z hárku, pričom ak sa študent výuky účastní a je prítomný, prihlási sa a dochádzku mu vyučujúci zapíše

Tieto spôsoby zápisu dochádzky sú dostatočné, no narážame pri nich na niektoré problémy, ktoré by rôzne technologické riešenia dokázali čiastočne alebo úplne eliminovať. Medzi ich jednoznačné výhody patrí absolútna jednoduchosť prípravy. Vyučujúci si pomocou informačného portálu študijnej agendy Univerzity Palackého (ďalej IS STAG) vygeneruje zoznam (tabuľku) študentov daného predmetu spolu so stĺpcami obsahujúcimi dátumy výuky v záhlaví.

Stretávame sa však aj s radou nevýhod. Pri spôsobe a) učiteľ nedokáže zabrániť neoprávnenému záznamu dochádzky iným študentom, ktorý dochádzku zapíše neprítomnému spolužiakovi. Tento problém rieši spôsob b) tým, že záznamový hárok sa študentom do rúk vôbec nedostane. Obidva spôsoby a) aj b) nie sú vhodné z hľadiska zdieľania súkromných informácií o študentoch. Pri spôsobe a) si každý prítomný študent dokáže nielen prečítať informácie o dochádzke svojich ostatných spolužiakov, no umožňuje mu získať aj úplné zloženie študentov navštevujúcich daný predmet. Spôsob b) eliminuje prvú časť tohto problému, no pri hlasnom čítaní osobných údajov študentov – mena a priezviska – druhá časť problému pretrváva naďalej. Obe riešenia a) aj b) sú však stále iba uchovávanie dát v neelektronickej podobe, čo je náchylné na stratu tohto fyzického predmetu (záznamový hárok sa stratí – vyučujúci prichádza o záznamy dochádzky z celého časového obdobia), pričom spôsob a) má z dôvodu kolovania hárku medzi študentami vyššiu pravdepodobnosť straty hárku ako spôsob b). Taktiež by opäť z hľadiska ochrany osobných údajov bolo nevhodné, ak by sa tento hárok dostal do rúk niekomu tretiemu, ktorý nemá s výukou nič spoločné.

2.2 Technologické riešenia

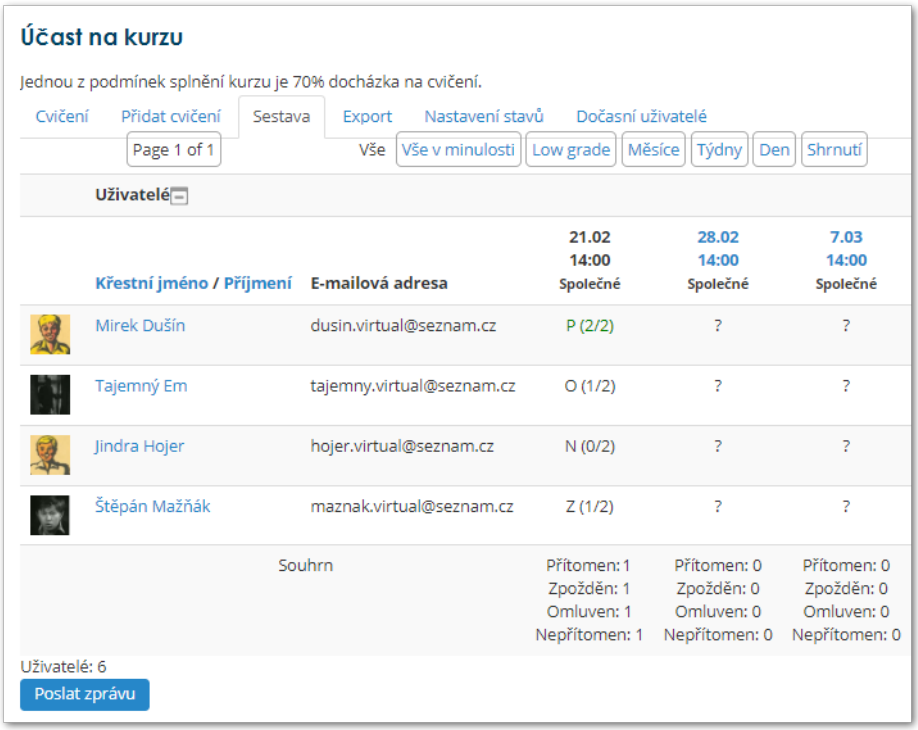
Niektorí vyučujúci sa rozhodli nespoliehať na fyzický dochádzkový hárok a rozhodli sa ho nahradiť kópiou online. Editácia tohto hárku je pre študentov uzamknutá, dochádzku môže zapisovať iba vyučujúci. Všetkým študentom daného predmetu je tento súbor nazdieľaný pomocou nejakej online služby (napríklad *Google Sheets*, *Microsoft Teams* alebo *Moodle*), a tak každý študent môže vidieť svoju dochádzku do hodín, no aj dochádzku svojich spolužiakov. Toto riešenie sa teda stretáva s podobným problémom ochrany súkromia ako predchádzajúce netechnologické riešenia.

Ďalším spôsobom môže byť využitie online služby určenej na zápis dochádzky[1]. Filozofická fakulta Masarykovej univerzity v Brne má k dispozícii neštandardný modul „*Dochádzka*“ v službe *Moodle* (obrázok 1). Umožňuje bodovať dochádzku u jednotlivých študentov podľa prednastavenej škály:

- **P** – prítomnosť (pridelené 2 body)
- **Z** – meškanie (pridelený 1 bod)
- **O** – ospravedlnená neprítomnosť (pridelený 1 bod)
- **N** – neospravedlnená neprítomnosť (0 bodov)

Taktiež tento modul podporuje vytvoriť si vlastnú škálu bodovania, potrebný počet bodov pre získanie zápočtu z daného predmetu, či podmieniť dochádzkou prístup k záverečnému testu, skúške alebo zápisu k ďalšiemu kurzu [1]. Toto riešenie je z hľadiska ochrany súkromia oveľa vhodnejšie, pretože dochádzka študenta do danej hodiny sa zobrazuje iba študentovi, ktorého sa to týka. Implementácia a údržba systému *Moodle* však nemusí byť pre organizáciu ideálne riešenie z finančných a časových dôvodov. Taktiež tento systém poskytuje nespočetné množstvo ďalších funkcií, ktoré organizácia nemusí využívať, preto sa neoplatí inštalovať *Moodle* v organizácii, ktorá by ho využívala iba na zápis a správu dochádzky študentov do hodín.

[1] narozdiel od služieb spomínaných v predošlom odstavci, ktoré slúžili primárne na zdieľanie súboru s dochádzkovým hárkom



Obr. 1: Moodle modul „*Docházka*“ z pohľadu učiteľa. [1]

2.3 Zhrnutie

Všetky tieto riešenia majú svoje výhody, ktoré sa budem snažiť v mojom riešení maximalizovať, no aj značné nevýhody. Moja okenná aplikácia pre systém Windows bude slúžiť pre osobné pracovné zariadenia učiteľov, ktorí vyučujú na základnej, strednej či vysokej škole. Uľahčí a zjednotí rôzne spôsoby zápisu dochádzky, eliminuje neoprávnený zápis dochádzky študentom, ktorí sa vyučovania nezúčastnili a učiteľovi poskytne jednoduchý prehľad dochádzky svojich študentov, na základe ktorého ich môže efektívne hodnotiť.

3 Použité technológie

V tejto kapitole sa pozrieme na technológie, ktoré som použil pri programovaní mojej aplikácie s názvom *Absence Manager*.

3.1 C#

Základným programovacím jazykom aplikácie *Absence Manager* je jazyk C# vyvinutý spoločnosťou Microsoft. Jazyk C# je moderný, objektovo a komponentovo orientovaný programovací jazyk, ktorý programátorom umožňuje vývoj bezpečných a robustných aplikácii širokej škály na platforme .NET. Syntakticky je podobný jazyku C++, no prináša niekoľko vylepšení a moderných funkcií. C# podporuje statické typovanie, dedičnosť, polymorfizmus, ako aj rôzne ďalšie koncepty objektovo orientovaného programovania.

Jazyk C# je silne typovaný a pamäťovo bezpečný, čo znamená, že kontroluje typy počas kompilácie a runtime. To zabraňuje mnohým typom chýb, ako sú napríklad neoprávnené prístupy na nedostupné miesta v pamäti. Taktiež obsahuje takzvaný garbage collector, čo znamená, že programátor sa nemusí obávať o únik pamäte a nemusí ju ručne uvoľňovať.

Aj keď *Absence Manager* vyvíjam pre operačný systém *Windows*, C# je platformovo nezávislý a je ho možné použiť na vývoj aplikácií pre rôzne iné operačné systémy, vrátane *Linux*, *macOS*, *Android* či *iOS*. Taktiež je možné okrem okenných desktopových aplikácií vyvíjať aplikácie určené pre mobilné zariadenia alebo rôzne iné webové aplikácie. [2]

3.2 .NET

Platforma .NET je bezplatný nástroj typu open-source, ktorý slúži na vytváranie mnoho druhov aplikácií. Poskytuje možnosť spúšťať programy napísané v rôznych jazykoch, pričom je najviac previazaná práve s jazykom C#. Používa vysoko výkonný modul *runtime* (CLR), ktorý sa používa v produkčnom prostredí veľkého množstva aplikácií. .NET obsahuje komponenty, ktoré medzi sebou komunikujú a spolupracujú. Medzi piliere platformy .NET patria komponenty modulu *runtime*, ktorý kód prekladá už počas programovania, knižnice, ktoré poskytujú funkčné nástroje, kompilátor a iné. [3]

3.3 WPF

Windows Presentation Foundation (WPF) je technológia patriaca pod platformu .NET. Umožňuje vytvoriť funkčné, interaktívne a vizuálne atraktívne užívateľské rozhranie aplikácií. Využíva deklaratívny prístup k tvorbe užívateľského rozhrania pomocou markup jazyka XAML. Tento prístup oddeľuje dizajn a logiku aplikácie. To umožňuje lepšiu organizáciu kódu a efektívne využitie zdrojov. Pomocou technológie WPF je programátor schopný vytvoriť rôzne vizuálne prvky, ako sú animácie, efekty, transformácie a štýly, čo umožňuje tvorbu dynamického rozhrania. WPF poskytuje podporu vektorovej grafiky, čo znamená, že aplikácie vyvíjané pomocou WPF sú vhodné pre rôzne zariadenia s rôznymi grafickými rozlíšeniami. Medzi ďalšie funkcie, ktoré WPF obsahuje, sú dátové väzby a rôzne ovládacie prvky, ktoré si programátor môže prispôsobiť. [4]

3.4 XAML

Extensible Application Markup Language (XAML) je deklaratívny jazyk, ktorý sa používa v technológii WPF. Vo svojej syntaxi využíva deklaratívne značky, ktorými vytvára viditeľné prvky užívateľského rozhrania. Týmito značkami programátor dokáže oddeliť definíciu užívateľského rozhrania od logiky aplikácie dôsledkom čoho je prehľadný a efektívny vývoj. Prvkami jazyka XAML sú objekty, ich atribúty, kolekcie či návrhové vzory, podľa ktorých si programátor môže navrhnúť vizuálne prvky so špecifickými vlastnosťami. [5]

3.5 Entity Framework

Entity Framework (EF) je objektovo-relačný mapovací (ORM) framework vyvinutý spoločnosťou Microsoft pre platformu .NET. Jedným z jeho hlavných účelov je umožniť vývojárom pracovať s relačnými databázami typu SQL v rámci aplikácií .NET bez potreby písania SQL dotazov priamo v kóde. Namiesto toho sa pracuje s objektami a triedami, pričom EF má za úlohu namapovať tieto objekty na záznamy v databáze a naopak. [6]

3.6 SQLite

SQLite je serverovo nezávislá relačná databáza, ktorá je schopná ukladať dáta do lokálneho súboru. Je navrhnutá tak, aby fungovala bez potreby samostatného servera, čo ju robí ideálnou pre aplikácie, kde je potrebná jednoduchá, efektívna a samostatná databázová správa. Využíva syntax SQL (*Structured Query Language*), čo umožňuje vykonávať štandardné operácie pomocou SQL príkazov. *SQLite* je používaná v rôznych typoch aplikácií, kde slúži na ukladanie dát priamo v zariadeniach užívateľov. V mojom projekte som ju zvolil pre jej jednoduchú integráciu do .NET prostredia cez *Entity Framework*, čo umožňuje jednoduchú manipuláciu s dátami bez nutnosti zriaďovať externý databázový server. [7] [8]

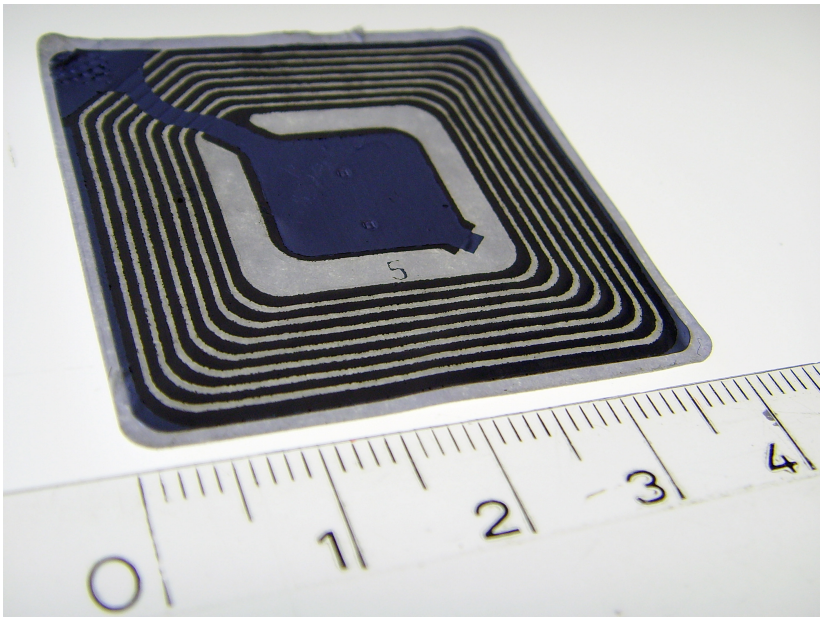
3.7 Externé knižnice

V mojom riešení som tiež využil funkcionalitu viacerých externých knižníc, ktoré som si nainštaloval pomocou nástroja *NuGet*. Nimi sú:

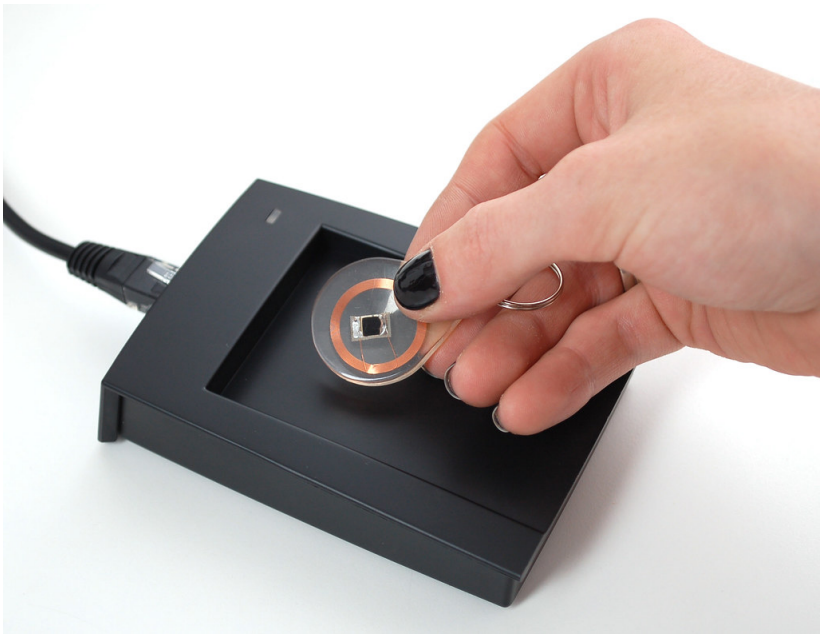
- **BCrypt.NET**: Variácia knižnice *BCrypt* pre jazyk C#. Knižnica *BCrypt* je implementáciou Blowfish hashovacieho algoritmu pre heslá pôvodne vyvinutá pre UNIXový operačný systém *OpenBSD*. Tento nástroj hashuje heslá pomocou upravenej blokovej šifry Blowfish navrhnutej tak, aby zvýšila náklady na prelomenie hesla a znemožnila rýchlu hardvérovú implementáciu nástrojov na prelomenie hesla. [9] [10]
- **Microsoft.EntityFrameworkCore.Sqlite**: Balík nástrojov pre správu SQLite databázy. [11]
- **Microsoft.EntityFrameworkCore.Tools**: Balík nástrojov určených na manažovanie migrácií, vytvorenie databázového kontextu a typov entít podľa databázovej schémy. [12]
- **CsvHelper**: Univerzálna a vysoko výkonná knižnica pre .NET Standard 2.0, ktorá poskytuje spoľahlivé čítanie a zápis CSV súborov, dodržiavajúc štandardy RFC 4180 a zároveň ponúka flexibilitu, nízku spotrebu pamäte a kompatibilitu s rôznymi systémami. [13]
- **System.Text.Encoding.CodePages**: Knižnica umožňujúca vytváranie jedno- a dvojbytových kódovaní textových reťazcov, ktoré nie sú dostupné v základnej verzii nástrojov .NET. Jedná sa napríklad o kódovanie *Windows-1250* (kódovanie znakov českého jazyka), ktoré som použil pri čítaní a zapisovaní dát z a do súborov formátu CSV. [14]

3.8 RFID

Rádiofrekvenčná identifikácia (*Radio-Frequency Identification* - RFID) je bezkontaktná komunikácia (obrázok 3) a predávanie dát medzi transpondérom RFID a zapisovacím alebo čítacím zariadením RFID. Toto zariadenie (v našom prípade čítacie zariadenie RFID) indukuje elektromagnetické pole. Ak sa v medziach tohto poľa nachádza RFID transpondér (obrázok 2), môže dôjsť k výmene dát, komunikácii, zápisu alebo čítania dát do alebo z transpondéra. Konštrukcia transpondéra RFID sa môže líšiť v závislosti od konkrétnej varianty, avšak v základnej podobe obsahuje mikročip a anténu. Pamäť čipu sa pohybuje od 4 bytov po 8 kilobytov, pričom využitie a zabezpečenie pamäte sa prispôsobuje danému použitiu. Princíp funkcie RFID spočíva v indukcii elektromagnetického poľa, ktoré dodáva energiu (v našom prípade pasívnemu) transpondéru. Transpondér môže byť aktívny alebo pasívny. Aktívny transpondér obsahuje vlastný zdroj energie (napríklad batériu) a umožňuje komunikáciu na oveľa väčšiu vzdialenosť, ako pasívny transpondér (obrázok 2), ktorý získava energiu na komunikáciu prostredníctvom indukovaného elektromagnetického poľa. [15]



Obr. 2: Pasívny RFID transpondér, takzvaný "*tag*". [16]



Obr. 3: Bezkontaktná komunikácia RFID. [17]

RFID technológia má široké spektrum využitia v rôznych odvetviach, vrátane logistiky, bezpečnosti, zdravotníctva a dopravy. Je to rýchle, efektívne a jednoduché riešenie pre sledovanie a identifikáciu objektov a ľudí v reálnom čase. [15]

4 Návrh riešenia

V tejto kapitole sa pozrieme na štruktúru celého projektu, spôsob ukladania dát, architektúru aplikácie.

4.1 Architektúra MVVM

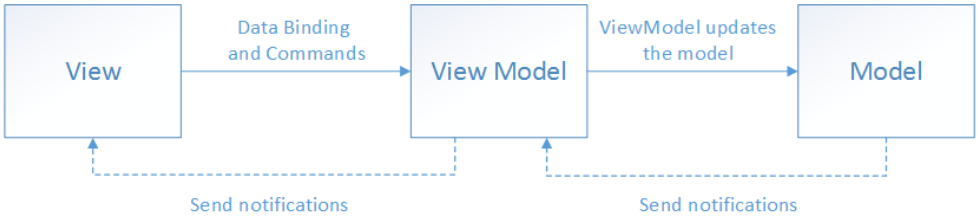
Pri vývoji *Absence Manager* som si vybral návrhový vzor architektúry MVVM (*Model-View-ViewModel*), ktorý sa mi javil ako najlepšie riešenie pre aplikáciu typu *Windows Presentation Foundation*. MVVM oproti iným typom architektúr ponúka silné oddelenie implementačnej logiky od užívateľského rozhrania, robustnú podporu dvojsmerných dátových väzieb a lepšiu integráciu s funkčnosťami *Windows Presentation Foundation*, čo uľahčuje vývoj, testovanie a údržbu aplikácií. Táto architektúra zabezpečuje rozdelenie kódu do viacerých súborov, ktoré sa radia do troch kategórií:

- **Model**: V súboroch typu „Model“ sa nachádzajú triedy, ktoré zapuzdrujú dáta aplikácie. Jedná sa o triedy, ktoré definujú dátové typy, a s ktorými pracujem v celej štruktúre aplikácie. Moje modely sa starajú o korektnú definíciu tried užívateľov aplikácie, študentov, vyučovacích predmetov, záznamov o dochádzke a absencii. V tejto časti štruktúry je zahrnutý aj súbor „UserContext.cs“, ktorý zabezpečuje korektné pripájanie k SQLite databázovému súboru, kde ukladám a spravujem dáta aplikácie. Samotné súbory typu „Model“ nevedia o žiadnych zmenách v ostatných častiach kódu, slúžia len ako vzor, podľa ktorého definujeme, ako majú vyzerať vstupné, výstupné dáta a jednotlivé objekty, s ktorými pracujeme. [18]

- **View**: Súbory typu „View“ sa starajú o vizuálnu časť môjho projektu. Sú to súbory napísané jazykom XAML, ktoré zobrazujú užívateľské rozhranie používateľom aplikácie. Obsahuje textové polia, tlačidlá, tabuľky a ďalšie objekty. Pomocou tohto rozhrania je užívateľovi poskytnutá všetka funkcionalita aplikácie. Objekty, ktoré zobrazujú dáta a informácie, sú naviazané na vlastné ViewModely pomocou datových väzieb (Data Binding). Každý View má okrem XAML súboru k dispozícii aj takzvaný „code-behind“ súbor formátu C#, v ktorom môže programátor definovať správanie objektov, ktoré sa nedá definovať jazykom XAML (napríklad animácie objektov, jednoduché akcie tlačidiel, alebo iné vizuálne prvky). [18] *Absence Manager* obsahuje niekoľko Views, pričom nie každé z nich je samostatné okno. DesktopView je View, ktorý zobrazuje užívateľom ich základnú pracovnú plochu. Okrem tlačidiel obsahuje dynamickú časť „Content“, ktorá sa zobrazuje na základe zvoleného tlačidla. Užívateľ si tak môže zobraziť priamo v pracovnej ploche ním vyučované predmety, pridanie nového predmetu alebo svoj užívateľský profil. Tieto Views nie sú samostatné okná, pre svoje načítanie a zobrazenie potrebujú byť vsadené do DesktopView.

- **ViewModel**: Treťou zložkou MVVM architektúry sú súbory typu „ViewModel“. Zabezpečujú pre Views logiku, príkazy pre stlačené tlačidlá a implementujú správne vypĺňanie formulárov. Na Views sú naviazané pomocou dátových väzieb a svoje View upozorňujú na zmenu dát prostredníctvom vyvolávania udalostí o zmenách (*PropertyChanged* a *INotifyPropertyChanged*). ViewModelu sú k dispozícii tiež všetky triedy typu Model, s ktorými pracuje pri spracovaní dát alebo pri vytváraní dát nových. [18] V mojom projekte má každý View svoj vlastný ViewModel, ktorý zabezpečuje správny chod všetkých jeho objektov.

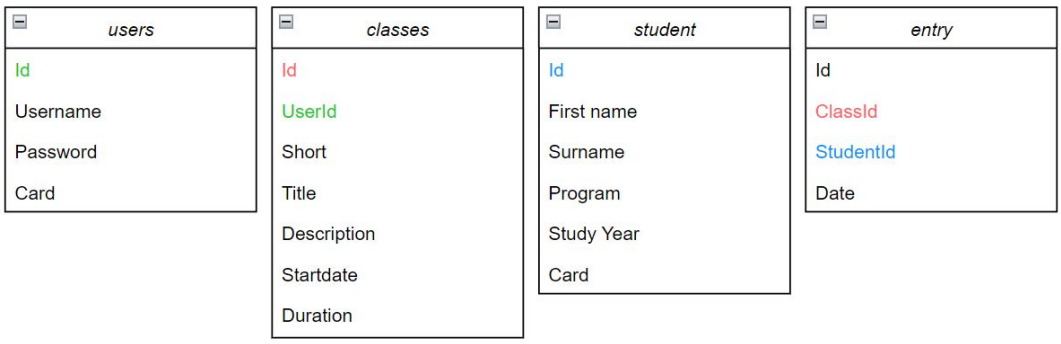
Toto rozdelenie kódu architektúrou *Model-View-ViewModel* má za následok vyšší výkon aplikácie, kód je prehľadnejší, čitateľnejší, umožňuje znovupoužitie kódu a uľahčuje pridávať do projektu novú funkcionalitu v budúcnosti.



Obr. 4: Diagram architektúry Model-View-ViewModel. [18]

4.2 Databáza SQLite

Pre ukladanie dát potrebných pre správny chod *Absence Manager* a korektné manažovanie dochádzky som zvolil databázu Entity Framework, ktorá pracuje s relačným databázovým modelom SQLite. Dáta som rozdelil do jednotlivých skupín a typov a podľa nich som navrhol databázový model.



Obr. 5: Databázové tabuľky a ich atribúty.

Atribúty zvýraznené rovnakou farbou signalizujú previazanosť jednotlivých atribútov a tabuliek. Moja databáza má teda štyri tabuľky:

- *users* -– tabuľka užívateľov aplikácie a ich údaje.
- *classes* -– tabuľka všetkých vytvorených predmetov, ktoré vyučujú užívatelia a ktoré sú k nim jednoznačne pridelené.
- *student* -– tabuľka obsahujúca všetkých vytvorených študentov. Každý študent má jednoznačne pridelený presne jeden predmet, ktorý navštevuje.
- *entry* -– tabuľka dochádzky. Ak sa študent v daný deň dochádzky zúčastnil, je mu vytvorený záznam v tejto tabuľke.

Prístup k databáze zabezpečuje Model „UserContext.cs“. V priečinku *data* sa nachádza súbor „UserData.cs“, ktorý obsahuje všetky metódy na získanie, spracovanie, zapisovanie a mazanie dát z databázy. Majú k nemu prístup všetky ostatné časti aplikácie a teda zvyšuje znovupoužiteľnosť kódu.

5 Programové riešenie

V tejto kapitole predstavím zabezpečenie citlivých užívateľských údajov uložených v mojej databáze, predstavím jednotlivé okná a funkcionalitu aplikácie ako prihlasovanie, registrácia, vytvorenie alebo zvolenie vyučovaného predmetu, pridanie alebo úpravu študentov navštevujúcich daný predmet a zápis dochádzky.

5.1 Zabezpečenie hesla

Zabezpečenie citlivých užívateľských údajov je kľúčovým aspektom dôvery užívateľov voči systémom. Pri ukladaní citlivých údajov, ako sú heslá, je nevyhnutné používať robustné metódy šifrovania. Heslo ako citlivý osobný údaj užívateľa nikdy nemôžeme ukladať ako jednoduchý text. V prípade, že útočník získa prístup k databáze, kde sú heslá uložené ako jednoduchý text, má takmer okamžitý prístup k prihlasovacím údajom užívateľa. Z tohto dôvodu je dôležité venovať dostatočnú pozornosť zabezpečeniu hesla ako citlivého údaju. Ďalším aspektom dobre zabezpečeného hesla je jeho tvar. Užívatelia, ktorí si volia príliš krátke a jednoduché heslá, ktoré obsahujú iba generické znaky, sa taktiež môžu stať terčom útočníkov. V tejto časti predstavím všeobecný problém ukladania hesiel, možné riziká vyplývajúce z nesprávneho ukladania hesla a aký prístup som zvolil v mojom projekte ja.

5.1.1 Hashovanie hesla

Hashovanie hesla je kryptografická technika, ktorá slúži na zabezpečenie užívateľských hesiel v informačných systémoch. Pri hashovaní hesla sa pôvodné heslo prevedie na vždy rovnako dlhý reťazec znakov, nazývaný hash, pomocou matematického algoritmu, nazývaného hashovacia funkcia. Kryptografická hashovacia funkcia je jednosmerný matematický algoritmus, ktorý mapuje dáta akejkoľvek veľkosti na bitový reťazec pevnej veľkosti. Tieto funkcie sa používajú pri zabezpečení rôznych informácií, ale tiež pri digitálnych podpisoch alebo kódoch pre overovanie pravosti správ. [19] Kryptografické hashovacie funkcie by mali spĺňať nasledujúce základné požiadavky:

- **Jednosmernosť**: Hashovacia funkcia je jednosmerná, teda je jednoduché zo vstupných dát získať *hash*, ale veľmi zložité získať vstupné dáta z hodnoty *hash*.
- **Nemennosť**: Rovnaké vstupné údaje hashovacia funkcia prevedie na rovnakú a hodnotu *hash*.
- **Lavínovitosť**: Iba malá zmena (napríklad jedného znaku) vo vstupných údajoch má za následok úplnú zmenu hodnoty *hash*.
- **Bezkolíznosť**: Ak pre dva rôzne vstupy algoritmus vygeneruje rovnakú hodnotu *hash*, hovoríme, že nastala kolízia. Bezkolíznosť hashovacích algoritmov by mala byť v najväčšej možnej miere maximalizovaná. [20]

5.1.2 Hashovací nástroj BCrypt

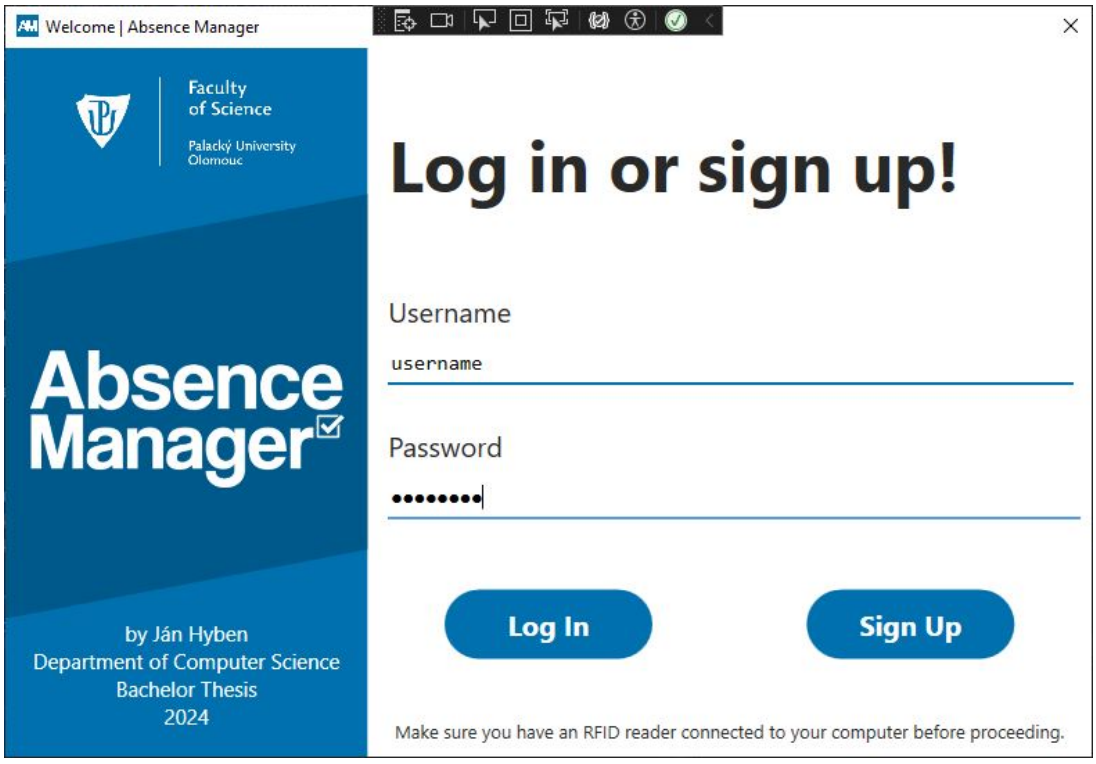
Hashovací nástroj *BCrypt*, ktorý používam pomocou importovanej externej knižnice *BCrypt.NET* slúži na pridelenie hashu pre zadané heslo akéhokoľvek typu. Užívateľské heslo od užívateľa získavam pomocou objektu *BindablePasswordBox*, ktorý som si sám naprogramoval. Tento objekt má podobné vlastnosti ako *PasswordBox*, dá sa však dátovo naviazať na atribúty v súbore ViewModel. Po prečítaní dát z tohto objektu s nimi pracujem v dátovom type *SecureString*, ktorý je súčasťou systémovej knižnice. Údaj typu *SecureString* pomocou hashovacieho nástroja *BCrypt* prevádzam na vždy rovnako dlhý *hash*, ktorý ukladám v databáze. Tento *hash* potom pri prihlasovacom procese pomocou funkcie poskytovanej knižnicou *BCrypt.NET* overujem so zadaným heslom užívateľa a ak sa zhoduje, prihlasovací proces je úspešne ukončený.

5.1.3 Neoprávnené získanie hesla

Použitie hashovacieho nástroja zabezpečí, že užívateľské heslo z databáze nezistí žiadna tretia osoba, aj keď by sa k databázovému súboru dostala. Vychádza to z vlastností hashovania a hashovacích algoritmov, ktoré sú okrem iných vlastností jednosmerné, teda z hodnoty *hash* sa nedá (alebo je veľmi výpočtovo náročné a zdĺhavé) získať pôvodné údaje. Ďalším sťažením prelomenia prihlasovacieho procesu je požiadavka na priloženie zamestnaneckej RFID karty po úspešnom prihlásení. Tým zabezpečím, že aj keď tretia strana samotné heslo zistí, do aplikácie sa nedostane a žiadne údaje nebudú zmenené.

5.2 Prihlasovanie

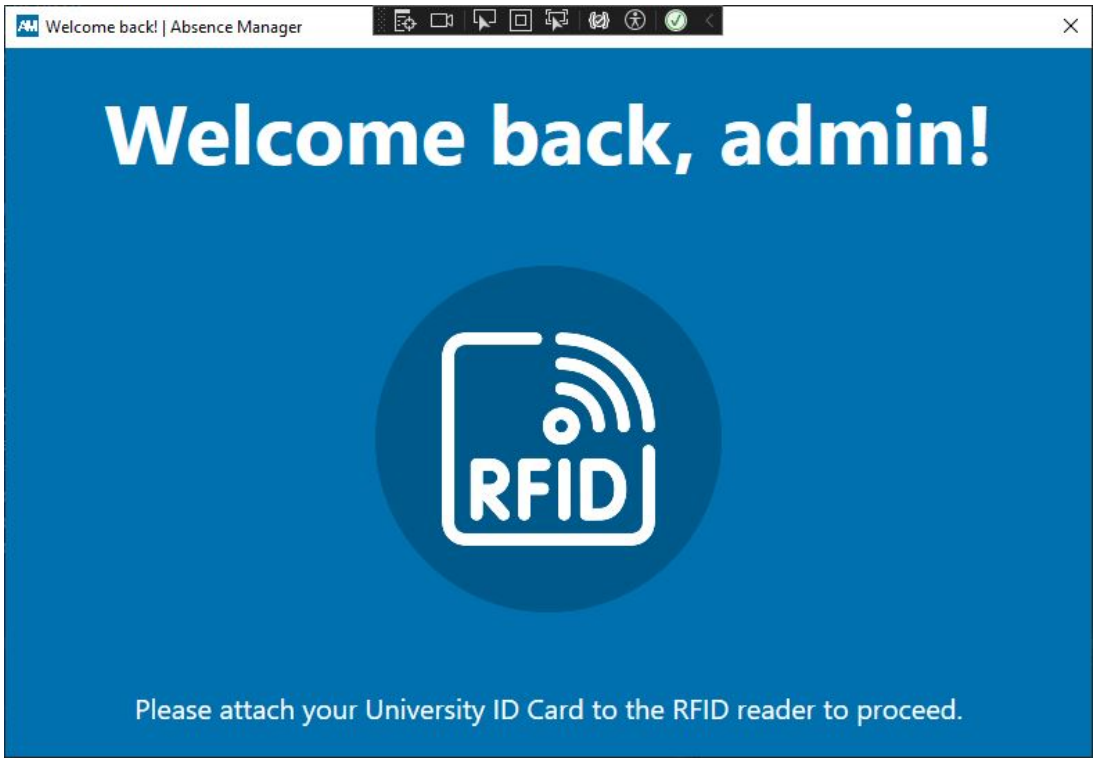
Prihlasovanie je jedným z najdôležitejších faktorov aplikácie *Absence Manager*. Slúži na verifikáciu identity používateľa aplikácie, teda vyučujúceho. Prihlasovacie okno (obrázok 6) sa zobrazí ako prvé okno po spustení aplikácie. Obsahuje dve polia – *Username* (používateľské meno) a *Password* (heslo) a dve tlačidlá – *Log In* (prihlásiť) a *Sign Up* (vytvoriť nový profil). Tlačidlo *Log In* je spočiatku zablokované a dynamicky kontroluje validitu zadaných údajov. V tomto formulári tomu nemusíme venovať až takú veľkú pozornosť, kontrolu údajov som pridal, aby užívateľ neodoslal formulár s prázdnym poľom. Tlačidlo *Log In* má nastavený príznak *IsDefault = true*, čo v praxi znamená, že odoslanie vyplneného formuláru môže užívateľ vykonať pomocou klávesy *Enter*, ak sú údaje validné a tlačidlo pre prihlásenie je odomknuté. Používateľ sa prihlási pomocou užívateľského mena a hesla, ktoré si zvolil pri registrácii. Pri zadaní nesprávnych údajov prihlásenie zlyhá a užívateľovi sa zobrazí informačné okno s oznámením, že prihlásenie zlyhalo. Okrem funkčných vlastností obsahuje prihlasovacie okno v ľavej časti grafiku s logom *Absence Manager*, logo Přírodovědecké fakulty UPOL a moje bibliografické údaje. Prihlasovanie zabezpečuje View „LoginView.xaml“, chod a logiku tomuto oknu dodáva ViewModel „LoginViewModel.cs“.



Obr. 6: Absence Manager - uvítacie okno aplikácie s prihlasovacím formulárom.

5.2.1 Overenie RFID karty užívateľa

Ak užívateľ prihlasovací formulár odošle, zadané užívateľské meno a heslo sa skontroluje s údajmi v databáze, konkrétne v tabuľke *users*. Po úspešnom prihlásení sa užívateľovi zobrazí okno s výzvou na priloženie svojej zamestnaneckej RFID karty k čítaciemu zariadeniu. Táto dodatočná požiadavka pri prihlasovaní posilňuje bezpečnosť a pridáva tak ďalšiu vrstvu obrany pred potencionálnym prelomením prihlasovacieho procesu. Jedná sa o jednoduché okno (obrázok 7), ktoré obsahuje uvítanie užívateľa, výzvu na priloženie svojej zamestnaneckej karty a piktogram RFID. Tento piktogram RFID používam aj v ďalšej časti *Absence Manager*. Zvolil som tento piktogram, ktorý jednoznačne signalizuje potrebu priloženia identifikačnej karty k čítaciemu zariadeniu, teda užívateľovi alebo študentovi bude ihneď jasné, akú akciu je potrebné vykonať. Okrem spomenutých objektov okno obsahuje aj skryté textové pole, ktoré každých 0,5 sekundy skenuje, kontroluje a vymazáva toto textové pole. Keďže moje RFID čítacie zariadenie funguje ako virtuálna klávesnica, po priložení karty sa do aktívneho poľa vypíše číselné označenie RFID transpondéru z priloženej karty. Priloženie nesprávnej zamestnaneckej karty nie je nijak signalizované, nesprávne hodnoty sa z poľa vymažú a program čaká na priloženie správnej karty k čítačke. View „WelcomeBackView.xaml“ a spolu s ViewModelom „WelcomeBackViewModel.cs“ zabezpečujú správne fungovanie tohto overovacieho okna.



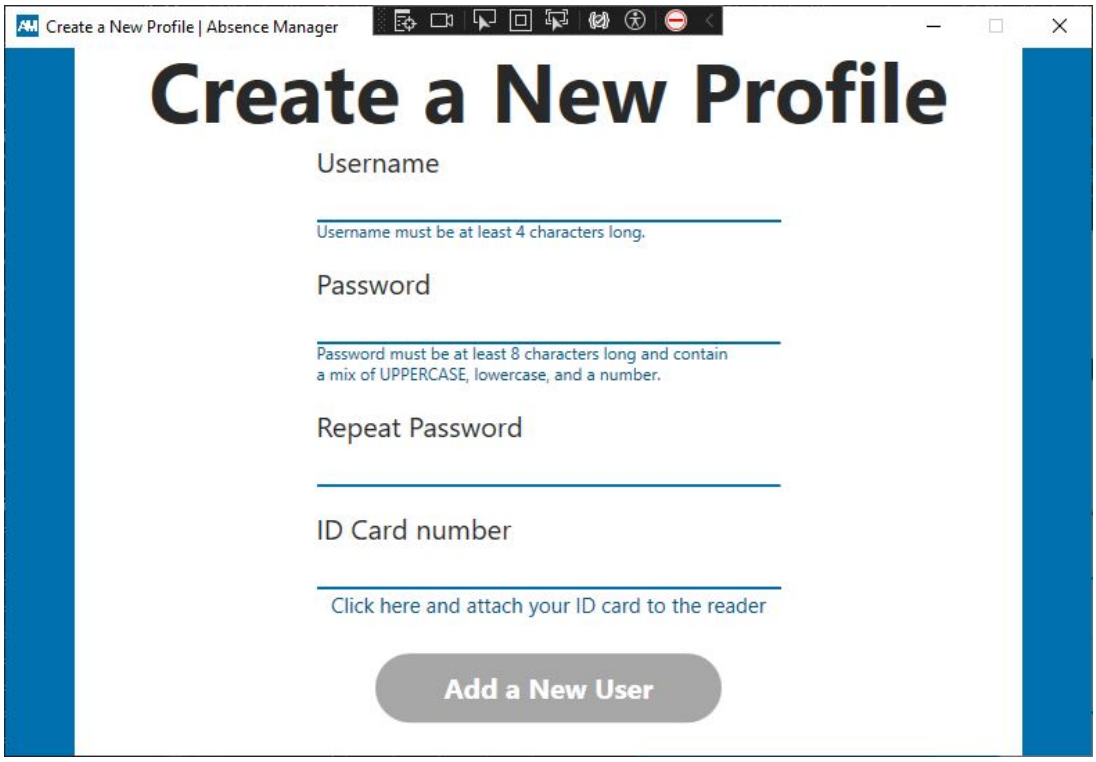
Obr. 7: Absence Manager - výzva na priloženie RFID karty pri prihlásení.

5.3 Registrácia

Pri prvotnom otvorení *Absence Manager* novým užívateľom je potrebné, aby si vytvoril nový používateľský účet. Registrovať sa v mojej aplikácii dá kliknutím na tlačidlo *Sign In* na uvítacej obrazovke, ktoré je vždy aktívne. Po kliknutí sa užívateľovi otvorí nové okno s registračným formulárom (obrázok 8), kde musí vyplniť nasledujúce údaje:

- **Username** – užívateľské meno
- **Password** – užívateľské heslo
- **Repeat Password** – opakovanie užívateľského hesla
- **ID Card Number** – zamestnanecká identifikačná karta

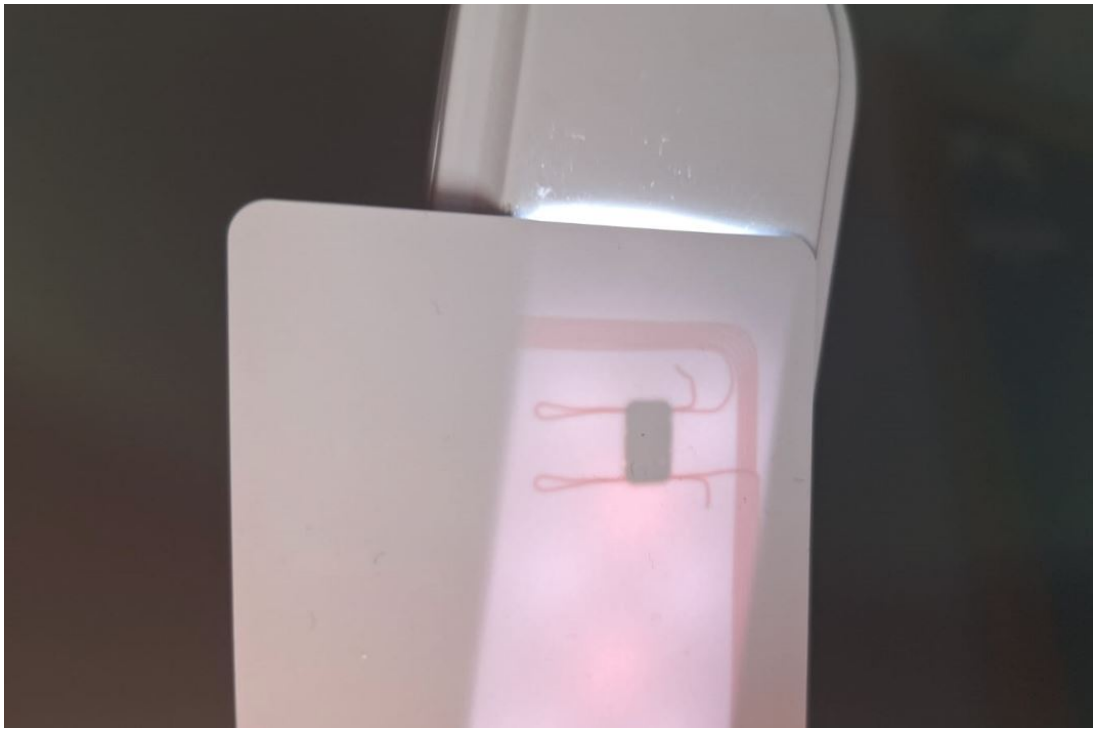
Po korektnom vyplnení údajov užívateľ klikne na tlačidlo *Add a New User* a zobrazí sa mu informačné okno informujúce o úspešnom pridaní nového užívateľa. Po zatvorení tohto informačného okna sa okno s registráciou zatvorí a užívateľ s novovytvoreným profilom sa môže prihlásiť. Proces registrácie je definovaný pomocou View „SigninView.xaml“ a jeho ViewModelu „SignInViewModel.cs“.



Obr. 8: Absence Manager - okno s registračným formulárom.

5.4 Práca s RFID čítačkou

Ako som už spomínal v predošlých častiach, moja RFID čítačka sa pripája k počítaču pomocou rozhrania USB ako virtuálna klávesnica. Pri priložení RFID transpondéru čítacie zariadenie vypíše sled znakov za sebou pomocou kláves 0 až 9 v hlavnej časti klávesnici. Tento sled znakov je identifikačné číslo čipu transpondéru zataveného v plaste identifikačnej karty (obrázok 9). Vo všetkých prípadoch, kedy je vyžadované priloženie identifikačnej karty k čítaciemu zariadeniu, je toto číslo čipu čítané do textového poľa. V oknách, kde sa nachádza piktogram RFID je toto textové pole skryté a teda užívateľ nevidí, kde a ako sa tieto údaje v okne zapisujú. Pri registrácii a pridávaní či editácii študentov navštevujúcich predmet je však potrebné do príslušného poľa najprv kliknúť, aby sa stalo aktívnym a dalo sa do neho vpisovať. Po priložení karty sa do tohto poľa zapíše sled znakov reprezentujúci identifikačné číslo čipu - údaj je správne zachytený. Detailný popis čítacieho zariadenia RFID, ktoré som používal pri vývoji mojej aplikácie nájdete v prílohe A mojej práce.



Obr. 9: Transpondér zaliaty do plastu RFID karty pod silným osvetlením.

5.5 Dashboard

Užívateľ sa po úspešnom prihlásení a priložení svojej univerzitnej RFID identifikačnej karty dostáva na základnú pracovnú plochu *Absence Manager – Dashboard*. Táto pracovná plocha obsahuje záhlavie, v ktorom sa nachádza logo *Absence Manager* a naľavo od neho informácia o práve prihlásenom používateľovi - užívateľské meno profilu. Na pravej strane Dashboardu je umiestnená navigačná lišta so štyrmi tlačidlami. Pomocou tejto lišty sa užívateľ orientuje v rôznych častiach aplikácie. Tieto štyri tlačidlá sú (v smere zhora-dole):

- **My Classes** – zobrazenie tabuľky s predmetmi, ktoré užívateľ vyučuje.
- **Add New Class** – zobrazenie formuláru na pridanie nového predmetu.
- **My Profile** – zobrazenie informácií o užívateľskom profile prihláseného používateľa spolu s formulárom slúžiacemu k zmene týchto údajov.
- **Logout & Restart** – odhlásenie z aplikácie a reštart *Absence Manager* so zobrazením uvítacej prihlasovacej obrazovky.

Okrem týchto prvkov *Dashboard* obsahuje priestor *content*, do ktorého sa zasadí záložka, ktorú si užívateľ vyberie z navigačnej lišty. Toto zobrazenie je dynamicky generované a závisí od tlačidla, ktoré bolo na navigačnej lište naposledy aktivované. Pri prihlásení a otvorení *Dashboardu* sa natívne zobrazí v priestore *content* zobrazenie *My Classes* – teda tabuľka vyučovaných predmetov daného užívateľa. Rozhodol som zobraziť práve tento View, nakoľko ho považujem spomedzi všetkých záložiek za najdôležitejší. Okno *Dashboard* je otvorené iba jedenkrát a ostáva otvorené počas celého behu *Absence Manager*, teda žiadna užívateľská akcia nemá za následok zatvorenie tohto okna. Okno *Dashboard* generuje kód napísaný v súbore „DesktopView.xaml“ a o správu tlačidiel, akcií a zobrazovacej logiky časti *content* jeho ViewModel „DesktopViewModel.cs“.

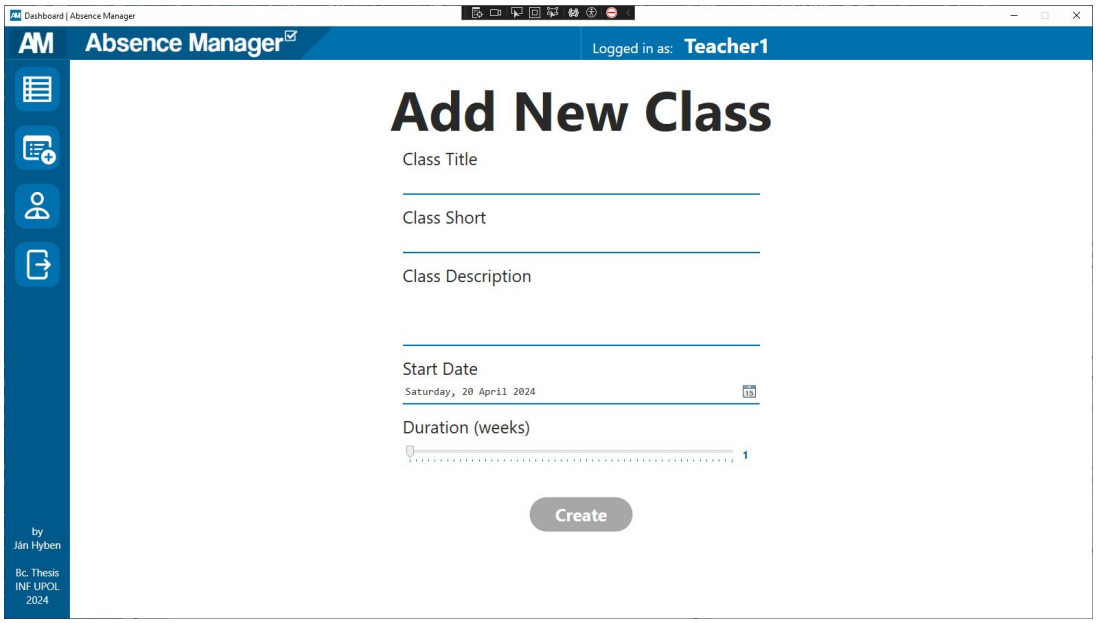
5.5.1 Pridanie nového predmetu

Po prvotnom prihlásení nového užívateľa do *Absence Manager* nemá ešte pridaný žiadny vyučovací predmet. Ak sa po prihlásení dostane do hlavnej obrazovky aplikácie *Dashboard*, kde je natívne zobrazená tabuľka ním vyučovaných predmetov, v záhlaví tejto stránky sa mu zobrazí správa, že k jeho profilu sa neviaže žiadny predmet. V záložke *Add New Class* má k dispozícii formulár (obrázok 10), ktorý mu umožní nový predmet pridať a naviazať ho k svojmu profilu. Väzba predmetu a užívateľského profilu prebehne po odoslaní formulára a k zázname o predmete sa v databáze uloží *ID* užívateľa. Predmetov môže užívateľ pridávať neobmedzené množstvo, podľa jeho potreby.

Formulár pre pridanie nového predmetu obsahuje nasledujúce polia:

- **Class Title** – názov predmetu (napríklad *Diskrétní struktury 1*).
- **Class Short** – skratka pre názov predmetu, častokrát aj s označením katedry (napríklad *KMI/DISK1*).
- **Class Description** – popis predmetu. Užívateľ si môže zvoliť ľubovoľný text, ktorý popisuje jeho predmet, napríklad učebňu, kde sa predmet vyučuje alebo iné informácie.
- **Start Date** – dátum prvej hodiny predmetu v danom semestri. Od tohto dňa sa budú každý týždeň počítať jednotlivé hodiny výuky.
- **Duration** – trvanie predmetu vyjadrené celým číslom v týždňoch. Užívateľ si v rozsahu <1, 52> vyberie počet týždňov výuky daného predmetu. Spolu s vlastnosťou *Start Date* sa vypočítajú jednotlivé dátumy výuky počas celého semestra. Medzi týmito dátumami výuky je vždy rozdiel 7 dní.

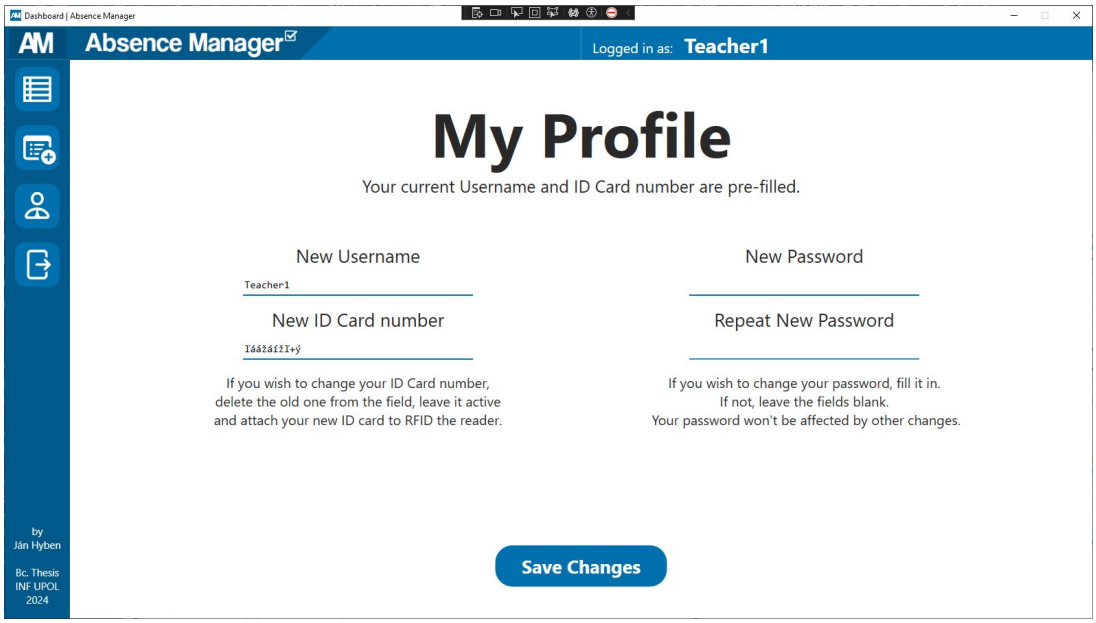
Tento View pre pridávanie nového predmetu generuje súbor „AddNewClassView.xaml“ a operačnú logiku zabezpečuje súbor ViewModel „AddNewClassViewModel.cs“.



Obr. 10: Absence Manager - záložka *Add New Class* zasadená v *Dashboarde*.

5.5.2 Môj profil

V záložke *My Profile* užívateľ uvidí formulár s predvyplnenými údajmi o svojom profile (obrázok 11). Zvolil som preto taký spôsob zobrazenia, že formulár je predvyplnený (okrem užívateľského hesla) aktuálnymi údajmi o užívateľovi z databáze. Tým sa mu editácia uľahčí. Tieto údaje má možnosť zmeniť tým, že ich prepíše a formulár tlačidlom *Save* odošle. Pri zmene údajov musia byť taktiež dodržané ustanovené požiadavky na tvar užívateľského mena a hesla. Pri zmene svojich údajov má užívateľ na výber, ktoré údaje chce editovať a ktoré ponechať. Ak si užívateľ nepraje zmeniť heslo, ale iba niektorý z údajov na ľavej strane zobrazenia, ako je užívateľské meno alebo číslo svojej identifikačnej karty, formulár na pravej strane (polia *New Password* a *Repeat New Password*) môže nechať prázdny. Avšak ak akcia, na ktorú je dátovo naviazané tlačidlo *Save* zistí, že polia hesla nie sú prázdne, tlačidlo sa zablokuje a očakáva sa zmena hesla alebo vyprázdnenie polí hesla. Po úspešnej zmene údajov sa užívateľovi zobrazí informačné okno s informáciou o úspešnom aktualizovaní dát v databáze. Záložku *My Profile* generuje kód zo súboru „MyProfileView.xaml“ a o jeho operačnú logiku sa stará „MyProfileViewModel.cs“.



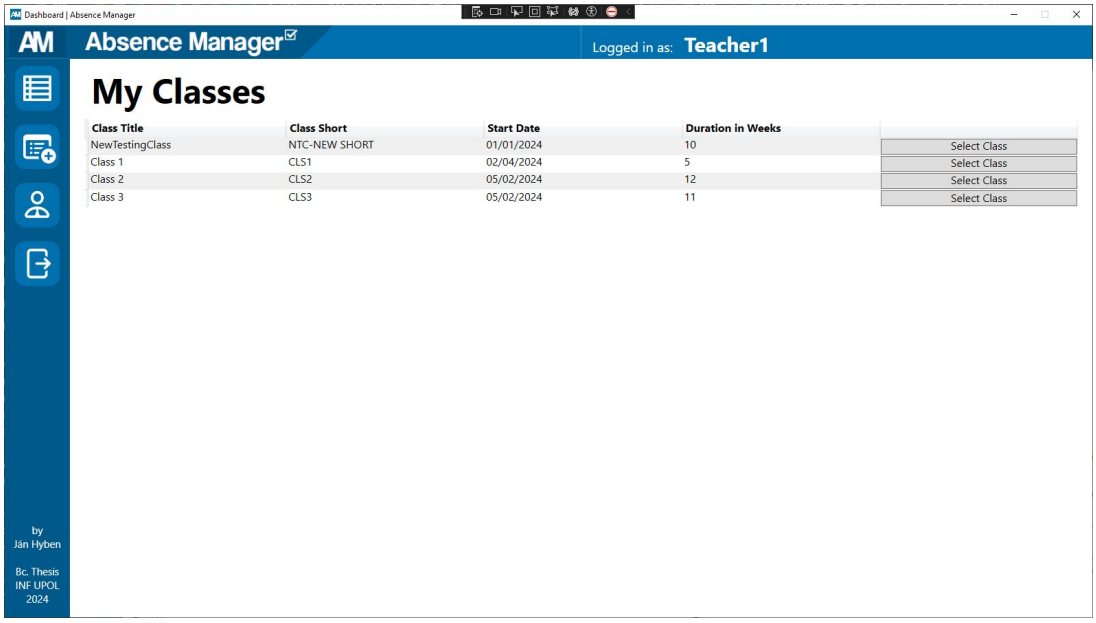
Obr. 11: Absence Manager - záložka *My Profile* zasadená v *Dashboarde*.

5.5.3 Odhlásenie a reštart

Kliknutím na štvrté tlačidlo v ponuke *Dashboardu* sa užívateľ odhlási a *Absence Manager* sa reštartuje na uvítacom okne, ktoré obsahuje prihlasovací formulár.

5.5.4 Moje predmety

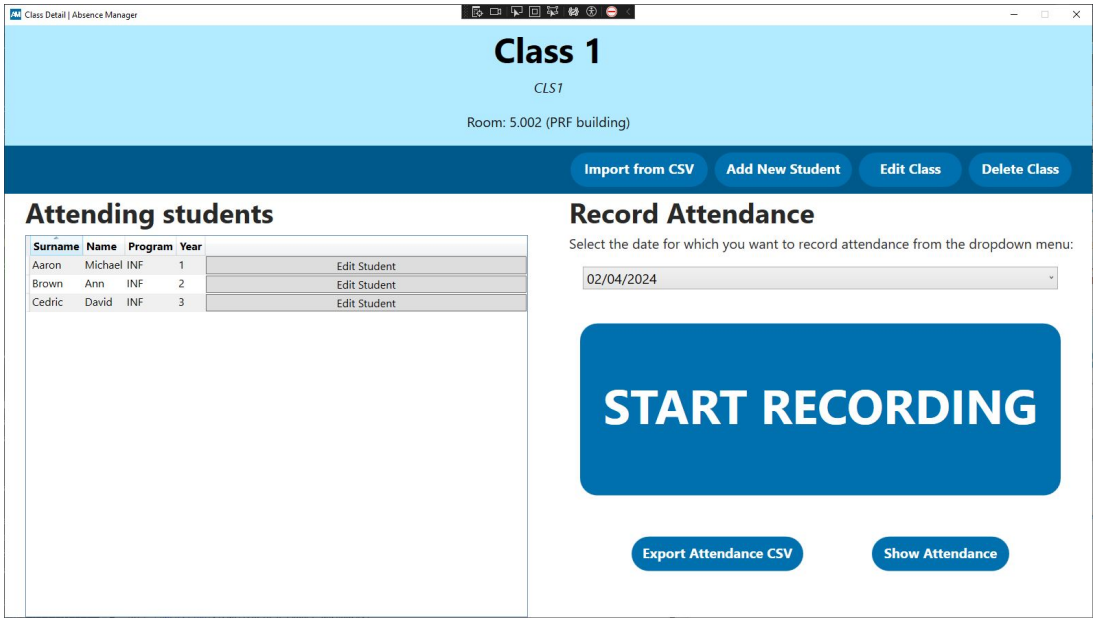
Ako som už spomínal pri popise Dashboardu, záložku *My Classes* (moje predmety) spomedzi ostatných záložiek Dashboardu považujem za najdôležitejšiu. Obsahuje zoznam predmetov, ktoré prihlásený používateľ vytvoril. Sú jednoznačne priradené k jeho profilu. Tento zoznam sa nachádza v tabuľke, ktorá okrem vlastností ako sú *Class Title*, *Class Short*, *Start Date* či *Duration* obsahuje aj samostatný stĺpec obsahujúci tlačidlo *Select Class* (obrázok 12). Pomocou tohto tlačidla si užívateľ zvolí daný predmet v riadku a v ďalších oknách s ním môže detailne pracovať. V tomto zozname si môže užívateľ prispôsobiť poradie zobrazovaných predmetov kliknutím na záhlavie stĺpca, a tak ich zoradiť abecedne zoradiť podľa názvu, skratky, alebo zoradiť podľa počiatočného dátumu výuky alebo jej trvania vzostupne či zostupne. Dokonca môže jednotlivé stĺpce tabuľky aj vymeniť medzi sebou potiahnutím ich záhlavia. Ak užívateľ nemá pridané žiadne predmety, pri nadpise okna sa mu zobrazí informácia, ktorá ho upozorní, že nemá k svojmu profilu pridané žiadne predmety a môže tak vykonať v záložke Dashboardu *Add New Class*. View *My Classes* nájdeme definovaný v súbore „MyClassesView.xaml“ a jeho logiku v súbore „MyClassesViewModel.cs“.



Obr. 12: Absence Manager - záložka *My Classes* zasadená v *Dashboarde*.

5.6 Detail predmetu

Okno s detailnými informáciami o danom predmete (obrázok 13) sa užívateľovi zobrazí po kliknutí na tlačidlo *Select Class* v jednom z riadkov tabuľky obsahujúcej predmety. Je to základné zobrazenie pre daný predmet, z ktorého má užívateľ prístup k širokej škále funkcií slúžiacim k správe predmetu a študentov, no najmä sa z tohto zobrazenia spúšťa samotný zápis dochádzky. Prístupy k týmto funkciám zabezpečujú tlačidlá rozmiestnené v tomto View. V záhlaví zobrazenia sú umiestnené informácie o vybranom predmete, konkrétne jeho názov, skratka a popis. Potom nasleduje panel s tlačidlami *Import from CSV*, *Add New Student*, *Edit Class* a *Delete Class*. Jednotlivé funkcie týchto tlačidiel si postupne predstavíme. Naľavo má užívateľ možnosť vidieť zoznam všetkých študentov, ktorí navštevujú vybraný predmet. Ak je tento zoznam prázdny, teda žiadny študent nie je priradený k vybranému predmetu, pri nadpise *Attending students* sa užívateľovi zobrazí správa s informáciou ako študentov do predmetu pridať. Zoznam je riešený tabuľkou rovnako ako pri zobrazení predmetov užívateľa a okrem informácií študentov obsahuje stĺpec s tlačidlami *Edit Student*, ktoré umožňujú užívateľovi upraviť informácie o študentovi. Pravá časť okna s nadpisom *Record Attendance* umožňuje užívateľovi zápis dochádzky študentov pre vybraný dátum výuky, zobrazenie dochádzky a export dochádzky do súboru formátu CSV. Správne zobrazenie a chod tohto View zabezpečujú súbory „SelectedClassView.xaml“ a „SelectedClassViewModel.cs“. Jedná sa o rozsiahlejšie súbory môjho projektu.



Obr. 13: Absence Manager - okno *Class Detail*.

5.6.1 Editácia predmetu

Editáciu predmetu užívateľ spustí kliknutím na tlačidlo *Edit Class* v *Class Detail* okne. V aplikácii sa otvorí nové okno s krátkym formulárom, kde sa nachádzajú predvyplnené údaje o vybranom predmete. Užívateľ tak môže zmeniť názov predmetu, jeho skratku či popis. Z dôvodov kontinuity a konzistencie dát nie je možné zmeniť údaj *Start Date* alebo *Duration*. Ak si užívateľ praje zmeniť jeden alebo obidva z týchto údajov, je odporúčané si založiť novú triedu. Pri zmene týchto dát by nebolo možné ľahko vypočítať nové dátumy výuky a pridelená dochádzka by taktiež nebola platná. Údaje užívateľ uloží kliknutím na tlačidlo *Save*. Po úspešnej aktualizácii údajov v databáze je užívateľ informovaný pomocou informačného okna a okno s editáciou predmetu sa zatvorí. View detail predmetu aktualizuje zobrazované údaje o vybranej triede. Editáciu predmetu generuje View „EditClassView.xaml“ a funkcionalitu „EditClassViewModel.cs“.

5.6.2 Vymazanie predmetu

Ak si užívateľ praje predmet vymazať, môže tak vykonať kliknutím na tlačidlo *Delete Class* v *Class Detail* okne. Po stlačení tlačidla sa zobrazí konfirmačné okno, ktoré užívateľ musí potvrdiť. Toto okno ošetruje prípadné nechcené kliknutie na nesprávne tlačidlo, teda užívateľ má šancu si túto nenávratnú akciu rozmyslieť. Po potvrdení akcie v konfirmačnom okne sa z databáze vymaže záznam triedy, všetky záznamy študentov, ktorí sú na triedu naviazaní a tiež záznamy dochádzky tejto triedy. Užívateľovi na konci procesu mazania vyskočí okno s informáciou o úspešnom vymazaní triedy z databázy. Zobrazenie detailu triedy sa zatvorí a užívateľ je presmerovaný naspäť na *Dashboard*, konkrétne na záložku *My Classes*, ktorej zobrazovaný zoznam tried je aktualizovaný. Mazanie predmetu zabezpečuje ViewModel vybranej triedy „SelectedClassViewModel.cs“. Nie je potrebný samostatný View alebo ViewModel, pretože pri tejto akcii sa užívateľovi nič, okrem konfirmačného okna, nezobrazuje.

5.6.3 Pridanie študentov pomocou CSV

Po vytvorení predmetu mu nie sú pridelení žiadni študenti. Pre zjednodušenie pridávania študentov som do mojej aplikácie pridal spôsob pridávania študentov pomocou importu zo súboru formátu CSV, ktorý si učiteľ nechal vygenerovať informačným systémom STAG. CSV (*Comma-separated values*) je súbor obsahujúci dáta, ktoré sú oddelené oddeľovacím znakom – najčastejšie čiarkou, bodkočiarkou. Každý učiteľ si môže pre každý vyučovaný predmet pomocou informačného systému STAG vygenerovať súbor typu CSV, ktorý obsahuje zoznam všetkých zúčastnených študentov. Užívateľ aplikácie *Absence Manager* si tak pomocou tlačidla *Import from CSV* v zobrazení detailu predmetu môže jednoducho importovať študentov z informačného systému STAG. Po kliknutí na tlačidlo sa ukáže okno s vyhľadávaním súboru formátu CSV.

Ak užívateľ súbor potvrdí a súbor je správneho formátu, vytvoria sa vybranej triede jednotliví študenti a zobrazí sa informačné okno s oznámením o úspešnom vložení študentov. Importovaný nesprávny súbor alebo súbor CSV nesprávneho formátu vyvolá výnimku a chybová hláška sa užívateľovi vypíše v informačnom okne. Vložení študenti pomocou tejto funkcionality nemajú pridelené identifikačné karty. Je teda potrebné, aby užívateľ pridelil každému študentovi identifikačnú kartu tak, že otvorí okno s editáciou študenta, klikne do poľa *ID Card*, vyzve študenta na priloženie identifikačnej karty a informácie uloží. Táto funkcionalita je opäť zabezpečená pomocou ViewModel „SelectedClassViewModel.cs“.

5.6.4 Pridanie študentov ručne

Ďalšia možnosť ako pridať študentov je pridať ich pomocou formuláru k tomu určenému. Po kliknutí na tlačidlo *Add New Student* v zobrazení detailu predmetu, užívateľovi sa zobrazí formulár s nasledujúcimi poľami:

- **Name** – krstné meno študenta.
- **Surname** – priezvisko študenta.
- **Study Year** – rok štúdia študenta.
- **Study Program** – študijný program študenta.
- **ID Card** – číslo čipu identifikačnej RFID karty študenta.

Po vyplnení všetkých položiek a kliknutí na tlačidlo *Create* sa vytvorí nový študent, ktorý navštevuje vybraný predmet. Tiež sa zobrazí oznámenie o úspešnom vytvorení študenta a aktualizuje sa zoznam študentov v detaile predmetu. O zobrazenie a funkcionalitu tejto časti sa starajú súbory „AddNewStudentView.xaml“ a „AddNewStudentViewModel.cs“.

5.6.5 Editácia študenta

Editovať údaje študenta užívateľ môže po kliknutí na tlačidlo *Edit Student* v zozname študentov pri zobrazení detailu predmetu. Formulár je totožný s formulárom slúžiacim na vytvorenie nového študenta, aktuálne údaje daného študenta sú predvyplnené a užívateľ ich môže zmeniť. Tlačidlo v pravom hornom rohu slúži na odstránenie študenta z vybraného predmetu. Po jeho zakliknutí a potvrdení v konfirmačnom okne sa z databázy odstráni študent a všetky jeho prípadné záznamy o dochádzke. Ak užívateľ dokončil editáciu údajov študenta, kliknutím na *Save* údaje uloží. V spodnej časti má užívateľ dodatočnú možnosť vymazať dochádzku editovanému študentovi za vybraný dátum. Užívateľ si vyberie dátum výuky z ponuky a po kliknutí na *Remove* sa záznam o dochádzke odstráni. Súbory zabezpečujúce túto časť aplikácie sú „EditStudentView.xaml“ a „EditStudentViewModel.cs“.

5.7 Zápis dochádzky

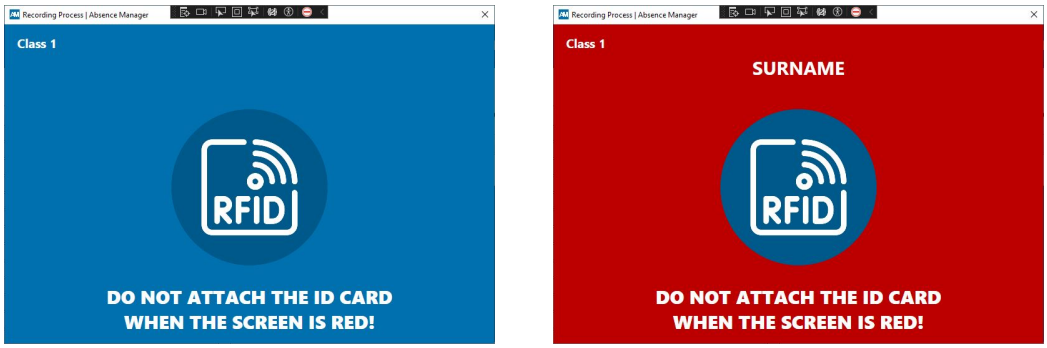
V tejto podkapitole predstavím, akým spôsobom môže učiteľ – teda užívateľ *Absence Manager* manažovať dochádzku študentov do výuky svojho predmetu.

Samotný zápis dochádzky sa vykonáva z okna detailu predmetu. Užívateľ si v pravej časti okna musí zvoliť dátum výuky z roztváracej ponuky, pre ktorý chce zapísať dochádzku. Tieto dátumy sú vypočítané a získané automaticky z vlastností daného predmetu, konkrétne *StartDate* a *Duration*. Po výbere dátumu sa odomkne tlačidlo *START RECORDING*, ktoré zahajuje zápis dochádzky.

Po kliknutí na toto tlačidlo sa otvorí okno zápisu dochádzky (obrázok 14), ktoré ostáva otvorené po celú dobu získavania dochádzky. Toto okno obsahuje nadpis (názov predmetu), dynamicky načítavané priezvisko overeného študenta, ikonu RFID a upozornenie. Upozornenie pripomína používateľom aplikácie, aby študentské karty prikladali postupne a iba vtedy, ak je pozadie okna modré, nie červené. Toto opatrenie zabezpečí správne načasovanie pri prikladaní kariet k čítačke a správne overovanie a zapisovanie dát. Po priložení karty študenta, ktorý navštevuje daný predmet, priezvisko overeného študenta sa zobrazí na červenom pozadí pre konfirmáciu správneho zápisu dochádzky. Ak sa k čítačke priloží karta, ktorá nepatrí žiadnemu študentovi v danom predmete, zobrazí sa text *NOT FOUND*. To indikuje, že zadaná karta nebola nájdená medzi študentmi, ktorí sú priradení danému predmetu. Študenti teda môžu prikladať svoje identifikačné RFID karty a po priložení a overení čísla identifikačnej RFID karty aplikácia zaznamená každého študenta do vlastného záznamu v tabuľke *entry*.

Ak chce učiteľ zaznamenávanie dochádzky ukončiť, okno so zápisom dochádzky zatvorí. Zápis dochádzky je ukončený po zatvorení tohto okna. Proces zápisu dochádzky generujú súbory „RecordingProcessView.xaml“ a „RecordingProcessViewModel.cs“.

V budúcom vývoji aplikácie by bolo vhodné tiež pridať aj záznam o ospravedlnenej dochádzke, napríklad, ak sa študent zúčastní inej školskej akcie a absenciu v tomto predmete má ospravedlnenú. Rozhodol som sa však v rámci bakalárskej práce uchovávať informáciu iba o prítomnosti a absencii.



Obr. 14: Absence Manager - okno zaznamenávania dochádzky.

5.8 Vizualizácia a export dochádzky

Po vyplnení dochádzky sa učiteľovi naskytuje potreba rýchleho a jednoduchého prehľadu, kde by mohol skontrolovať dochádzku svojich študentov. V *Absence Manager* nám na to slúži tlačidlo *Show Attendance*, ktoré otvára okno s tabuľkou študentov a ich dochádzkou (obrázok 15). Obsahuje stĺpce pre priezvisko a krstné meno študenta, percento účasti a samotné stĺpce s dátumami výuky. Jeden riadok predstavuje záznam dochádzky jedného študenta. Dochádzka študentov sa v stĺpcoch s dátumami výuky značí nasledovne:

- **X** – študent sa v tento deň výuky zúčastnil (prítomnosť).
- ***prázdne*** – študent sa v tento deň výuky nezúčastnil (absencia).

Tretí stĺpec predstavuje vypočítané percento s presnosťou na dve desatinné miesta účasti daného študenta na výuke. Toto percento sa počíta podľa vzorca:

$$\% = \left(\frac{\mathrm{p}}{\mathrm{n}}\right) \times 100$$

kde:

p : celkový počet prítomností
n : celkový počet hodín (vlastnosť *class.Duration*)

Užívateľ má aj pri tejto tabuľke dát možnosť zoradiť si dáta. Kliknutím na záhlavie stĺpca si dokáže zoradiť mená abecedne alebo percento účasti vzostupne a zostupne. Tabuľku dochádzky generujú súbory „ShowAttendanceView.xaml“ a „ShowAttendanceViewModel.cs“.

Surname	First Name	%	02/04/2024	09/04/2024	16/04/2024	23/04/2024	30/04/2024
Aaron	Michael	80.00	X	X		X	X
Brown	Ann	60.00	X	X		X	
Cedric	David	40.00		X			X

Obr. 15: Absence Manager - zobrazenie tabuľky dochádzky študentov.

5.8.1 Export vo formáte CSV

Užívateľ má tiež možnosť si záznamy o dochádzke exportovať do súboru formátu CSV. Okrem údajov, ktoré sa nachádzajú aj v tabuľke dochádzky je navyše tento súbor rozšírený o stĺpec *ročník* a *program*, aby mal vyučujúci o svojich študentoch tie najdôležitejšie informácie pokope. Označenie pre prítomnosť a absenciu v hodine je rovnaké, ako v tabuľke dochádzky. Klikom na tlačidlo *Export Attendance CSV* sa užívateľovi otvorí prehliadač súborov a užívateľ si zvolí názov súboru a adresár, kde ho uloží. Po úspešnom uložení sa užívateľovi zobrazí informačné okno so správou, že údaje boli úspešne exportované do súboru formátu CSV. Tento export implementuje ViewModel „SelectedClassViewModel.cs“.

5.9 Validácia zadaných údajov

Programátori a vývojári by mali validovať dáta prevzaté od užívateľa z viacerých dôvodov. Jedným z hlavných dôvodov je zabezpečenie platnosti dát v aplikácii. Užívateľ má možnosť vložiť do textového poľa akékoľvek údaje, a to bez ohľadu na ich správnosť alebo formát. Bez riadnej validácie môžu byť tieto údaje nesprávne alebo dokonca nebezpečné pre aplikáciu. Napríklad, ak aplikácia očakáva číslo, ale užívateľ zadá textový reťazec, môže to viesť k neočakávanému správaniu alebo chybám v aplikácii. Okrem toho môže užívateľ vložiť do textového poľa veľmi dlhý reťazec, čo môže spôsobiť pretečenie pamäte alebo iné problémy so spracovaním údajov. Preto je dôležité, aby programátori vykonávali dôkladnú validáciu vstupných údajov od užívateľa a zabezpečili tak správne a bezpečné fungovanie aplikácie. V *Absence Manager* si užívateľ môže skontrolovať požadovaný tvar zadávaného vstupu jednoduchým podržaním kurzora na danom poli užívateľského mena, hesla alebo iných údajov, kedy sa mu objaví takzvaný *hint* text obsahujúci nápovedu.

5.9.1 Tvar užívateľského mena a hesla

Pre zovšeobecnenie a uľahčenie práce s užívateľskými dátami som ustanovil spôsob, akým by mal užívateľ voliť tvar mena a hesla. Tieto obmedzenia pomáhajú udržať aspoň do určitej miery jednotný tvar zadávaných údajov a slúžia tiež na kontrolu, či sa daná akcia (v tomto prípade registrácia nového užívateľa) môže alebo nemôže vykonať. Tieto požiadavky sú:

- **Username** – užívateľské meno musí byť dĺžky rovnej alebo väčšej ako 4 znaky.
- **Password** – užívateľské heslo musí byť dĺžky rovnej alebo väčšej ako 8 znakov, musí obsahovať aspoň jedno veľké písmeno, aspoň jedno malé písmeno a aspoň jednu číslicu.

5.9.2 Tvar ostatných vstupných hodnôt

Tvar vstupu od užívateľa na ostatných poliach nie je potrebné kontrolovať až tak podrobne ako je to pri užívateľskom mene a hesle. Pri vstupoch ako je názov predmetu, skratka predmetu, popis predmetu, meno a priezvisko študenta, ročník študenta či číslo identifikačnej RFID karty fyzicky obmedzujem počet znakov, koľko užívateľ môže napísať do daného poľa. Toto riešenie je pre ostatné, nekritické vstupné údaje dostatočné a poskytuje užívateľovi maximálne prispôsobenie svojim potrebám.

6 Výsledky

V tejto poslednej kapitole zhodnotím kvalitu môjho riešenia, ako funguje v porovnaní s ostatnými existujúcimi riešeniami, aké problémy existujúcich riešení rieši a ako by sa môj projekt mohol dať v budúcnosti rozšíriť.

6.1 Kvalita riešenia

Aplikácia, ktorá je výsledkom mojej bakalárskej práce, rieši problémy, ktoré som opísal pri existujúcich riešeniach zápisu dochádzky.

- **Dáta uložené v elektronickej forme**: Dáta sa nespoliehajú na fyzickú kópiu vytlačenú na dochádzkovom hárku, pri ktorom môže dôjsť k znehodnoteniu údajov alebo k strate. Namiesto toho sú dáta uložené v elektronickej podobe a užívateľ si ich môže zálohovať na viacerých miestach pre prípad zlyhania jeho zariadenia.
- **Bezpečnosť prihlásenia**: Po prihlásení užívateľským menom a heslom je užívateľ vyzvaný, aby priložil svoju univerzitnú RFID kartu k čítaciemu zariadeniu. Tým eliminujem problém, ktorý by nastal, ak by prihlasovacie údaje získala tretia osoba. V aplikácii síce prejde riadnym prihlasovacím procesom užívateľským menom a heslom, no ďalej sa nedostane, nakoľko nemá prístup k zamestnaneckej RFID identifikačnej karte vyučujúceho.
- **Ochrana osobných údajov**: Osobné údaje študentov a záznamy o dochádzke nie sú prístupné iným osobám okrem vyučujúceho, ktorý má ako jediný prístup k záznamom o dochádzke a zoznamu študentov.
- **Sťažený neoprávnený zápis dochádzky**: Zaznamenanie dochádzky pomocou identifikačnej karty RFID študenta je jednoznačné a rýchle. Ak učiteľ zabezpečí, že každý študent (napríklad pri odchode z výuky) priloží RFID kartu k čítačke iba jedenkrát, neoprávnený zápis dochádzky nie je možný. Ak sú k čítaciemu zariadeniu RFID priložené dve a viac identifikačných kariet, čítacie zariadenie RFID je schopné pri jednom priložení prečítať najviac jednu identifikačnú kartu.
- **Prenositeľnosť a prispôsobenie**: Aplikáciu som navrhol a zhotovil s dôrazom na jej prenositeľnosť použitia v rôznom prostredí a taktiež jednoduché prispôsobenie, teda zmeny uchovávaných údajov, dizajnu aplikácie a iných prvkov.

Moja aplikácia funguje *offline*, teda nie je závislá od internetového pripojenia. Vyučujúci môže zapísať dochádzku aj bez prístupu na internet, teda v prípade výpadku univerzitného internetového pripojenia alebo v prípade výuky mimo školy. Taktiež by som vyzdvihol jej jednoduchosť používania a integráciu so súbormi formátu CSV. Vyučujúci tak nemusí pridávať veľké množstvo študentov, ktorí navštevujú jeho predmet ručne, ale pomocou súboru formátu CSV, ktorý si vygeneruje v informačnom systéme STAG.

6.2 Potenciálne rozšírenia aplikácie

Tento projekt je konceptom, na ktorom v rámci bakalárskej práce prezentujem svoju schopnosť navrhnúť a vyvinúť funkčnú aplikáciu. Ak by sa aplikácia *Absence Manager* mala v budúcnosti zaviesť do reálnej prevádzky, dala by sa rozšíriť o radu ďalších funkcií. Keďže som môj projekt viedol v architektúre typu *Model-View-ViewModel*, každé rozšírenie a úprava funkcionality sú jednoduché. Medzi rozšírenia, ktoré by pomohli viac efektívne spravovať dochádzku študentov by som zaradil:

- **Frekvencia výuky**: Momentálne *Absence Manager* pracuje so sedemdňovou frekvenciou výuky daného predmetu. Niektoré predmety sa však vyučujú v inej frekvencii, napríklad párny–nepárny týždeň, každý druhý týždeň, či predmety rozdelené na dve a viac zložiek (napríklad *prednáška* a *cvičenie*), ktoré spravidla nemajú výuku v jeden deň. Budúce rozšírenie aplikácie by tak mohlo brať do úvahy inú frekvenciu výuky ako sedem dní.

- **Grafy a ďalšie prehľady**: *Absence Manager* by mohla zobrazovať grafy o dochádzke študentov počas celého semestra, dochádzku študentov v rámci všetkých vyučovaných predmetov užívateľa, alebo iné prehľady, ktoré by uľahčili vyučujúcemu podmieniť získanie zápočtu alebo prístup k záverečnému testu určitým stavom dochádzky.

- **Prístup viacerých užívateľov k jednému predmetu**: Toto rozšírenie by zjednodušilo správu dochádzky predmetom, ktoré sa skladajú z dvoch a viac zložiek výuky (napríklad *prednáška* a *cvičenie*). Obaja vyučujúci by získali prístup k rovnakým dátam o dochádzke svojich študentov a samotný predmet by mohol byť tiež rozdelený na dve a viac zložiek (podľa predmetu).

- **Online integrácia so systémom STAG a zálohovanie**: *Absence Manager* by sa dala upraviť aj spôsobom, aby komunikovala s rozhraním informačného systému STAG API. To by jej umožnilo ešte jednoduchšie pridávanie študentov a predmetov, ktoré vyučujúci vedie. Užívateľ by si tak jedným kliknutím mohol pridať všetky predmety a všetkých študentov, ktorí tieto predmety navštevujú. Týmto by aplikácia síce prišla o svoju prenositeľnosť a výhodu fungovania v *offline* prostredí, jednoduchosť a priamočiarosť importu údajov by však túto stratu vykompenzovali. Zároveň by sa mohla využívať *online* záloha dát, kedy by pri zlyhaní pevného disku alebo nechceným odstránením databázového súboru mohli byť zálohované dáta obnovené z internetu.

Záver

Výsledkom mojej bakalárskej práce je plne funkčná okenná aplikácia *Absence Manager* pre operačný systém *Windows*, ktorá umožňuje efektívne, jednoznačne a prehľadne zapisovať a spravovať dochádzku študentov na vyučovacie hodiny. Poskytuje intuitívne používateľské rozhranie, ktoré prispieva k jednoduchej orientácii v aplikácii.

Bezpečnosť prihlasovacieho procesu zabezpečuje správne nakladanie s užívateľským heslom ako citlivým údajom a dvojfaktorové prihlasovanie s overením zamestnaneckej RFID karty vyučujúceho.

Vyučujúci si po prihlásení môže vytvoriť vlastný vyučovaný predmet s jeho popisom, pridať do neho študentov a efektívne spravovať dochádzku. Pridať množstvo študentov môže aj pomocou importu súboru formátu CSV, ktorý si nechal vygenerovať v informačnom systéme STAG. Toto zabezpečí, že vyučujúci nemusí pridávať veľké množstvo študentov manuálne, jedného po jednom.

Pre záznam dochádzky aplikácia *Absence Manager* využíva technológiu RFID, ktorá jednoznačne a rýchlo zapíše dochádzku danému študentovi. *Absence Manager* tiež poskytuje prehľadný náhľad dochádzky v tabuľkovej forme spolu s vypočítanými percentami účasti počas celého obdobia výuky. Tento náhľad si vyučujúci môže exportovať do súboru formátu CSV, čo uľahčuje archiváciu a uchovávanie dát pre budúce použitie a kontrolu.

Absence Manager rieši viaceré problémy existujúcich riešení zaznamenávania dochádzky ako sú neoprávnený zápis dochádzky treťou osobou, uchovávanie dát v elektronickej podobe namiesto fyzického dochádzkového hárku a v neposlednom rade zamedzenie zdieľania osobných údajov a údajov o dochádzke medzi účastníkmi výuky. Pri vývoji aplikácie som tiež dával dôraz na jej prenositeľnosť, aby sa jej použitie dalo prispôsobiť v akomkoľvek prostredí. Moje riešenie pracuje v *offline* prostredí, ktoré je nezávislé na dostupnosti pripojenia na internet a vyučujúci dochádzku dokáže zapisovať a spravovať aj bez neho.

Conclusions

The outcome of my bachelor thesis is a fully functional window application *Absence Manager* for the *Windows* operating system, which allows to efficiently, unambiguously and clearly record and manage students' attendance for classes. It provides an intuitive user interface that contributes to easy navigation of the application.

The security of the login process is ensured by proper handling of the user password as sensitive information and two-factor login with authentication of the teacher's employee RFID card.

After logging in, the teacher can create his own taught subject with its description, add students to it and manage attendance efficiently. They can also add a number of students by importing a CSV format file that they have had generated in the STAG information system. This will ensure that the teacher does not have to add a large number of students manually, one by one.

To record attendance, the *Absence Manager* application uses RFID technology to unambiguously and quickly record the attendance of a given student. *Absence Manager* also provides a clear view of attendance in tabular form, along with calculated attendance percentages throughout the course period. This view can be exported to a CSV file by the teacher, making it easy to archive and store the data for future use and review.

Absence Manager solves several problems of existing attendance recording solutions such as unauthorized recording of attendance by a third party, storing data electronically instead of a physical attendance sheet, and last but not least, preventing the sharing of personal data and attendance data between attendees. When developing the application, I also placed emphasis on its portability so that its use could be adapted to any environment. My solution works in an *offline* environment, which is independent of the availability of an internet connection and the teacher can record and manage attendance without it.

A Čítačka RFID

Pri vývoji *Absence Manager* som používal bezkontaktné RFID čítacie zariadenie. Čipy RFID identifikačných kariet načítava rýchlo a z väčšej vzdialenosti. Pre jej chod nie je potrebný žiadny ovládač, funguje na princípe *Plug-and-Play*. To znamená, že po pripojení je funkčná okamžite. K počítaču sa pripája prostredníctvom rozhrania USB ako virtuálna klávesnica. Identifikačné číslo čipu RFID karty prečíta a vypíše ho do aktívneho textového poľa vo forme znakov kláves 0 až 9 v hlavnej časti klávesnice. Jej výstup závisí od zvoleného systémového jazyka klávesnice, takže užívateľ musí byť konzistentný s používaním rovnakého jazyka pri inicializácií dát ako aj pri ich správe neskôr. Internetový obchod dratek.cz poskytuje neoficiálny konfiguračný softvér na stiahnutie, ktorý umožňuje nastaviť formát výstupných dát. Tento softvér umožňuje nastaviť kódovanie prečítaných údajov v desiatkovej alebo šestnástkovej číselnej sústave, vybrať si pridanie znaku na konci čítania, napríklad čiarku, bodkočiarku alebo klávesu *Enter*. Zmena týchto nastavení čítacích vlastností v konfiguračnom nástroji sa do čítačky uloží a po odpojení zariadenia si čítačka nastavenie pamätá aj po opätovnom pripojení. To umožňuje nie len jednoduchú prácu s čítačkou v rámci môjho projektu, ale aj pri zápise dát do súboru formátu CSV a iné. [21]

- **Hmotnosť**: 100 g
- **Rozmery**: 104 x 70 x 11 mm
- **Napájanie**: USB
- **Funkčná vzdialenosť**: až 8 cm
- **Frekvencia**: 13,56 MHz
- **Indikátory**: svetelný LED indikátor, zvukový indikátor



Obr. 16: Čítacie zariadenie RFID použité počas vývoja aplikácie.

B Obsah elektronických dát

Text práce, zdrojové kódy a všetky ostatné prílohy, ktoré sú súčasťou bakalárskej práce som riadne odovzdal prostredníctvom elektronického systému katedry informatiky.

`install/`
: Adresár so súbormi potrebnými na korektnú inštaláciu aplikácie *Absence Manager*.

`source/`
: Adresár, v ktorom sú umiestnené zdrojové kódy aplikácie *Absence Manager*.

`thesis/`
: Adresár obsahujúci text bakalárskej práce vo formáte PDF spolu so .zip archívom, kde sú zabalené zdrojové LaTeX kódy generujúce tento text, obrázky a ďalšie prílohy. Text bakalárskej práce je vytvorený s použitím záväzného štýlu katedry informatiky pre záverečné práce.

`README.md`
: Súbor, ktorý obsahuje inštrukcie pre korektnú inštaláciu aplikácie *Absence Manager*, jej spustenie a použitie.

Literatúra

[1] *Masarykova Univerzita - Centrum informačních technologií. : Moodle: návody pro uživatele - Docházka* [online]. [cit. 2024-2-4]. Dostupné z: ⟨https://moodledocs.phil.muni.cz/nestandardni-moduly/dochazka/⟩.

[2] *Microsoft Learn. : A tour of the C# language* [online]. [cit. 2024-2-4]. Dostupné z: ⟨https://learn.microsoft.com/sk-sk/dotnet/csharp/tour-of-csharp/⟩.

[3] *Microsoft Learn. : Úvod do technologie .NET* [online]. [cit. 2024-2-4]. Dostupné z: ⟨https://learn.microsoft.com/cs-cz/dotnet/core/introduction⟩.

[4] *Microsoft Learn. : Introducing Windows Presentation Foundation* [online]. [cit. 2024-2-4]. Dostupné z: ⟨https://learn.microsoft.com/en-us/previous-versions/dotnet/articles/aa663364(v=msdn.10)⟩.

[5] *Microsoft Learn. : Přehled XAML (WPF .NET)* [online]. [cit. 2024-2-4]. Dostupné z: ⟨https://learn.microsoft.com/cs-cz/dotnet/desktop/wpf/xaml/?view=netdesktop-8.0⟩.

[6] *Microsoft Learn. : Entity Framework documentation hub* [online]. [cit. 2024-2-4]. Dostupné z: ⟨https://learn.microsoft.com/en-us/ef/⟩.

[7] *Amazon Web Services. : What is SQL (Structured Query Language)?* [online]. [cit. 2024-2-4]. Dostupné z: ⟨https://aws.amazon.com/what-is/sql/⟩.

[8] *Simplilearn. : What is SQLite?* [online]. [cit. 2024-2-4]. Dostupné z: ⟨https://www.simplilearn.com/tutorials/sql-tutorial/what-is-sqlite⟩.

[9] *NuGet Gallery. : BCrypt.Net* [online]. [cit. 2024-2-4]. Dostupné z: ⟨https://www.nuget.org/packages/BCrypt.Net⟩.

[10] *Niels Provos, David Mazières. : Proceedings of the FREENIX Track: 1999 USENIX Annual Technical Conference: A Future-Adaptable Password Scheme* [online]. [cit. 2024-2-4]. Dostupné z: ⟨https://www.usenix.org/legacy/events/usenix99/provos/provos.pdf⟩.

[11] *NuGet Gallery. : Microsoft.EntityFrameworkCore.Sqlite* [online]. [cit. 2024-2-4]. Dostupné z: ⟨https://www.nuget.org/packages/Microsoft.EntityFrameworkCore.Sqlite⟩.

[12] *NuGet Gallery. : Microsoft.EntityFrameworkCore.Tools* [online]. [cit. 2024-2-4]. Dostupné z: ⟨https://www.nuget.org/packages/Microsoft.EntityFrameworkCore.Tools⟩.

[13] *Josh Close Github. : CsvHelper*. Dostupné z: ⟨https://joshclose.github.io/CsvHelper/⟩.

[14] *NuGet Gallery. : System.Text.Encoding.CodePages* [online]. [cit. 2024-2-15]. Dostupné z: ⟨https://www.nuget.org/packages/System.Text.Encoding.CodePages/8.0.0⟩.

[15] *smart-TEC. : Technologie RFID* [online]. [cit. 2024-2-4]. Dostupné z: ⟨https://www.smart-tec.com/cs/auto-id-svet/technologie-rfid⟩.

[16] *Wikimedia Commons. : RFID Chip 003.jpg* [online]. [cit. 2024-2-15]. Dostupné z: ⟨https://commons.wikimedia.org/wiki/File:RFID_Chip_003.JPG⟩.

[17] *AdaFruit Blog. : RFID MiFare S50 Card Reader* [online]. [cit. 2024-2-15]. Dostupné z: ⟨https://blog.adafruit.com/2012/07/13/new-product-rfid-mifare-s50-card-reader-ps2-interface/⟩.

[18] *Microsoft Learn. : Model-View-ViewModel (MVVM)* [online]. [cit. 2024-2-16]. Dostupné z: ⟨https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm⟩.

[19] *LavaDESIGN. : Hash* [online]. [cit. 2024-4-21]. Dostupné z: ⟨https://www.lavadesign.sk/hash/⟩.

[20] Ferek, Matúš. *Bezpečná komunikace mezi data loggerem a databazovým serverem* [online]. Brno: Vysoké učení technické v Brně, 2011 [cit. 2024-4-20]. 82 str. Dostupné z: ⟨https://www.vut.cz/www_base/zav_prace_soubor_verejne.php?file_id=39851⟩.

[21] *Drátek.cz. : USB RFID čtečka 13,56MHz* [online]. [cit. 2024-4-20]. Dostupné z: ⟨https://dratek.cz/arduino/1924-usb-rfid-ctecka-13.56mhz.html⟩.