UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

BAKALÁŘSKÁ PRÁCE

2023 Michal Bržezický

UNIVERZITA PARDUBICE

Fakulta elektrotechniky a informatiky

Aplikace pro plánování směn

Michal Bržezický

Bakalářská práce

2023

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2022/2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Michal Bržezický**
Osobní číslo:	**I20254**
Studijní program:	**B0688A140009 Informační technologie**
Téma práce:	**Aplikace pro plánování směn**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování

Cílem bakalářské práce je vytvořit desktopovou aplikaci pro plánování směn. Aplikace bude naprogramována v jazyku C# a pomocí technologie WPF. Aplikace umožní generovat sestavy směn pro následující měsíc na základě zadaných požadavků konkrétních zaměstnanců a na základě dat z předchozích měsíců. V rámci aplikace bude brán ohled na čerpání dovolené a rovnoměrné rozdělení směn zaměstnanců o svátcích a víkendech.

Rozsah pracovní zprávy: **min. 30 stran**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

Troelsen, Andrew W. C# 6.0 and the .NET 4.6 Framework. Apress, 2016. ISBN 978-1484213339.

Petzold, Charles. Mistrovství ve Windows Presentation Foundation : [aplikace = kód + markup]. Brno: Computer Press, 2008. ISBN 978-80-251-2141-2.

Vedoucí bakalářské práce: **Ing. Jan Panuš, Ph.D.**
Katedra informačních technologií

Datum zadání bakalářské práce: **16. prosince 2022**
Termín odevzdání bakalářské práce: **12. května 2023**

L.S.

Ing. Zdeněk Němec, Ph.D. v.r.
děkan

Ing. Jan Panuš, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 28. února 2023

Prohlašuji:

Práci s názvem Aplikace pro plánování směn jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 7/2019 Pravidla pro odevzdávání, zveřejňování a formální úpravu závěrečných prací, ve znění pozdějších dodatků, bude práce zveřejněna prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 10.05.2023

Michal Bržezický

Poděkování

Rád bych poděkoval vedoucímu mé práce Ing. Janu Panušovi, Ph.D. za jeho ochotu, cenné rady a odborné vedení. Dále bych rád poděkoval za spolupráci staniční sestře ZZS Vysokého Mýta Janě Zamastilové a v neposlední řadě také své rodině za neustálou podporu a motivaci.

ANOTACE

Bakalářská práce se věnuje tvorbě aplikace určené ke generování směn pro jednotlivé týmy posádek zdravotnické záchranné služby ve Vysokém Mýtě. V rámci této práce byl vytvořen nástroj, který zjednodušuje manuální plánování a tvorbu směn dle požadavků zaměstnanců s ohledem na čerpání řádné dovolené a rovnoměrného rozdělení směn o víkendových a svátečních dnech. Aplikace uchovává informace z předchozích měsíců, které využívá k vytváření směn. V určitém týmu posádky je také znázorněno vozidlo, které posádka v dané směně využívá.

KLÍČOVÁ SLOVA

WPF, generace směn, C#, LiteDB, ZZS, rovnoměrné rozplánování směn, plánovač směn

TITLE

Application for shift sheduling

ANNOTATION

The bachelor thesis is focused on the development of an application designed for generating shifts for individual crews of the emergency medical services in Vysoké Mýto. As part of this work, a tool was created that simplifies manual planning and shift creation according to employees' requirements, taking into account regular vacation time and an even distribution of shifts on weekends and holidays. The application stores information from previous months, which it uses to create shifts. In a specific crew team, the vehicle that the crew uses during a particular shift is also displayed.

KEYWORDS

WPF, shift generation, C#, LiteDB, EMS, Even shift distribution, shift sheduler

Obsah

Seznam ilustrací a tabulek

Seznam značek a zkratek

IDE	Integrated Development Environment
WPF	Windows Presentation Foundation
ZZS	Záchranná zdravotnická služba
DB	Databáze
NoSQL	Non Structured Query Language
XAML	Extensibile Application Markup Language
XLSX	Microsoft Excel Spreadsheet
UX	User Experience
UI	User Interface
MVVM	Model-View-ViewModel
RZP	Posádka rychlé zdravotnické pomoci
RV	Posádka Rendez-vous
DI	Dependency Injection
OS	Operační systém
MDIX	MaterialDesignInXaml

Terminologie

UX User Experience, zkráceně UX, popisuje uživatelskou zkušenost s aplikací. Týká se interakce mezi uživatelem a aplikací a zaměřuje se na vytváření přívětivého a intuitivního řešení. Cílem UX je zohlednit potřeby uživatele a reagovat podle očekávání. Minimalizuje pocit frustrace a snaží se tak vytvořit co nejlepší zážitek aplikace.

UI Uživatelské rozhraní, zkráceně UI, se týká vzhledu a způsobu, jakým uživatel interaguje s aplikací. UI design se zaměřuje na vizuální prvky, jako jsou barvy, typografie a uspořádání prvků, aby bylo co nejpříjemnější a zároveň nejefektivnější pro práci s aplikací.

NoSQL NoSQL je termín používaný k popisu široké kategorie databází, které se od tradičních relačních databází liší způsobem, jakým ukládají a přistupují k datům. Relační databáze ukládají data do tabulek na základě předem definovaných vztahů mezi nimi, zatímco NoSQL databáze využívají různé modely pro organizaci a ukládání dat, jako jsou například grafové nebo dokumentové přístupy.

Úvod

Bakalářská práce se věnuje tvorbě aplikace, která má usnadnit proces plánování směn pro staniční sestru zdravotnické záchranné služby (ZZS). Tento proces plánování byl dosud prováděn manuálně a zahrnoval časové vytížení s řadou komplikací, které se týkaly především požadavků jednotlivých zaměstnanců.

Vzhledem k příplatkům nočních směn, o svátcích nebo víkendech bylo nutné směny rovnoměrně rozplánovat, aby mzda každého zaměstnance byla přiměřeně stejná dle určité pracovní pozice.

V rámci této bakalářské práce byl vytvořen nástroj, který bude sloužit ke generování směn pro zaměstnance zdravotnické záchranné služby Vysoké Mýto. Aplikace je navržena tak, aby umožňovala snadné, intuitivní a rychlé plánování směn, které budou respektovat požadavky jednotlivých zaměstnanců, jejich řádnou dovolenou a rovnoměrné rozdělení zmíněných příplatkových směn.

Algoritmus generování zohledňuje informace z předchozích měsíců a plánuje v rozumném pořadí noční a denní služby. Součástí aplikace je také možnost uložení a editace základních informací jednotlivých pracovníků. Dále aplikace poskytuje informace o dostupných vozidlech, což umožňuje plánování směn s ohledem na jejich stav.

Cílem této bakalářské práce je tedy vytvoření nástroje, který výrazně zjednoduší práci staniční sestře ZZS, která plánuje směny ve svém volném čase.

1 Organizace ZZS Vysoké Mýto

Záchranná služba Vysokého Mýta je jednou ze dvanácti poboček ZZS Pardubického kraje, která je nepostradatelnou složkou, zajišťující přednemocniční neodkladnou péči obyvatelům kraje. [1]

Od roku 2021 disponuje novou základnou, která umožňuje efektivnější poskytování zdravotnických služeb. Základna je kompletně vybavena moderní zdravotnickou technikou a umožňuje rychlé vyslání týmu na místa událostí. [2]

1.1 Posádky ZZS

ZZS Vysoké Mýto se skládá ze dvou posádek s moderními vozidly. Tyto vozy jsou plně vybaveny veškerou potřebnou zdravotnickou technikou podle nejnovějších předpisů a vyhlášek, aby v případech akutních zdravotních stavů zajistily rychlou a efektivní pomoc při nehodách a jiných mimořádných událostech. [3]

1.1.1 Posádka Rendez-vous (RV)

Výjezdová posádka RV rychlého nasazení poskytuje přednemocniční neodkladnou péči. Tato posádka využívá osobní speciální sanitní vozidlo, které je vybaveno veškerým vybavením klasických sanitních vozů, pouze bez transportních nosítek. RV posádka se skládá z řidiče-záchranáře ze ZZS a nasmlouvaného doktora.

1.1.2 Posádka rychlé zdravotnické pomoci (RZP)

Výjezdová skupina RZP je dvoučlenný tým, jehož součástí je řidič záchranář sanitního vozidla a zdravotnický záchranář. Zdravotnický záchranář má obvykle vyšší odborné vzdělání než řidič záchranář.

1.2 Směny

Záchranná služba poskytuje nepřetržitou 24hodinovou péči, kdy se pracovní den této služby skládá ze dvou 12hodinových směn – směna denní a noční. Každá směna obsahuje jednu posádku RV a dvě posádky RZP. Tímto způsobem je zajištěno, že jsou v každou dobu k dispozici tři týmy připravené poskytnout pomoc v případě mimořádných událostí a zdravotních potíží.

2 Analýza požadavků

Tato kapitola se věnuje popisu funkčních a nefunkčních požadavků, na kterých jsme se dohodli s paní Zamastilovou, staniční sestrou ZZS, během našich schůzek. Následující uvedené požadavky jsou klíčové pro správné fungování aplikace a plánování směn.

Vzhledem k tomu, že aplikace je vytvářená pro konkrétní subjekt s ohledem na specifické požadavky, práce se nezabývá analýzou obdobných řešení.

2.1 Funkční požadavky

Správa zaměstnanců

Aplikace umožní přidávat nové zaměstnance, editovat a mazat stávající. Ke každému zaměstnanci bude možné přiřadit jméno a příjmení a jednu pracovní pozici.

Správa vozidel

V sekci vozidel bude podobně jako u správy zaměstnanců umožněno přidávat, upravovat nebo mazat vozidla. U vozidla bude možné zadat jeho název, zkrácený název pro zobrazení v exportovaném dokumentu směn a také stav vozidla, který se bude upravovat podle častých poruch sanitních vozů.

Vyhledávání

Pro rychlejší práci v jednotlivých sekcích aplikačních entit bude dostupné vyhledávací pole, dle kterého se automaticky po zadání textu vyfiltruje přehled položek.

Pravidla plánování směn

Aplikace bude využívat algoritmus ke generování směn pro následující měsíc, který bude brát v úvahu čerpání dovolené, požadavky a přesuny hodin z předchozího měsíce. Tento algoritmus bude dbát na rozumné rozdělení počtu směn pro svátky, víkendy, noční a denní s maximálním rozdílem o tři směny pro každého uživatele.

Algoritmus počítá vždy s 25 zaměstnanci na plný pracovní úvazek, z toho deset lidí budou záchranáři a 15 zaměstnanců budou řidiči. Ke generaci je potřeba mít nejméně dvě pojízdná vozidla pro posádky RZP. Při nesplnění těchto podmínek, nebude možné pracovní plán generovat.

Aplikace bude navržena s ohledem na lidské faktory, aby se minimalizovaly negativní dopady na zdraví a výkon zaměstnanců. Aby bylo dosaženo tohoto cíle, aplikace bude plánovat směny s dostatečným časovým rozestupem tak, že denní směny nebudou naplánovány ihned za nočními směnami a budou se vylučovat případy, kdy by se směny navzájem překrývaly v jednom dnu.

Každý rok v lednu začátkem nového kalendářního roku dochází k resetování směn. To znamená, že všechny předchozí směny a informace mezi měsíci jsou zrušeny a proces generování nových směn začíná od začátku, s čerstvými informacemi a údaji pro nový rok.

Export směn ve formátu XLSX

Kvůli zvyklostem s programem Excel, ve kterých se původně plánovali směny, bude pomocí aplikace umožněno exportovat směny určitého měsíce do formátu XLSX.

Export dle pracovní pozice

Aplikace bude nabízet možnost exportovat směny podle pracovní pozice zvlášť od ostatních přímo na pracovní plochu počítače.

2.2 Nefunkční požadavky

Spolehlivost aplikace

Aby byla aplikace považována za spolehlivou, musí být řádně ošetřena proti výjimkám a neočekávaným událostem, jako jsou například chybné vstupy od uživatelů nebo nepovolené interakce s UI. Je důležité zajistit, aby aplikace poskytovala plynulý a bezproblémový běh bez výpadku, aby se minimalizovaly negativní dopady při generování, ukládání a exportování dat.

Součástí spolehlivosti aplikace je také schopnost informovat uživatele o výskytu chyb a o způsobu jejich řešení. Tato informace by měla být srozumitelná a uživatelsky přívětivá, aby uživatel mohl snadno odhalit a odstranit problémy, které by mohly ovlivnit chod aplikace.

Příjemné a intuitivní UI

Aplikace bude uživatelsky přívětivá a jednoduchá k používání. Na úvodní stránce budou záložky uživatelů, vozidel a vygenerovaných směn, které zobrazí přehled a patřičné akce položek. Aplikace bude smysluplně rozdělená, jednotně nastylována a musí poskytovat intuitivní řešení pro uživatelské ovládání.

Vzhled exportovaných dokumentů

Exportovaný soubor bude obsahovat informace o konkrétním měsíci a posádce, včetně celkového počtu hodin podle zákoníku práce za daný měsíc.

V levé části tabulky směn bude uvedeno příjmení zaměstnance. Následovat bude výčet jednotlivých dnů v daném měsíci v hlavičkách tabulky. Pod hlavičkami bude uvedeno, zda se jedná o noční směnu (N) nebo denní směnu (D). V případě RZP posádek bude k uvedeným směnám označeno vozidlo, které posádka využívá během své služby. V pravé části tabulky bude uveden souhrn dat ke každému zaměstnanci, včetně poměru mezi D/N směnami, celkového odpracovaného počtu hodin v daném měsíci a přesunu hodin do dalšího pracovního měsíce. Jednotlivé dny budou vizuálně odlišeny – svátek bude zvýrazněn tmavě oranžovou barvou, víkend světle oranžovou a běžné pracovní dny budou bez barvy.

Možnost rozšíření

V rámci návrhu aplikace je nutné zohlednit možnost budoucího rozšíření a vylepšení. Mezi potenciální rozšíření patří například implementace dalších modulů, rozšíření nabízených funkcionalit, nebo optimalizace stávajícího plánovacího algoritmu.

3 Použité technologie

Tato kapitola je věnována použitým technologiím a jejich využitím v aplikaci pro plánování směn. Hlavním jazykem pro implementaci desktopové aplikace je použit jazyk C# a pro tvorbu uživatelského rozhraní technologie Windows Presentation Foundation (WPF), která je určená pro operační systém Windows. Následující zmíněné knihovny a balíčky jsou nainstalovány pomocí správce balíčků NuGet pro platformu .NET.

3.1 Jazyk C#

Jazyk C# je vysokoúrovňový programovací jazyk, který byl vyvinut společností Microsoft. Je založen na jazycích C++ a Java, a představuje tak nepřímého potomka nízko-úrovňového jazyka C. Jedná se o moderní, jednoduchý, objektově orientovaný jazyk, který umožňuje tvorbu aplikací webových technologií, mobilních zařízení a desktopových aplikací. [4] V tomto řešení se používá nejnovější verze jazyka C# 10 společně s frameworkem .NET 6 pro tvorbu desktopové aplikace mířené na operační systém (OS) Windows.

3.2 .NET 6

Framework .NET 6 od společnosti Microsoft přináší vylepšený výkon, který umožňuje rychlejší a výkonnější běh aplikace. Nabízí nové funkce pro Windows Presentation Foundation technologii. To zajišťuje vylepšené vykreslování a rychlejší načítání aplikace. Zahrnuje také validaci NuGet a tím zajišťuje konzistenci a správné nastavení nainstalovaných balíčků. [5]

3.3 Windows Presentation Foundation (WPF)

WPF technologie je UI framework využívající vektorový renderingový engine[1], který je postaven na využití moderního grafického hardwaru a je nezávislý na rozlišení. WPF také využívá rozšiřitelný značkovací jazyk aplikace (XAML), ovládací prvky, data binding[2], animace, styly a další vizuální položky, které jsou nezbytné pro tvorbu uživatelského rozhraní. [6] Protože je WPF součástí .NET frameworku, byla tato technologie jasnou volbou pro použití s jazykem C# a frameworkem .NET v aplikaci pro plánování směn.

[1] Software, jehož úkolem je převést grafické prvky a data na obrazové pixely obrazovky.
[2] Technika spojení oddělující prezentaci dat od aplikační logiky a datového modelu v kódu.

3.4 Knihovny

Knihovna je jedním ze způsobů organizace a sdílení kódu a funkcí napříč projekty. Knihovna je kolekce, která obvykle obsahuje určité funkce, třídy a metody, které jsou připraveny k použití a mohou být integrovány do kódu projektu. Obvykle jsou napsány v určitém programovacím jazyce a lze je používat nezávisle na jiných knihovnách.

3.4.1 EPPlus

EPPlus je open-source[3] knihovna pro práci se soubory aplikace Excel. Jedná se o snadno použitelnou knihovnu, která umožňuje vytvářet, číst a upravovat soubory aplikace Excel. Poskytované funkce usnadňují vytváření buněk, řádků a sloupců, nastavováním stylů a formátováním dat. Umožňují také snadnou práci s grafy a dalšími vizuálními prvky. Knihovna podporuje všechny verze formátu programu Excel včetně XLS a XLSX. [7]

EPPlus je dobře zdokumentovaná, efektivní a snadno použitelná knihovna pro export a manipulaci s daty v aplikaci Excel pomocí jazyka C#. V rámci vytvořeného nástroje je použita verze 6.2.2.

3.4.2 PublicHoliday

Knihovna PublicHoliday nabízí funkce pro zjištění mezinárodních svátků a pracovních dnů. [8] Níže uvedený kód napsaný v jazyce C# popisuje použití této knihovny ve verzi 2.27.0, pomocí které se vytváří instance CzechRepublicPublicHoliday. Tato instance poskytuje funkci IsPublicHoliday, která vrátí výsledek typu Boolean, zda vstupní datum odpovídá státnímu svátku v České republice. Direktiva using v jazyce C# importuje knihovnu PublicHoliday do konkrétního souboru s metodou IsCelebrationDay.

```
using PublicHoliday;
...
public static bool IsCelebrationDay(int year, int month, int day)
{
        DateTime date = new DateTime(year, month, day);
        var czechHolidays = new CzechRepublicPublicHoliday();
        return czechHolidays.IsPublicHoliday(date);
}
```

[3] Veřejně dostupný kód, který lze rozšiřovat a upravovat. Některé úpravy musí dodržovat určitá pravidla.

3.5 Balíčky

Balíčky jsou dalším způsobem sdílení znovupoužitelného kódu mezi různými projekty. Součástí balíčku jsou knihovny a jiné potřebné soubory k určité funkcionalitě. Balíčky se distribuují většinou prostřednictvím online repositářů.

3.5.1 CommunityToolkit.Mvvm

Balíček CommunityToolkit.Mvvm je moderní a rychlou součástí sady nástrojů .NET Community Toolkit. CommunityToolkit.Mvvm zjednodušuje flexibilitu a práci při psaní kódu aplikace využívající návrhový vzor Model-View-ViewModel (MVVM) vysvětlený v následující kapitole Struktura. Balíček poskytuje například implementaci INotifyPropertyChanged rozhraní pro snadnou propagaci změn v datech, základní implementaci ICommand rozhraní pro snadnou práci s událostmi z uživatelského rozhraní a další užitečné funkce. [9] Řešení aplikace využívá balíček CommunityToolkit.Mvvm ve verzi 8.0.0.

3.5.2 Microsoft.Extensions.DependencyInjection

Balíček Microsoft.Extensions.DependencyInjection je nástroj poskytující prostředky pro správu registrace služeb a jejich následného vkládání do aplikace, aniž by se vývojář musel zabývat vytvářením všech potřebných závislostí projektu. K tomu využívá návrhový vzor Dependency Injection (DI). [10] DI kontejner doplní potřebné instance závislostí do konstruktorů tříd nebo metod jako parametry specifikované vývojářem. Tento způsob dosazování značně zjednodušuje kód a zvyšuje flexibilitu a rozšiřitelnost projektu. Balíček Microsoft.Extensions.DependencyInjection je v rámci aplikace použit ve verzi 7.0.0.

3.5.3 MaterialDesignInXaml (MDIX)

Balíček MDIX umožňuje snadno a efektivně vytvářet a navrhovat moderní uživatelské rozhraní v souladu s designem Material Design od společnosti Google. MDIX disponuje širokou škálou vizuálních prvků, animací a ikon, které jsou charakteristické pro Material Design.

Vývojáři využívající MDIX mohou poměrně rychle a jednoduše přizpůsobit komplexní vizuální řešení požadavkům své aplikace. Mezi nejpoužívanější vizuální prvky patří tlačítka, karty, dialogy a další prvky, které jsou nezbytné pro práci v aplikaci. MDIX poskytuje také předpřipravené šablony a efekty pro příjemnější zážitek uživatele. [11]

Balíček je kompatibilní s rozšířením Microsoft.Extensions.DependencyInjection, což umožňuje snadnou správu služeb a závislostí aplikace. MDIX je open-source a dostupný zdarma a jeho hlavním důvodem popularity je široká a aktivní komunita uživatelů.

3.5.4 LiteDB

LiteDB je jednoduchá, snadno instalovatelná a nenáročná open-source NoSQL databáze navržená pro vývoj aplikací v C# a .NET. Databáze LiteDB je dokumentově orientovaná a ukládá veškeré informace do malých a středně velkých datových sad. Tyto datové sady se většinou ukládají do jediného souboru jako kolekce. [12] LiteDB je ideálním nástrojem pro jednoduché desktopové aplikace, které nevyužívají ke své práci samostatný server, což zjednodušuje vývoj a nasazení aplikace.

4 Struktura aplikace

Kapitola se zabývá použitým návrhovým vzorem Model-View-ViewModel, popisem struktury projektu a popisem datové vrstvy s použitím technologie LiteDB.

4.1 Model-View-ViewModel (MVVM)

MVVM je architektonický návrhový vzor využívaný pro vývoj webových, mobilních a desktopových aplikací. Cílem tohoto vzoru je oddělit uživatelské rozhraní od zbytku aplikace pro snadnější rozšiřitelnost a spravovatelnost projektu. [13] MVVM vzor je založen na následujících třech komponentách, popsaných níže.

4.1.1 Model

Model v MVVM architektuře představuje datový model aplikace, který popisuje jednotlivé entity v systému. Tyto entity jsou reprezentovány pomocí primitivních tříd, které obsahují jejich vlastnosti a chování.

4.1.2 View

Komponenta View se zabývá prezentací dat v uživatelském rozhraní. View je vzhled aplikace, se kterým uživatel interaguje. Zahrnuje okna, dialogy, tlačítka a další prvky a zobrazuje data z Modelu. Uživatelské rozhraní je definováno jazykem XAML.

4.1.3 ViewModel

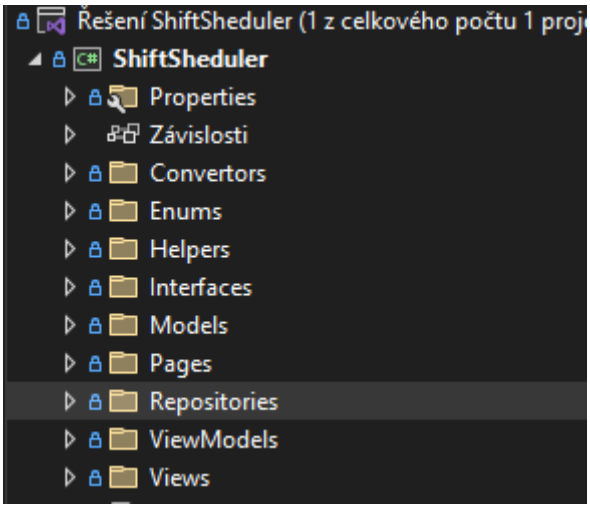
ViewModel je prostředníkem mezi komponenty Model a View, který udržuje stav aplikace. Poskytuje komponentě View přístup k datům z Modelu a zároveň zpracovává uživatelské vstupy a provádí změny. Součástí ViewModel komponenty je také validace vstupů a nastavení viditelnosti vizuálních prvků.

4.2 Struktura projektu

V rámci návrhu a implementace projektu byla brána v potaz architektura MVVM, která rozděluje aplikaci do tří hlavních vrstev – Model, View a ViewModel. Tento architektonický vzor umožňuje oddělit logiku a zobrazování dat, což vede k lepšímu řízení aplikace a dalšímu rozvoji projektu.

Struktura projektu reflektuje tuto architekturu a obsahuje složky pro každou z těchto tří vrstev. Složka Models obsahuje třídy představující aplikační data. Složka Views obsahuje třídy pro zobrazení těchto dat, zatímco složka ViewModel obsahuje třídy sloužící jako rozhraní mezi daty a zobrazovací vrstvou.

Dále jsou obsahem projektu složky pro konvertory, výčty, pomocné třídy, rozhraní, repositáře a stránky, viz obrázek 4. Tyto složky jsou důležité pro organizaci a umožňují oddělení funkcionalit do samostatných bloků, což zvyšuje modularitu a opětovné použití kódu.



Obrázek 1: Struktura projektu

4.3 Dependency injection (DI)

Pro snadnější správu a udržitelnost kódu byl použit návrhový vzor DI pomocí balíčku Microsoft.Extensions.DependencyInjection jak již bylo vysvětleno v kapitole 3. Projekt v rámci DI využívá také návrhový vzor Singleton pro repositáře datových modelů a transientní služby pro třídy ViewModel. Kompnenta ViewModel má krátkou životnost a při každém zobrazení je vhodné vytvořit novou instanci této třídy. [14]

4.3.1 Singleton

Singleton je v uvedeném kontextu služba vytvořená pouze jednou v rámci celé aplikace a poté je sdílena mezi všemi komponentami, které ji využívají. [15] Singleton je přesný opak transientní služby.

4.3.2 Transientní služba

Transientní služba je služba vytvářená každým požadavkem znova. To znamená že pokud je služba vložená do aplikace pomocí DI kontejneru do několika komponent zároveň, bude každá komponenta obsahovat vlastní instanci této služby.

Na obrázku číslo 5 je vidět konfigurace služeb ve třídě App, kde se inicializuje vlastnost Services a pomocí privátní metody ConfigureServices se vytvoří seznam služeb pro aplikaci. Konkrétně se nastaví instance jednotlivých tříd implementujících rozhraní IGenericRepository pro všechny entity aplikace a instance tříd implementujících jednotlivé ViewModels.

```
public partial class App : Application
{
    public IServiceProvider Services { get; }
    public new static App Current => (App)Application.Current;

    public App()
    {
        Services = ConfigureServices();
        this.InitializeComponent();
        GenerateWholeDb();
    }

    private static IServiceProvider ConfigureServices()
    {
        var services = new ServiceCollection();

        // repos as singleton
        services.AddSingleton<IGenericRepository<Employee>, EmployeeRepository>();
        services.AddSingleton<IGenericRepository<Vehicle>, VehicleRepository>();
        services.AddSingleton<IGenericRepository<Holiday>, HolidayRepository>();
        services.AddSingleton<IGenericRepository<MonthPlan>, MonthPlanRepository>();
        services.AddSingleton<IGenericRepository<RvWorkday>, RvWorkdaysRepository>();
        services.AddSingleton<IGenericRepository<RvDriversPlan>, RvDriversPlanRepository>();
        services.AddSingleton<IGenericRepository<EmployeeMonthInfo>, EmployeeMonthInfoRepository>();
        services.AddSingleton<IGenericRepository<RzpDriverWorkday>, RzpDriverWorkdayRepository>();
        services.AddSingleton<IGenericRepository<RzpDriversPlan>, RzpDriversPlanRepository>();
        services.AddSingleton<IGenericRepository<NursesWorkday>, NursesWorkdayRepository>();
        services.AddSingleton<IGenericRepository<NursesPlan>, NursesPlanRepository>();

        // viewModels as transient
        services.AddTransient<MainWindowViewModel>();
        services.AddTransient<EmployeesPageViewModel>();
        services.AddTransient<EmployeeDialogModelView>();
        services.AddTransient<VehiclesPageViewModel>();
        services.AddTransient<VehicleDialogViewModel>();
        services.AddTransient<ShiftsPageViewModel>();
        services.AddTransient<ShiftsDialogViewModel>();
        services.AddTransient<HolidayDialogViewModel>();
        services.AddTransient<RequirementDialogViewModel>();

        return services.BuildServiceProvider();
    }
```

Obrázek 2: Konfigurace služeb aplikace

5 Implementace aplikace

Tato kapitola se zabývá samotné implementaci a designu aplikace. Je zde popsána datová vrstva, která obsahuje potřebná data pro chod aplikace, včetně informací o zaměstnancích, dovolené, směnách, přesunech hodin a dalších informací měsíce. Tato data jsou uložena v prosté LiteDB databázi, která byla speciálně navržena pro potřeby aplikace.

Dále se bude kapitola věnovat uživatelskému rozhraní aplikace, které je snadno ovladatelné a poskytuje uživatelům všechny potřebné funkce pro správu zaměstnanců a následné generování a export směn.

Design aplikace byl vytvořen s použitím moderní technologie MaterialDesignInXAML, která zajišťuje příjemný a profesionální vzhled aplikace. Použité barvy a fonty jsou pečlivě vybrány tak, aby nebyly rušivé. Vytvořená aplikace je snadno použitelná a poskytuje uživateli všechny potřebné funkce pro generování a export směn.

5.1 Datová vrstva

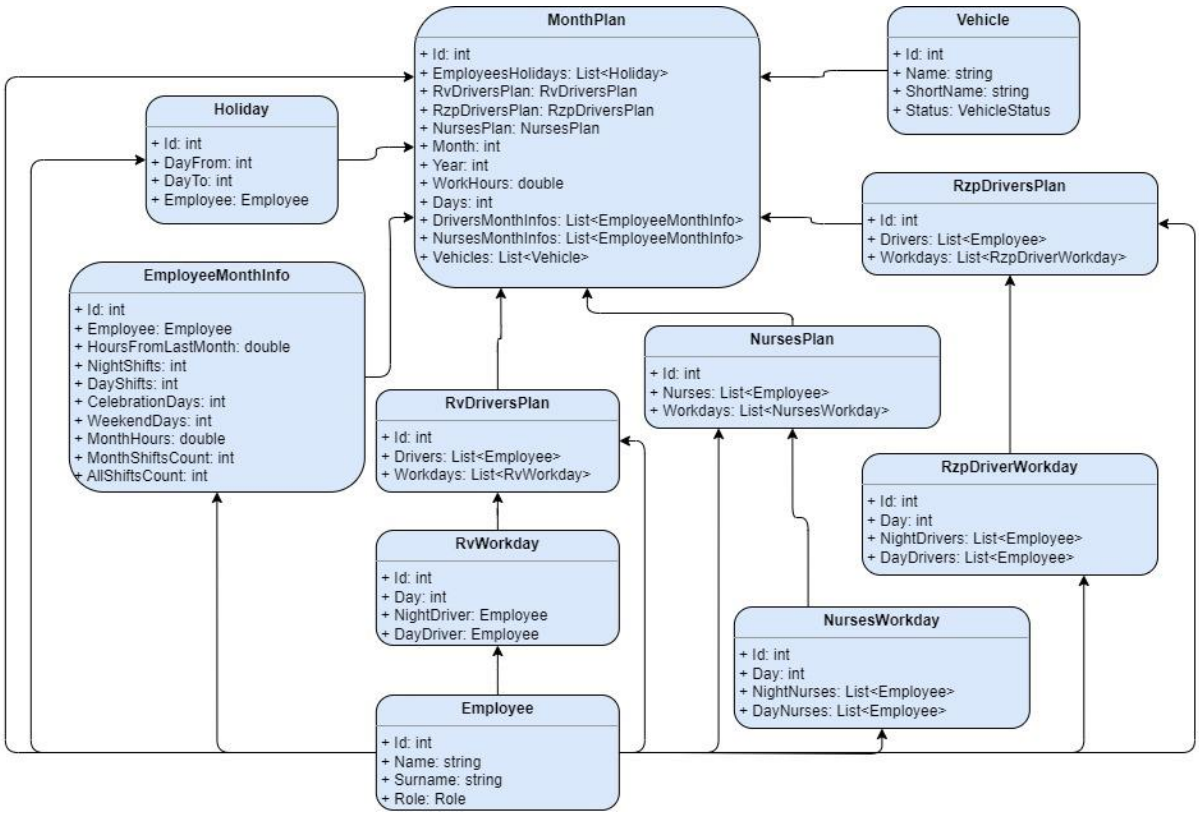
Jak bylo uvedeno k ukládání a manipulaci dat byl použit nástroj LiteDB. Technologie obsahuje funkce pro čtení, úpravu, zápis, mazání a filtrování dat. Pro usnadnění psaní dotazů, lze využít výrazy LINQ (lambda funkce) přímo v jazyce C#. Výrazy LINQ vypadají velmi podobně jako syntaxe jazyka SQL. Na následujícím obrázku je příklad využití LINQ výrazu, který vyfiltruje nepojízdná vozidla a vrátí určitý počet vozidel parametrem `count`.

```
public List<Vehicle> GetVehicles(int count)
{
    using var db = new LiteDatabase(path);
    var col = db.GetCollection<Vehicle>(dataCollectionName);

    List<Vehicle> vehicles = col.Query()
        .Where(x => x.Status == Enums.VehicleStatus.OK)
        .Limit(count)
        .ToList();

    return vehicles;
}
```

Obrázek 3: LiteDB dotaz s využitím LINQ

5.1.1 Data model diagram

Následující diagram představuje datový model tříd, který se používá pro plánování směn a zachycuje vztahy mezi nimi. Třída MonthPlan zahrnuje informace o měsíčním plánu, jako je dovolená, seznam použitých vozidel a další údaje. Tyto informace se používají pro export směn všech posádek. Třídy RvDriversPlan, NursesPlan a RzpDriversPlan slouží k uchovávání informací o pracovních dnech a zaměstnancích, kteří jsou součástí dané posádky v průběhu měsíce. Důvodem tohoto návrhu je skutečnost, že řidiči posádek RV a RZP se mohou vzájemně střídat a nahrazovat. Pro budoucí účely se takovéto řešení vztahuje i na záchranáře, kteří mohou být také řidiči. Třída EmployeeMonthInfo uchovává informace o měsíci určitého zaměstnance, například počet nočních a denních směn, odpracované hodiny, celkový počet odpracovaných směn za celý rok a přesměrované hodiny z minulého měsíce. Třída Vehicle reprezentuje vozidlo a uchovává jeho celý a zkrácený název pro export a stav vozidla (pojízdné/nepojízdné). Podobně třída Employee reprezentuje zaměstnance s vlastnostmi pracovní pozice, jména a příjmení. Datový model Holiday je určen k uchovávání informací o řádné dovolené a příslušnému zaměstnanci.



Obrázek 4: Data Model Diagram

5.1.2 Repositáře využívající LiteDB

Každá výše zmíněná třída data Modelu má vlastní třídu Repository, která dědí z třídy BaseRepository. BaseRepository je abstraktní generická třída sloužící jako základ a je určená k implementaci repositářů pro databázi LiteDB. Repositář implementuje rozhraní IGenericRepository a umožňuje základní operace nad databází LiteDB, jako jsou čtení, zápis, aktualizace a mazání dat.

```
public abstract class BaseRepository<T> : IGenericRepository<T> where T : class
{
    protected const string path = @".\shifts.db";
    protected string dataCollectionName;

    public BaseRepository(string dataCollectionName)
    {
        this.dataCollectionName = dataCollectionName;
    }

    public void Delete(int id)
    {
        using var db = new LiteDatabase(path);

        var col = db.GetCollection<T>(dataCollectionName);
        col.Delete(id);
    }

    public List<T> GetAll(string query = "")
    {
        using var db = new LiteDatabase(path);

        var col = db.GetCollection<T>(dataCollectionName);
        var list = col.Query().ToList();
        return list;
    }

    public T GetById(int id)
    {
        using var db = new LiteDatabase(path);

        var col = db.GetCollection<T>(dataCollectionName);
        return col.FindById(id);
    }

    public void Insert(T entity)
    {
        using var db = new LiteDatabase(path);

        var col = db.GetCollection<T>(dataCollectionName);
        try
        {
            col.Insert(entity);
        }
        catch (LiteException)
        {
            Console.WriteLine("LiteException was thrown");
        }

        this.EnsureIndex(col);
    }

    public void Update(T entity)
    {
        using var db = new LiteDatabase(path);

        var col = db.GetCollection<T>(dataCollectionName);
        col.Update(entity);
        this.EnsureIndex(col);
    }

    public abstract void EnsureIndex(ILiteCollection<T> col);
}
```

Obrázek 5: Základní generická třída repositáře pro LiteDB

Třída BaseRepository<T> používá konstantu path k určení cesty souboru s databází LiteDB. Dále obsahuje proměnnou dataCollectionName určující název kolekce ukládaných dat. Tento název se používá ve všech metodách třídy pro práci s kolekcemi.

BaseRepository<T> obsahuje následující metody:

Delete(int id)

Metoda slouží k odstranění záznamu z databáze na základě zadaného id. Vnitřně používá instanci databáze LiteDB a kolekci odpovídající předanému typu T jako parametr.

GetAll(string query = "")

Metoda slouží k získání všech záznamů z databáze odpovídající danému typu T. Může být také použita pro vyhledání záznamů na základě parametru query, což je řetězec obsahující podmínky pro vyhledávání.

GetById(int id)

Metoda slouží k získání záznamu z databáze na základě zadaného id.

Insert(T entity)

Metoda slouží k vložení nového záznamu do databáze.

Update(T entity)

Metoda slouží k aktualizaci záznamu v databázi.

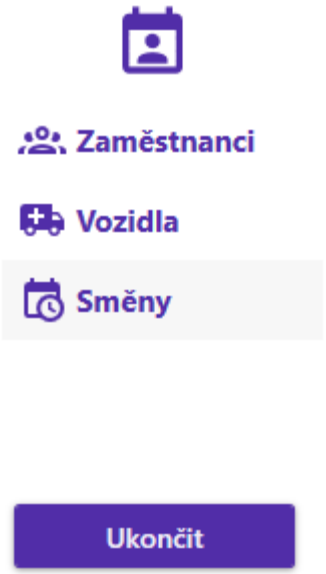
EnsureIndex(ILiteCollection<T> col)

Tato abstraktní metoda vytváří index na kolekci v databázi LiteDB. Metoda musí být implementována v potomcích třídy BaseRepository<T>.

5.2 Uživatelské rozhraní (UI)

Uživatelské rozhraní je rozděleno do tří modulů, které zahrnují informace o zaměstnancích, vozidlech a směnách. Toto rozdělení nabízí přehlednější organizaci obsahu aplikace a lepší orientaci uživatele v programu.

Každý z těchto modulů je pojmenován, označen příslušnou ikonou a je vybaven vyhledávacím polem pro snadnější vyhledávání patřičné entity aplikace. Jednotlivé moduly lze otevřít skrz navigační panel, kde lze najít i tlačítko Ukončit, které slouží pro bezpečné ukončení aplikace.



Obrázek 6: Navigační panel

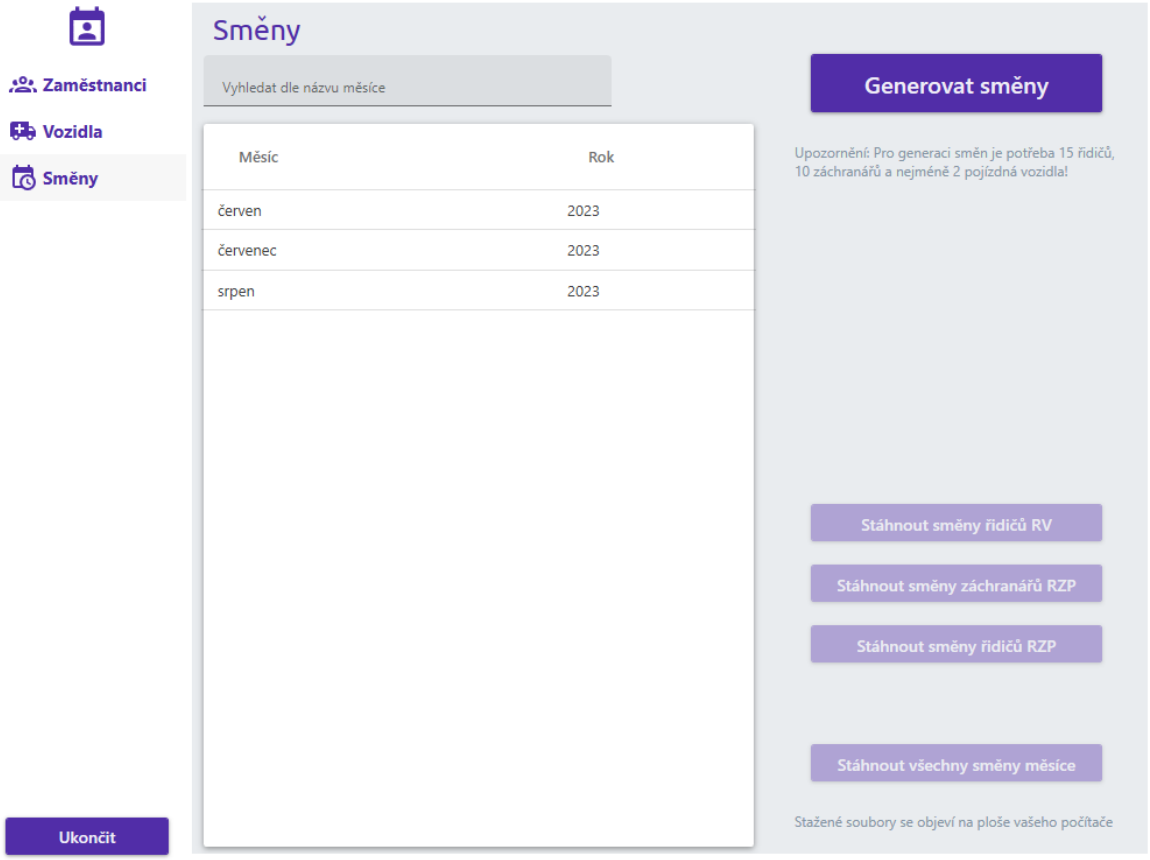
Vytvořená aplikace nenabízí žádné přihlášení ani autorizaci, protože je navržená tak, aby ji používal pouze jeden uživatel, který je zodpovědný za plánování směn. Data o směnách neuchovávají žádné citlivé informace zaměstnanců.

5.2.1 Tlačítka

Tlačítka uživatelského rozhraní jsou ošetřená na akce, které požadují výběr určité položky v seznamech. Pokud položka není vybraná nebo vytvořené je tlačítko nedostupné a ani nejde provést akci, kterou tlačítko nabízí. Nedostupná tlačítka jsou označená světlejším odstínem modré barvy, jak je vidět například na dalším obrázku číslo 7.

5.2.2 Modul Směny

Při spuštění aplikace, je automaticky zobrazen modul Směny, kde lze vygenerovat směny pomocí tlačítka Generovat směny, nebo vybrat již vytvořený plán směn a vyexportovat si obsah daného měsíce v souboru XLSX, který je podporován programem Excel.

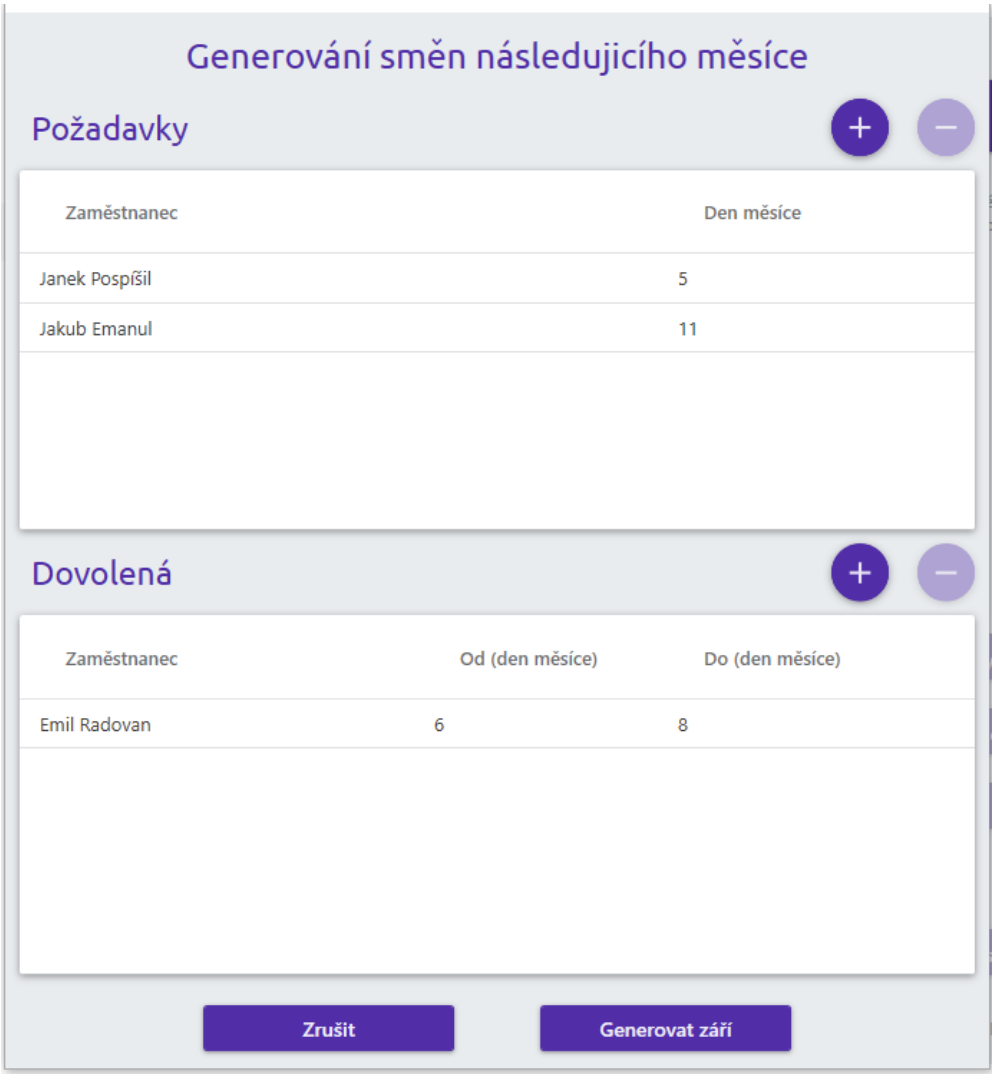


Obrázek 7: UI modul Směny

5.2.2.1 Generování

Pokud by byl seznam vygenerovaných směn prázdný, aplikace automaticky bude počítat s generováním směn na následující měsíc od aktuálního dne. Tento plán bude vygenerovaný automaticky tak, že nepočítá s žádným přesunem hodin z předchozí měsíce a dalších zohledněných informací podobně, jak tomu je při nově započatém roku v měsíci leden. V opačném případě se zohlední veškeré informace z posledně vygenerovaného měsíce a program by generoval plán směn na měsíc další.

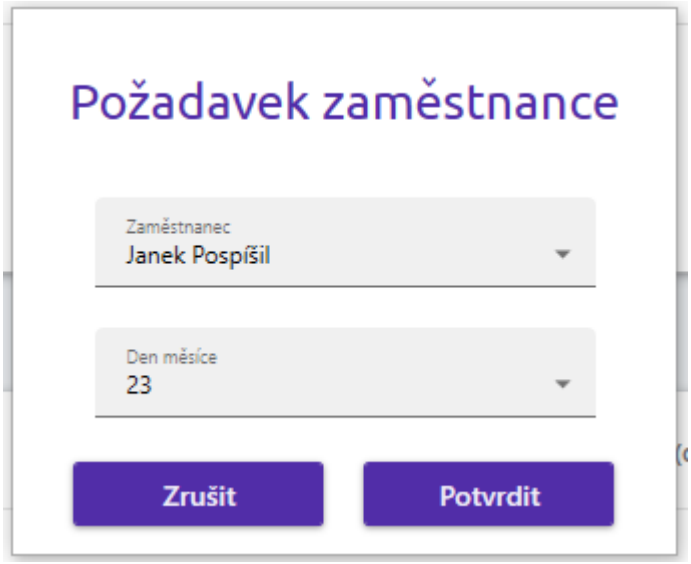
Po kliknutí na tlačítko směny se uživateli zpřístupní dialogové okno, kde si může před generováním směn zadat požadavky a řádnou dovolenou k příslušnému zaměstnanci.



Obrázek 8: Nabídka požadavků a dovolené před zahájením generace směn

Jednotlivé požadavky lze zadat po stisku tlačítka + v sekci požadavků. Po stisku tlačítka se zobrazí uživateli následující dialog (viz obrázek 9), který obsahuje jednoduchý formulář s nabídkou zaměstnanců a čísla dnů měsíce. Jakmile uživatel vyplní potřebné údaje, zpřístupní se mu tlačítko Potvrdit. Následně se dialog zavře a požadavek se zobrazí v seznamu Požadavky.

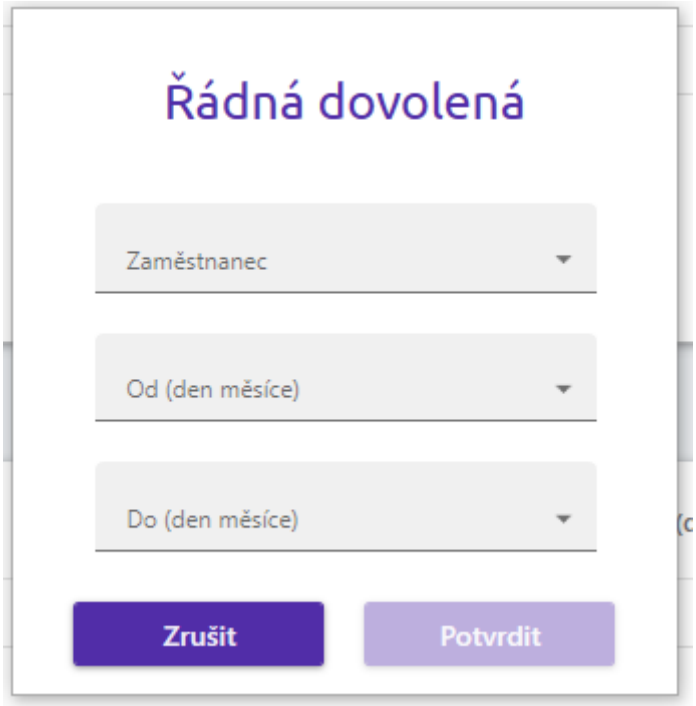
V případě, že se uživatel spletl, lze po výběru požadavek smazat kliknutím na tlačítko – v sekci požadavků.



Obrázek 9: Dialog pro vytvoření požadavku zaměstnance

Zadávání řádné dovolené je velmi podobné zadávání požadavků. Pro vytvoření záznamu řádné dovolené musí uživatel kliknout na tlačítko + tentokrát v sekci Dovolená. Po kliknutí se mu zobrazí dialog řádné dovolené. Dialog se liší tím, že uživatel musí vybrat den od kdy a do kdy chce zaměstnanec čerpat dovolenou. Dialog dovolené je znázorněn na obrázku 10.

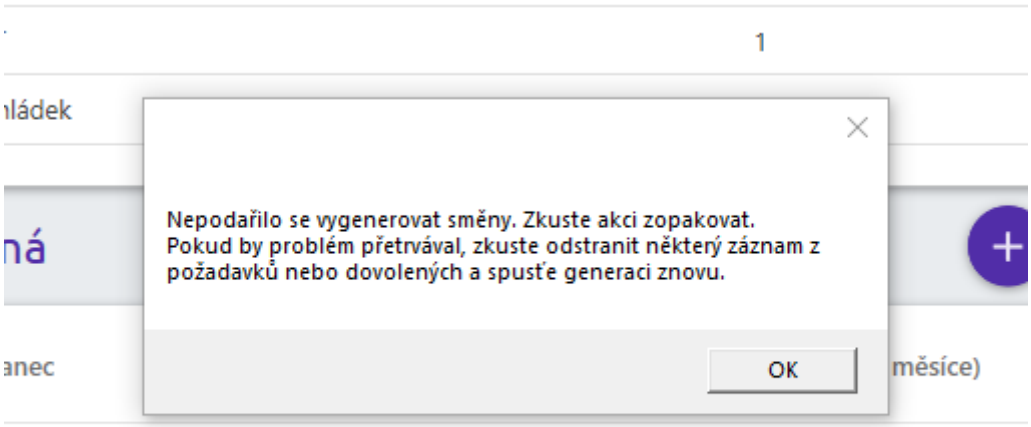
Mazání probíhá stejným způsobem jako u požadavků s rozdílem, že se tlačítko – zpřístupní v sekci dovolené.



Obrázek 10: Dialog řádné dovolené

Jak je vidět na obrázku 8, vedle tlačítka Generovat se v dialogu zobrazí měsíc, ke kterému chce uživatel aktuálně generovat směny.

Pokud by uživatel zadal požadavky nebo dovolenou tak, že by nebylo možné vytvořit směnu, zobrazí se dialog vyobrazený na obrázku 11. Po potvrzení chybového dialogu se uživateli zpřístupní jeho požadavky a dovolená a nedojde k jejich přemazání. Uživatel může pokračovat v úpravě dál.



Obrázek 11: Chybová hláška při generaci směn

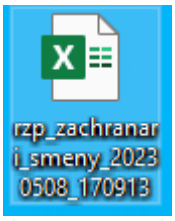
Při generování se vytvoří plán pro všechny posádky RV i RZP zároveň a jsou vloženy jako jednotlivé plány posádek do data Modelu MonthPlan, dle kterého se následně exportují soubory v aplikaci Excel. Bližší informace k jednotlivým plánům posádek jsou popsané v podkapitole 5.1.1 Data Model Diagram.

V případě, že celý proces generování proběhl v pořádku, zavře se dialogové okno znázorněné na obrázku 8 a zobrazí se nový záznam směn ve výpisu v přehledu směn.

5.2.2.2 Export

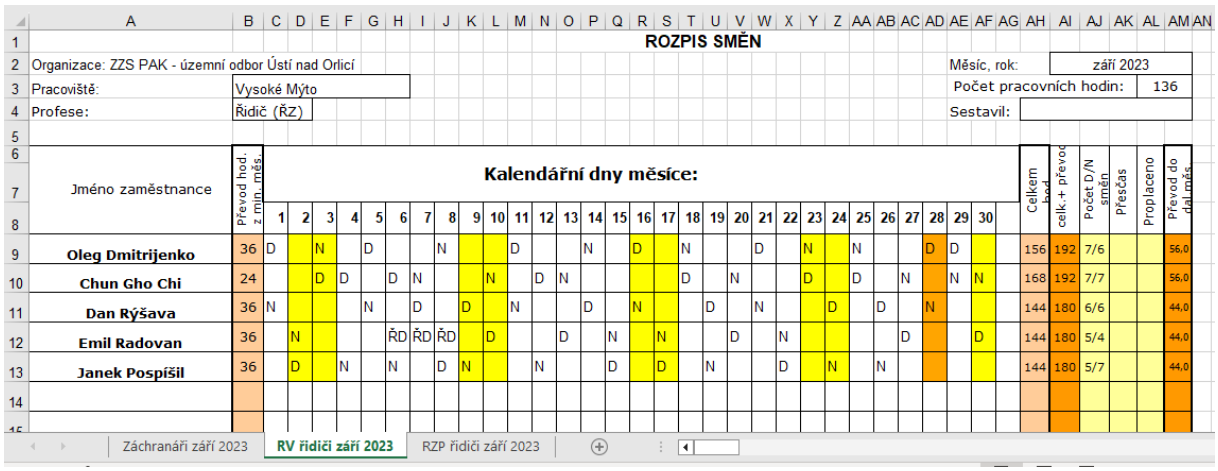
Modul Směny je vybaven čtyřmi tlačítky pro export směn, viz obrázek 7. První tři tlačítka slouží pro export směn do souboru XLSX zvlášť od ostatních posádek podle vybraného tlačítka. Poslední tlačítko stáhne všechny posádky do jednoho souboru a to tak, že každou posádku zapíše do samostatného listu aplikace Excel. Před exportem je však nutné vybrat z přehledu směn konkrétní měsíc.

Pro pojmenování exportovaných souborů se používá aktuální datum a čas ve formátu posadka_pracovniPozice_smeny_RRRRMMDD_HHMMSS. Stažené soubory se uživateli objeví přímo na ploše počítače.



Obrázek 12: Exportovaný soubor směn pro záchranáře RZP

Pro export je zachována původní šablona, která se využívala pro manuální plánování. Následující obrázek je výsledkem exportu po kliknutí na tlačítko Stáhnout všechny směny měsíce. Ve spodní liště jsou vidět tři listy – Záchranáři září 2023, RV řidiči září 2023 a RZP řidiči září 2023. Každý z listů obsahuje směny a informace o měsíci jedné pracovní pozice konkrétní posádky.



ROZPIS SMĚN

Organizace: ZZS PAK - územní odbor Ústí nad Orlicí | Měsíc, rok: září 2023
Pracoviště: Vysoké Mýto | Počet pracovních hodin: 136
Profese: Řidič (ŘZ) | Sestavil:

Jméno zaměstnance	Převod hod. z min. měs.	Kalendářní dny měsíce: 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	Celkem hod.	celk.+ převoc	Počet D/N směn	Přesčas	Proplaceno	Převod do dal.měs.
Oleg Dmitrijenko	36	D		N		D			N			D			N		D		N			D		N		N			D	D		156	192	7/6			56,0
Chun Gho Chi	24			D	D		D	N			N		D	N					D		N			D		D		N		N	N		168	192	7/7			56,0
Dan Rýšava	36	N				N		D		D		N			D		N			D		N			D		D		N				144	180	6/6			44,0
Emil Radovan	36		N				ŘD	ŘD	ŘD		D			D		N		N			D		N					D			D			144	180	5/4			44,0
Janek Pospíšil	36		D		N		N		D	N			N			D		D		N			D		N		N							144	180	5/7			44,0

Záchranáři září 2023 | RV řidiči září 2023 | RZP řidiči září 2023

Obrázek 13: Export směn všech posádek

Obrázek 13 popisuje vygenerovaný a exportovaný přehled směn měsíce září roku 2023 pro řidiče posádky RV. V horní části dokumentu jsou zobrazeny pracovní hodiny měsíce září a profese, pro kterou je tabulka vygenerovaná.

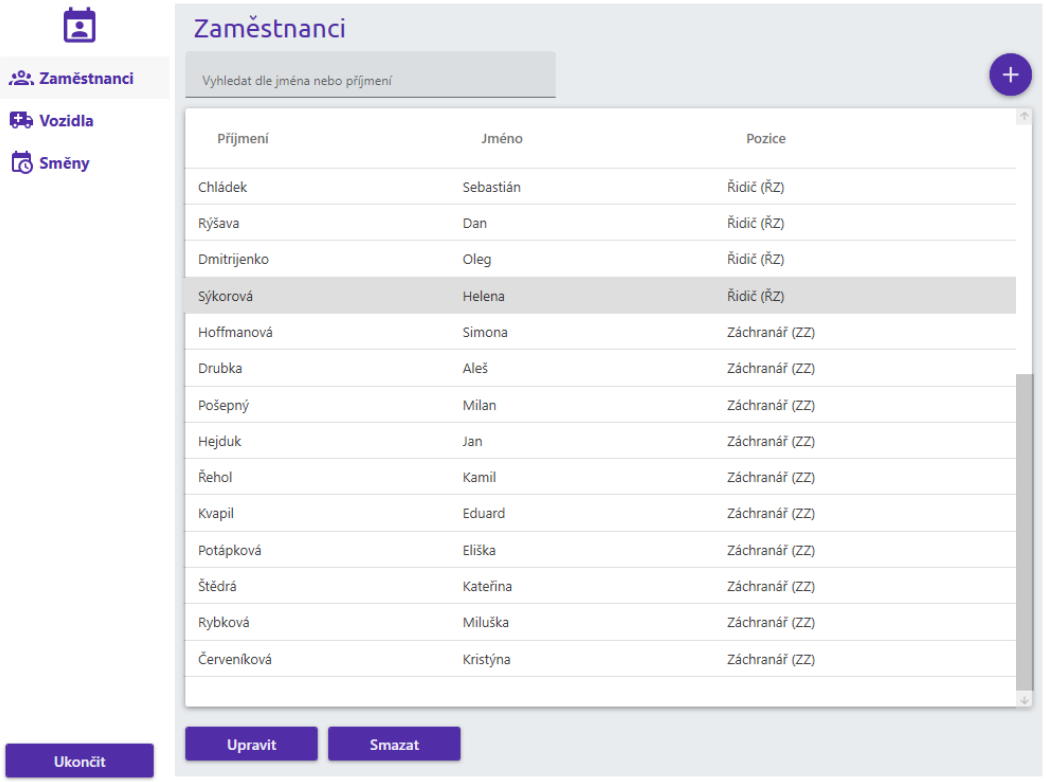
Obsahem tabulky jsou řidiči posádky RV a dny ve kterých řidiči slouží směnu. Víkendové dny jsou označeny žlutou a státní svátky tmavě oranžovou barvou. Na levé straně tabulky hned za jmény je vidět přesunutý počet hodin z minulého měsíce a na pravé straně jsou vypočítané hodnoty hodin a směn. Tabulka obsahuje například poměr nočních a denních směn, počet odpracovaných hodin v měsíci září, včetně převodu z předchozího měsíce, a převod hodin

do měsíce následujícího. RZP záchranáři a řidiči obsahují oproti RV posádce navíc označení vozidla, které je k dispozici během jejich směny. Označení vozidla se zapisuje hned za označením směny.

Hodnoty D a N určují, zda se jedná o denní nebo noční směnu. Pokud se v tabulce vyskytuje hodnota ŘD, jedná se o řádnou dovolenou, kterou zadal uživatel aplikace před generací směn. Viz předchozí kapitola Generování.

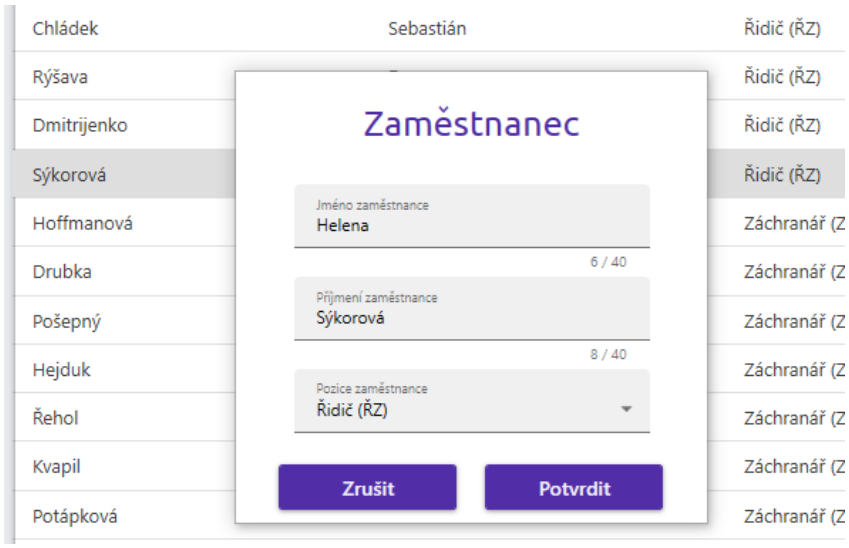
5.2.3 Modul Zaměstnanci

Modul Zaměstnanci slouží ke správě zaměstnanců ZZS. V rámci modulu je možné vyhledat, přidat, upravit, nebo smazat zaměstnance. Přidání zaměstnance probíhá skrz tlačítko + v pravém horním rohu (viz obrázek 12). Po kliknutí na tlačítko se zobrazí dialog zaměstnance, pomocí kterého lze vyplnit údaje a zaměstnance přidat.



Obrázek 14: UI Modul Zaměstnanci

Následující dialog zobrazený na obrázku 13 slouží pro vytváření nového a editaci stávajícího zaměstnance. Obsahem dialogu je textové pole pro jméno a příjmení a selectbox[4] pro výběr pracovní pozice. Pro editaci zaměstnance je potřeba vybrat osobu v přehledu a zmáčknout tlačítko Upravit.

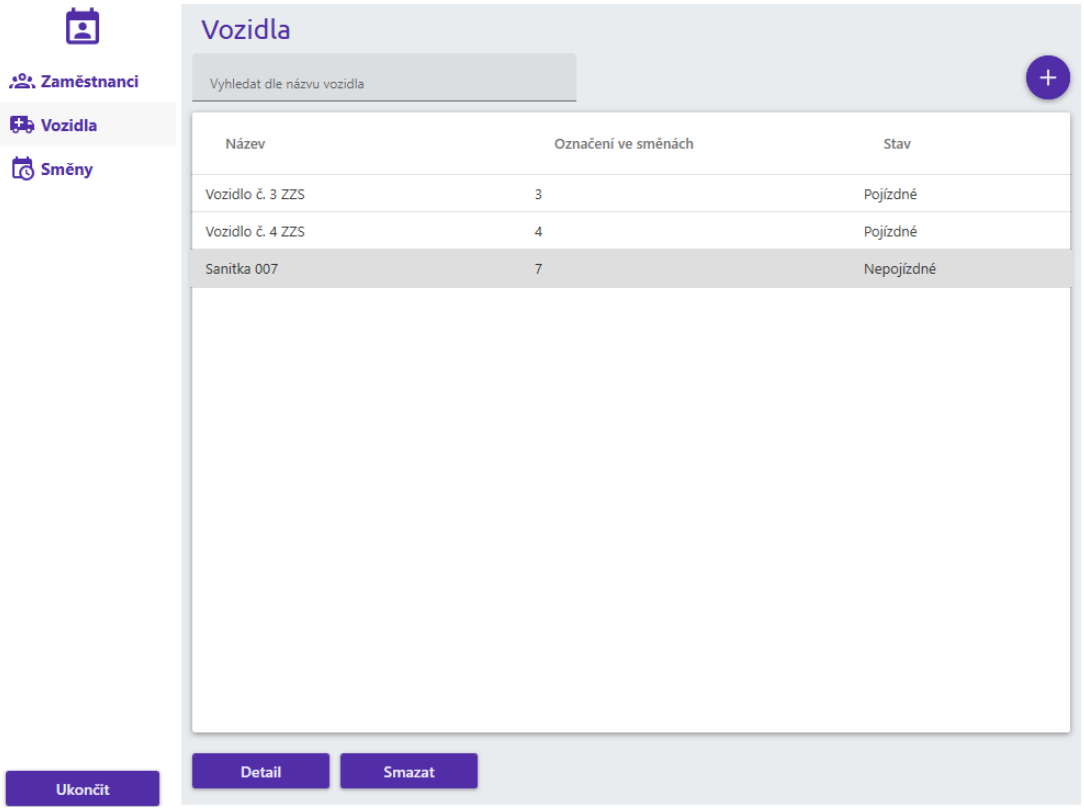


Obrázek 15: Editace v dialogu zaměstnance

5.2.4 Modul Vozidla

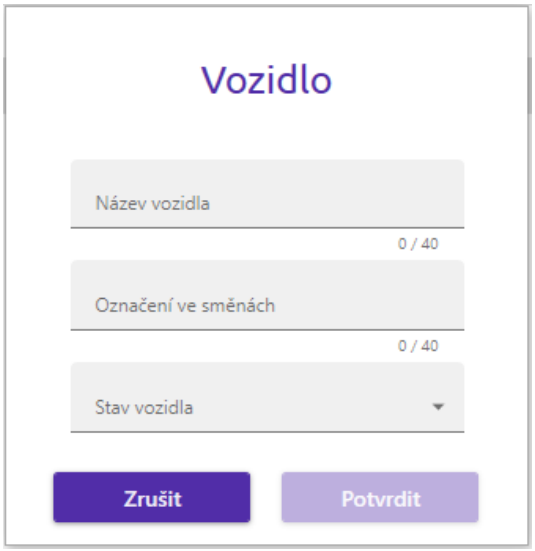
Modul Vozidla obsahuje obdobné řešení jako modul Uživatelé. Zaměřuje se na zprávu vozidel a nabízí veškerý přehled pojízdných i nepojízdných vozidel. Součástí modulu je dialog, pomocí kterého lze vytvořit nové vozidlo po zmáčknutí tlačítka + v pravém horním rohu, nebo upravit vozidlo vybrané z nabídky v hlavní části po stisku tlačítka Detail.

[4] Grafický prvek, který nabízí předem definované položky k výběru



Obrázek 16: UI Modul Vozidla

V dialogu vozidla lze zadat zkrácený a celý název vozidla a stav vozidla, zda je vozidlo pojízdné, nebo nepojízdné. Dialog slouží pro přidání nového a editaci stávajícího vozidla.



Obrázek 17: Dialog vozidla

6 Zhodnocení aplikace

Aplikace je plně funkční, ale některé skutečné situace nejsou zohledněny. Jedním z příkladů je nemožnost manuálního zásahu do vygenerovaných směn. V průběhu měsíce se často zapisuje pracovní neschopnost zaměstnanců, kterou nelze předem odhadnout a směny je nutno přepisovat dle potřeb a aktuálního pracovního nasazení. Proto je nutné aplikaci rozšířit o další funkcionality, které jsou popsány v následující kapitole 6.1.

6.1 Budoucí rozšíření

Další kapitola se nevěnuje pouze nezbytným funkcionalitám pro reálné využití aplikace. Obsahem je také rozšíření, které se nemusí nutně týkat jen generování směn a mohou značně usnadnit práci všem zaměstnancům ZZS. Jde spíše o budoucí návrh, jak aplikaci posunout na další úroveň použitelnosti.

6.1.1 Manuální zásah

Jak bylo zmíněno v popisu hlavní kapitoly, je nutné zohlednit reálné aspekty a situace, které mohou v průběhu měsíce nastat. Proto je nezbytně nutné umožnit manuální zásah do vygenerovaných směn, aby aplikace rozpoznala a zohlednila změny uživatele aplikace.

6.1.2 Mazání a úprava směn

Bylo by vhodné umožnit uživateli také vygenerovaný plán smazat, pokud by z jistých důvodu rozhodl, že jde o nepoužitelný nebo nevyhovující návrh směn.

6.1.3 Požadavky zadané zaměstnancem

V rámci desktopové aplikace by bylo možné vytvořit vedlejší mobilní aplikaci, která by umožnila zaměstnancům zadívat vlastní požadavky do systému. Staniční sestra pak nebude muset sbírat informace kdy, kdo a jak může danou směnu pracovat. S tím bude souviset i víceúrovňová míra oprávnění, autentizace a změna databáze LiteDB.

6.1.4 Priority požadavků

Značnou výhodou by bylo zohledňovat priority důležitějších požadavků zaměstnanců, tak jako ve skutečném životě. Algoritmus generování by následně rozpoznal a odstranil méně důležitější požadavky dle svého uvážení v případě, kdy je jich zadáno příliš mnoho.

6.1.5 Kontrola čerpání dovolené

Aplikace by mohla kontrolovat čerpání dovolené. V případě vyčerpání dovolené by aplikace zobrazila upozornění na uživatelskou obrazovku, pokud by uživatel zadával dovolenou mimo rozsah stanovený zákonem a danou organizací.

6.1.6 Přehled informací

Aplikace by mohla vést statistiku a přehled dat o směnách a dovolených určitého zaměstnance. Tento přehled by byl v dalším modulu, který by usnadnil orientaci a práci uživatele při zadávání dovolené, či požadavků dalšího měsíce před generací směn. Tato funkcionality by také mohla výrazně pomoci při vedení účetnictví a bonusového mzdového odměňování zaměstnanců.

Závěr

Celkově lze říct, že návrh a implementace aplikace pro plánování směn ZZS byla pro mě velkou výzvou a zároveň i velmi zajímavým projektem. Zvolil jsem si dané téma sám a snažil jsem se, aby byl výsledek mé práce co nejvíce přínosný a reálně využitelný.

Díky implementaci této aplikace jsem se naučil hledat řešení pro různé problémy a plánovat postup práce tak, aby byl co nejefektivnější. Zároveň jsem se seznámil s technologiemi C# a WPF, konkrétně s knihovnou MaterialDesignInXaml, což mi rozšířilo obzory v oblasti tvorby desktopových aplikací.

Věřím, že tato aplikace může být skutečně užitečná pro ZZS při plánování směn a doufám, že bude v budoucnu dále vylepšována a rozšiřována. Celkově považuji svou bakalářskou práci za úspěšnou a cennou zkušenost pro můj další profesní růst.

Citovaná literatura

[1] Organizace ZZS. In: *ZSSPAK* [online]. Průmyslová 450, 530 03 Pardubice: Pardubický kraj, 2023 [cit. 2023-05-08]. Dostupné z: https://www.zzspak.cz/o-nas/o-zdravotnicke-zachranne-sluzbe

[2] *Záchranka se v Mýtě stěhuje do nové budovy, do terénu pošle víc posádek* [online]. Pardubický kraj: Pokorná, 2021 [cit. 2023-05-08]. Dostupné z: https://orlicky.denik.cz/zpravy_region/zachranka-se-v-myte-stehuje-do-nove-budovy-do-terenu-posle-vic-posadek-20210127.html

[3] *Výjezdové základny* [online]. Pardubický kraj: Pardubický kraj, 2023 [cit. 2023-05-08]. Dostupné z: https://www.zzspak.cz/o-nas/vyjezdove-zakladny

[4] *A tour of the C# language* [online]. Washington: Microsoft, 2023 [cit. 2023-05-08]. Dostupné z: https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/

[5] *Announcing .NET 6 Preview 1* [online]. Washington: Lander, 2021 [cit. 2023-05-08]. Dostupné z: https://devblogs.microsoft.com/dotnet/announcing-net-6-preview-1/

[6] *Lekce 1 - Úvod do WPF (Windows Presentation Foundation)* [online]. Praha: Čápka, 2020 [cit. 2023-05-08]. Dostupné z: https://www.itnetwork.cz/csharp/wpf/c-sharp-tutorial-wpf-uvod-a-prvni-formularova-aplikace

[7] *EPPlus 5/6* [online]. Stockholm: EPPlus Software, 2023 [cit. 2023-05-08]. Dostupné z: https://epplussoftware.com/en/Developers/Features

[8] *Public Holidays* [online]. -: Willey, 2023 [cit. 2023-05-08]. Dostupné z: https://github.com/martinjw/Holiday

[9] *Introduction to the MVVM Toolkit* [online]. Rome: Pedri, 2023 [cit. 2023-05-08]. Dostupné z: https://learn.microsoft.com/en-us/dotnet/communitytoolkit/mvvm/

[10] *.NET dependency injection* [online]. Pewaukee: Pine, 2023 [cit. 2023-05-08]. Dostupné z: https://learn.microsoft.com/en-us/dotnet/core/extensions/dependency-injection

[11] *Material Design In XAML Toolkit* [online]. London: Material design, 2023 [cit. 2023-05-08]. Dostupné z: https://github.com/MaterialDesignInXAML/MaterialDesignInXamlToolkit

[12] *Embedded NoSQL database for .NET* [online]. Belo Horizonte: Maurício, 2023 [cit. 2023-05-08]. Dostupné z: https://www.litedb.org/

[13] Mvvm: model-view-viewmodel. In: *Dotnetportal.cz* [online]. Praha: Dajbych, 2009 [cit. 2023-05-08]. Dostupné z: https://www.dotnetportal.cz/clanek/4994/MVVM-Model-View-ViewModel

[14] Understanding AddTransient Vs AddScoped Vs AddSingleton In ASP.NET Core. In: *C# Corner* [online]. Bangalore: Maiti, 2021 [cit. 2023-05-08]. Dostupné z: https://www.c-sharpcorner.com/article/understanding-addtransient-vs-addscoped-vs-addsingleton-in-asp-net-core/

[15] Singleton Design Pattern In C#. In: *C# Corner* [online]. Mumbai: Alle, 2021 [cit. 2023-05-08]. Dostupné z: https://www.c-sharpcorner.com/UploadFile/8911c4/singleton-design-pattern-in-C-Sharp/