ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ v PRAZE

FAKULTA BIOMEDICÍNSKÉHO INŽENÝRSTVÍ

Katedra biomedicínské informatiky

Aplikace pro plánování služeb lékařů oční kliniky

Application for planning the services of ophthalmologists

Bakalářská práce

Studijní program:	Biomedicínská a klinická technika
Studijní obor:	Biomedicínská informatika

Autor bakalářské práce:	David Aron
Vedoucí bakalářské práce:	Mgr. Radim Krupička, Ph.D.

Kladno 2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení:	**Aron**	Jméno:	**David**	Osobní číslo:	**491779**
Fakulta:	**Fakulta biomedicínského inženýrství**				
Garantující katedra:	**Katedra biomedicínské informatiky**				
Studijní program:	**Biomedicínská a klinická technika**				
Studijní obor:	**Biomedicínská informatika**				

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Aplikace pro plánování služeb lékařů oční kliniky

Název bakalářské práce anglicky:

Application for planning the services of ophthalmologists

Pokyny pro vypracování:

Cílem bakalářské práce je na základě požadavků oční kliniky Fakultní nemocnice Královské Vinohrady navrhnout a vytvořit aplikaci pro plánování služeb lékařů. Aplikace bude umožňovat nastavit požadavky na služby jednotlivých lékařů, editaci ambulancí, generování měsíčního rozvrhu a jeho editaci. Výsledný rozvrh by měl být zobrazen v přehledném kalendáři služeb. Vybrané implementační technologie by měly umožňovat budoucí multiplatformní spuštění. Důraz kladte na otevřenost a udržitelnost aplikace a možnost jejího předání pro další vývoj.

Seznam doporučené literatury:

[1] Matt Lacey , Marcel Alexander Wagner, Creating Cross-Platform C# Applications with Uno Platform, ed. 1, Packt, 2021, ISBN 9781801078498
[2] Hana Kanisová, Miroslav Müller, UML srozumitelně, ed. 2, Computer Press a.s., 2006, ISBN 80-251-1083-4

Jméno a příjmení vedoucí(ho) bakalářské práce:

Mgr. Radim Krupička, Ph.D.

Jméno a příjmení konzultanta(ky) bakalářské práce:

Ing. Patrik Pluhovský

Datum zadání bakalářské práce: **14.02.2022**
Platnost zadání bakalářské práce: **18.09.2023**

doc. Ing. Zoltán Szabó Ph.D.
vedoucí katedry

prof. MUDr. Jozef Rosina, Ph.D., MBA
děkan

PROHLÁŠENÍ

Prohlašuji, že jsem bakalářskou práci s názvem „Aplikace pro plánování služeb lékařů oční kliniky“ vypracoval samostatně a použil k tomu úplný výčet citací použitých pramenů, které uvádím v seznamu přiloženém k bakalářské práci.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu § 60 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.

V Kladně 12.05.2022 …..……..……...

David Aron

PODĚKOVÁNÍ

Velice rád bych poděkoval vedoucímu této práce, doktoru Radimu Krupičkovi, za motivaci k této práci, cenné rady a pohotovou komunikaci. Dále bych chtěl poděkovat zaměstnancům oční kliniky Fakultní nemocnice Královské Vinohrady, především panu inženýrovi Patriku Pluhovskému a paní doktorce Ditě Rejholcové, za spolupráci při analýze momentální situace plánování služeb a jak ji co nejlépe zpřehlednit a zjednodušit. V neposlední řadě děkuji své rodině a přítelkyni za podporu během celého studia.

ABSTRAKT

Aplikace pro plánování služeb lékařů oční kliniky

Hlavním cílem této práce je vytvoření aplikace, která usnadní plánování služeb na oční klinice Fakultní nemocnice Královské Vinohrady, kde momentální situace není zcela ideální a zabírá zbytečně mnoho času. Důraz je kladen na budoucí multiplatformní použití aplikace, její přehlednost, udržitelnost a možnost dalšího rozvoje. Za tímto účelem bylo zvoleno použití platformy UNO, která umožňuje běh aplikace psané jedním kódem na více platformách, a vytvoření webového API pro přístup k databázovému serveru. Aplikace disponuje správou pracoviště, zaměstnanců a služeb interaktivním způsobem s částečnou automatizací, který zpřehledňuje a urychluje tvorbu rozvrhu.

Klíčová slova

Multiplatformní aplikace, plánovač služeb, správa pracoviště, Uno Platform, C#

ABSTRACT

Application for planning the services of ophthalmologists

The main goal of this work is to create an application that will facilitate the planning of services at the ophthalmology clinic of the University Hospital Královské Vinohrady, where the current situation is not quite ideal and takes up an unnecessary amount of time. The emphasis is on future cross-platform use of the application, its clarity, sustainability and the possibility of further development. For this purpose, the use of the UNO platform was chosen, which allows running an application written with one code on multiple platforms and creating a Web API for accessing the database server. The application features the management of the workplace, employees and services in an interactive way with partial automation, which makes scheduling clearer and faster.

Keywords

Multiplatform application, service scheduler, workplace management, Uno Platform, C#

Obsah

Seznam symbolů a zkratek

Seznam zkratek

Zkratka	Význam
API	Rozhraní pro programování aplikací (Application Programming Interface)
WinUI	Knihovna s grafickými a ovládacími prvky (Windows User Interface Library)
UWP	Prostředí pro vývoj univerzálních aplikací (Universal Windows Platform)
HTML	Značkovací jazyk (Hypertext Markup Language)
WPF	Knihovna pro tvorbu grafického rozhraní (Windows Presentation Foundation)
XAML	Značkovací jazyk (Extensible Application Markup Language)
UX	User Experience
HTTP	Internetový protokol určený pro komunikaci (Hypertext Transfer Protocol)
URL	Webová adresa (Uniform Resource Locator)
JSON	Způsob zápisu dat do řetězce (JavaScript Object Notation)
PWA	Progresivní webové aplikace (Progressive web Apps)
REST	Komunikační architektura využívající HTTP (Representational State Transfer)
DOM	Objektově orientovaná reprezentace XML a HTML (Document Object Model)
SQL	Standardizovaný strukturovaný dotazovací jazyk (Structured Query Language)

1 Úvod

Dnešní doba jde čím dál tím rychleji dopředu a my se tak v téměř každé situaci našeho života setkáváme s určitou modernizací či automatizací pomocí různých technologií a mnohdy už si to ani neuvědomujeme. Stejně rychlý vývoj však zasahuje i dané technologie, a tak je potřeba s nimi držet krok. Dnes jsou moderní technologie, např. jednotlivé menší aplikace nebo rozsáhlé informační systémy, hojně využívány také v nemocnicích. Stejně tak je tomu i na oční klinice ve Fakultní nemocnici Královské Vinohrady, kde z klasického plánování služeb zaměstnanců pomocí zapisovacích knih přešli na tabulkový software Microsoft Excel.

Tento software jim umožnil zpřehlednění a usnadnění tvorby, lepší dostupnost pro ostatní zaměstnance a celkově převést plánování služeb do elektronické podoby. Ani tato aplikace ale není dokonalá a pro daný případ není stále tou ideální volbou. Klinice se však nepodařilo nalézt existující software, který by vyhovoval všem jejich požadavkům, a proto přišla na fakultu žádost, zda bychom nebyli schopni vytvořit aplikaci, která by jim ušetřila čas a ulehčila samotné plánování. Hlavním cílem práce je tak vytvoření databázové aplikace, která ještě více usnadní, zrychlí a zpřehlední plánování služeb na této oční klinice. Pro hladší průběh plánování bude aplikace rozšířena o dodatečné funkce a o částečnou automatizaci.

V první části této bakalářské práce jsou popsány možnosti, které dnešní technologie nabízí pro vývoj moderních a multiplatformních aplikací, základní modely architektur aplikací a obecný přehled. Poté následuje seznámení se s aktuálním stavem plánování služeb na oční klinice ve Fakultní nemocnici Královské Vinohrady a upřesnění jejich požadavků na funkčnost a rozsah nově vznikající aplikace. Další část práce se zabývá návrhem celé aplikace, popisem a odůvodněním zvolených technologií a celkovou implementací finální podoby. Poslední část je pak věnována uživatelskému manuálu, instrukcím k instalaci, zprovoznění výsledné aplikace a diskusi s konečným zhodnocením celé práce.

2 Přehled současného stavu

Vývoj aplikací postupuje raketovým tempem a s každým dnem jsou aplikace žádanějším produktem. Aplikaci chce dnes mít vesměs každý, ať už se jedná o jakékoli odvětví na trhu. Ovládají naše každodenní životy a s těží naleznete nějaký den, kdy byste se s nějakou aplikací nesetkali. Aplikace nalezneme nejen v počítačích, mobilních telefonech a tabletech, ale také i např. v některých moderních domácích spotřebičích, v informačních cedulích, a tedy ve většině elektronických zařízení, které nás obklopují a už si ani nedokážeme život bez nich představit.

2.1 Možnosti vývoje aplikací

Jelikož aplikace jsou určené pro konkrétní zařízení, která následně mohou mít rozdílné např. operační systémy, technické požadavky, možnosti kompatibility s dalšími zařízeními, komunikaci s okolním světem a mnoho dalšího, vzniká nám tak velká různorodost v podobě použitých prostředků k vytvoření dané aplikace, způsobu tvorby aplikace, kompatibility aplikace a další náležitosti, nad kterými je potřeba se před začátkem vývoje aplikace zamyslet a vybrat tu cestu, která bude pro plánovanou aplikaci ideální a zajistí splnění všech jejích funkčních a technických požadavků.

Prostředky potřebné k sestavení aplikace lze dělit různými způsoby na několik kategorií. Tři větší kategorie, na které můžeme tyto prostředky dělit, jsou prostředky na tvorbu mobilních aplikací, tvorbu desktopových aplikací a tvorbu webových aplikací. Speciální kategorií pak mohou být např. multiplatformní prostředky, které dokáží aplikaci uzpůsobit pro všechny tyto 3 kategorie a následně i eliminovat diverzitu jednotlivých zařízení v každé kategorii, například co se týče operačních systémů v případě desktopových aplikací. V této práci se zaměříme na kategorie desktopových a web aplikací s přesahem do multiplatformní kategorie, jelikož přesně těchto kategorií se bude naše aplikace týkat.

2.1.1 Desktopové aplikace

Desktopové aplikace jsou specifické tím, že musí být konkrétně nainstalované na paměťových discích všech zařízeních, na kterých chceme aplikaci využívat. Pro jejich vývoj se využívají vyšší programovací jazyky (pro Windows např. C#). Aplikace je zároveň závislá na operačním systému zařízení, a proto je potřeba pro každý operační systém vyvinout aplikaci znovu. Například desktopová aplikace vyvinutá pro operační systém Windows nelze za normálních podmínek nainstalovat na operační systémy MacOS, Linux a jiné [1]. Mezi desktopové aplikace patří např. aplikace Adobe Photoshop, Skype, Matlab, většina aplikací Microsoft Office a spousty dalších.

Jednou z výhod těchto aplikací je schopnost fungovat v offline režimu, kdy není třeba mít zařízení připojeno k internetu. Nastává tím však omezení např. zálohování souborů, kdy je ukládání prováděno pouze na konkrétní zařízení a omezuje se tím třeba sdílení dat a týmová práce. Další nevýhodou jsou případně složitější a ruční aktualizace. Většina těchto aplikací má však tyto nevýhody částečně eliminované, jelikož se po připojení k internetu dokáží připojit na svá cloudová uložiště, kam zálohují data nebo odkud stahují aktualizace. [1]

K vývoji takových aplikací je určeno několik frameworků[1], které usnadňují vývoj a umožní vytvořit aplikaci pro konkrétní vybranou platformu s využitím potřebných prostředků. Mezi nejpoužívanější frameworky pro tvorbu desktopových aplikací patří např. WPF a UWP, Cocoa, GNOME nebo čistě desktopový multiplatformní framework Electron.

Operační systém Windows

WPF a UWP jsou frameworky pro tvorbu desktopových aplikací na operační systém Windows. Oběma dříve předcházel framework Windows Forms, který byl s příchodem Windows Vista a 7 nahrazen právě frameworkem WPF. Ten byl poté doplněn frameworkem UWP, jenž se objevil s příchodem Windows 10. Ve všech případech se jedná převážně o knihovny vizuálních komponent. Nový operační systém Windows 11 pak opět přinesl změnu, kdy přešel na multiplatformní systém Fluent Design od společnosti Microsoft.

UWP je tedy jednou z možností, jak psát aplikace na operační systém Windows. Jedná se o platformu vyvinutou opět společností Microsoft při představení operačního systému Windows 10 a lze ji považovat za částečného nástupce WPF. UWP však není zpětně kompatibilní, a tak lze využít pouze pro operační systém Windows 10 a novější. UWP je možné programovat několika jazyky (C#, C++, Visual Basic a Javascript) a pro grafické uživatelské rozhraní využívá knihovny WinUI, XAML, HTML nebo DirectX [2]. Obsahuje několik univerzálních ovládacích prvků usnadňující vývoj desktopových aplikací, které lze využívat nejen pro PC, ale také pro Windows Mobile (operační systém pro telefony a tablety), Xbox One[2] a Hololens[3].

Hlavní knihovnou uživatelského rozhraní je WinUI. Slouží pro sestavení vizuální podoby UWP aplikací a jejich ovládacích prvků [3]. Pro navržení vzhledu a ovládacích prvků aplikace se využívá značkového jazyka XAML (Kód 2.1).

[1] Softwarová struktura, která slouží jako podpora při programování a vývoji

[2] Herní konzole od společnosti Microsoft

[3] Brýle na virtuální realitu od společnosti Microsoft

```
<Button Command="{Binding AddEmployeeCommand}">
      <StackPanel Orientation="Horizontal" VerticalAlignment="Center">
            <FontIcon FontFamily="Segoe MDL2 Assets" Glyph="&#xE710;"
                  VerticalAlignment="Center" FontSize="12"/>
            <TextBlock Text="Přidat zaměstnance" Margin="5 0 0 0"
                  VerticalAlignment="Center"/>
      </StackPanel>
</Button>
```

Kód 2.1: Ukázka využití jazyka XAML k definování tlačítka (autor)

Jak bylo již zmíněno, s příchodem Windows 11 přešel Microsoft z frameworku UWP na cross-platform systém Fluent Design. Díky tomuto frameworku je možné vytvářet aplikace pro Windows, Android, MacOS, IOS, web, nebo případně jejich kombinace. To např. umožňuje spouštět aplikace z mobilních telefonů Android na zařízeních s Windows 11. To vše je doplněno o nový balíček vývojových komponent a nástrojů Windows App SDK.

K vývoji desktopových aplikací pro Windows se nejčastěji využívá programovací jazyk C# a vývojové prostředí Visual Studio od společnosti Microsoft.

Operační systém MacOS

Pro vytvoření desktopové aplikace na operační systém MacOS od společnosti Apple se využívá například framework Cocoa (případně Cocoa Touch). Ve skutečnosti se jedná o aplikační prostředí tvořeného skupinou několika objektově orientovaných frameworků pro vývoj aplikací na OS X (MacBooky) a iOS (iPhony/iPady). Jde o sadu objektově orientovaných softwarových komponent (tříd), která umožňuje rychle vytvářet robustní a plnohodnotné aplikace. Tyto třídy jsou opakovaně použitelné a přizpůsobitelné softwarové stavební bloky, které můžete používat v původní podobě nebo je rozšířit podle svých specifických požadavků. [4]

Dalším frameworkem využívaným pro vývoj MacOS aplikací je SwiftUI. Jedná se o další knihovnu modernizovaných prvků uživatelského rozhraní s vylepšenou optimalizací a novými funkcemi. [5]

K vývoji těchto aplikací se používají programovací jazyky Objective-C (rozšíření jazyka C) nebo Swift, který je novější alternativou a umí s Objective-C plně komunikovat a spolupracovat i v rámci jednoho kódu. Jako vývojové prostředí se nejčastěji využívá software Xcode od společnosti Apple.

Operační systém Linux

Pokud chce vývojář vytvořit desktopovou aplikaci určenou čistě pro operační systém Linux, nabízí se např. prostředí pracovní plochy a vývojová platforma GNOME. Skládá se z knihoven, služeb a nástrojů potřebných k vývoji a distribuci aplikací. Pokrývá tvorbu uživatelského rozhraní, úpravu kódu a finální distribuci [6]. GNOME je postaven

na knihovně GTK+, což je skupina dalších podknihoven pro grafické uživatelské rozhraní. Knihovna GTK+ původně vznikla jako základ grafického editoru GIMP.

Knihovny jsou postavené na jazyce C, avšak dnes je již možné využívat jazyky C++, C#, Python a další. Jako vývojové prostředí pro takovéto aplikace se využívá software GNOME Builder. Zajímavostí je například to, že většina vývojářů platformy GNOME jsou dobrovolníci.

Pro vývoj Linux aplikací je možné použít také např. grafické knihovny KDE a vývojové prostředím KDevelop.

2.1.2 Webové aplikace

Oproti desktopovým aplikacím stojí velká, rychle se rozvíjející a čím dál tím častěji využívaná kategorie webových aplikací. Jejich vývoj je založen na jazycích pro tvorbu webových stránek, a to HTML, CSS, PHP a JavaScriptu. Nejedná se však o klasické statické webové stránky, ale o jednostránkové či vícestránkové interaktivní části, které běží ve webovém prohlížeči zobrazujícím obsah stránky a na pozadí vykonává logický kód. [1]

Pro využívání webových aplikací je tak zapotřebí internetového připojení, což v dnešní době není zas takový problém. Díky tomu mizí nutnost ruční instalace a manuální aktualizace, jelikož se zobrazuje vždy to nejnovější, co vývojář upravil a zveřejnil, aniž by pro to musel uživatel cokoli dělat. Speciálním případem jsou některé webové aplikace typu single-page[4] nebo PWA, které částečně eliminují internetové připojení, protože jim stačí pouze prvotní připojení a stažení všech dat do zařízení a následně mohou fungovat i offline [1].

Velkou výhodou webových aplikací je jejich multiplatformní pokrytí. Jelikož přistoupit k aplikaci lze skrze jakýkoli internetový prohlížeč a konkrétní URL adresu, je možné ji používat na jakémkoli zařízení s přístupem k internetu. Lze ji tedy využívat na počítačích, tabletech, mobilech atd. bez vývoje nových aplikací pro každou platformu zvlášť. Tím se stává vývoj takové aplikace mnohem méně náročný na čas a potřebné finance. Nevýhodou může být již zmíněné připojení k internetu nebo menší zisk z prodeje licencí k jednotlivé kopii softwaru [1]. To je dnes však ve webových aplikacích řešeno zobrazujícími se reklamami nebo jakoukoli formou předplatného pro plnohodnotné používání aplikace, které bývá často rozděleno do několik cenových kategorií, z nichž každá vyšší zpřístupňuje další nové funkce aplikace.

[4] Jednostránkové webové aplikace obnovující svůj obsah bez načítání nových stránek

Vývojový model

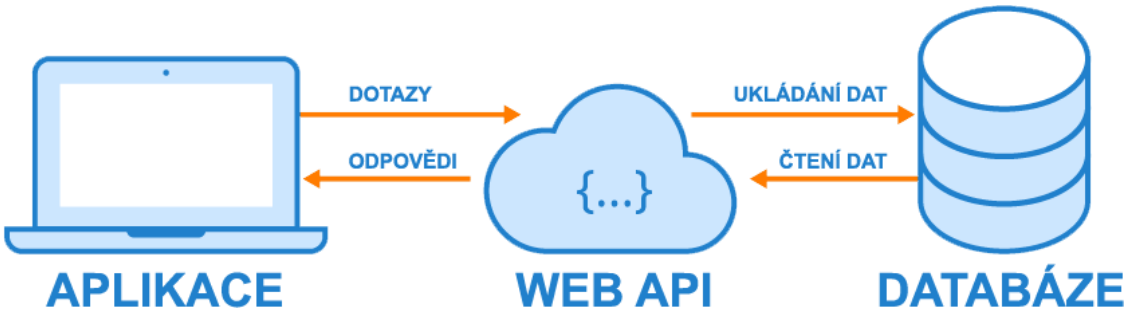
Vývoj webové aplikace je většinou rozdělen na tři hlavní samostatně fungující články, které při společném propojení vytváří finální proces (Obr. 2.1). Tyto části lze případně v budoucnu upravovat, zaměnit za jiné nebo k nim napojit další. Jedná se o front-end (uživatelskou aplikaci), back-end (server dodávající data) a databázi (uchovává data).

Front-endem se nazývá ta část, kterou vidí klasický uživatel. Jedná se o vizuální vzhled aplikace s ovládacími prvky, které využívá uživatel pro finální komunikaci se serverem. Front-endů napojených na jeden server může být několik a mohou vypadat zcela odlišně. Jako front-end může tak například fungovat i desktopová aplikace, která je nainstalovaná na počítači, ale data dostává z nějakého serveru. Po takové desktopové aplikaci se však již požaduje přístup k internetu.

Back-end je logická část aplikace. V tomto modelu jej představuje samostatný server tzv. Web API, který napřímo komunikuje s databází a dodává na uživatelskou aplikaci požadovaná data. Slouží jako komunikační propojovací uzel mezi jednotlivými aplikacemi (které mohou běžet na různých platformách a na jakémkoli zařízení) a databázovým serverem uchovávajícím data. Aby jednotlivé aplikace nemusely být zvlášť napojené na databázi, je na databázi napojené pouze toto Web API, se kterým všechny aplikaci následně komunikují a podávají mu různé dotazy. [7]

Databáze je pak místo, kde se shromažďují veškerá data jedné či více aplikací. Může ji představovat např. nějaký soubor nebo v našem modelu opět další server, který je propojen s Web API.

Klientská aplikace může využívat několik Web API najednou (i z jiných zdrojů, ke kterým má přístup) a vzájemně pak tyto získaná data různými způsoby kombinovat a pracovat s nimi. Komunikace klienta se serverem nejčastěji probíhá za dodržování architektury REST a pomocí jednoduchých HTTP volání. Pomocí toho je možné data na serveru vytvářet, mazat, číst a editovat. Web API následně dotaz zpracuje a provede nad daty požadované operace. Konkrétní URL adresy, které k tomu využívá, jsou pro každé Web API definovány v jeho manuálu.



Obrázek 2.1: Znázornění finálního procesu komunikace mezi jednotlivými částmi
Převzato z [8] a upraveno

Frameworky webových aplikací

K usnadnění vývoje webové aplikace existuje opět velké množství různých webových frameworků, které obsahují nespočet předepsaných komponent, šablon a částí kódu. Díky nim je vývoj rychlejší, stabilnější, drží se standartních praktik a umožňují vývojáři se především zaměřit na logickou stránku jeho aplikace. Frameworky je možné rozdělit na dvě kategorie viz Tabulka 2.1, avšak některé pokrývají obě části.

Tabulka 2.1: Porovnání front-end a back-end frameworků pro webové aplikace
Převzato z [9] a přeloženo

Front-end frameworky	Back-end frameworky
Část aplikace, která je viditelná pro uživatele	Logická funkční část aplikace
Klientské frameworky	Serverové frameworky
Zahrnuje návrh uživatelských ovládacích prvků a jejich vzhled, vytváření znovupoužitelných šablon	Zahrnuje správu databáze, zabezpečení, URL přesměrování, architekturu aplikace, ovládání serveru
Jazyky – HTML, CSS, JavaScript, JQuery	Jazyky – Python, JavaScript, PHP, Ruby, .NET
Frameworky – React, Vue, BootStrap, Ember, Angular	Frameworky – Django, Ruby On Rails, Express, Spring, ASP.NET / ASP.NET Core
Poskytují předpřipravené fragmenty kódu, opakovaně použitelné šablony, integrovatelné prvky a řídí interakci s uživatelem	Manipulace s databází, autorizace uživatelů, šifrování soukromí, opakovaně použitelné komponenty

Vybrané front-end frameworky

React

Jedná se o JavaScriptovou knihovnu pro vývoj uživatelského rozhraní webových aplikací, nejčastěji typu single-page, jelikož prosperuje v práci s rychle se měnícími daty. Byl vyvinut v roce 2013 společností Facebook (dnes Meta Platforms) jako další možnost konkurující dříve vzniklým frameworkům např. Angular a Ember [10]. Je založen na Node.js (softwarový systém pro webové servery) a npm (správci balíčků pro JavaScrip). Dnes je React jednou z nejpoužívanějších a nejžádanějších knihoven s širokou komunitou vývojářů.

Opět jde o sadu funkčních HTML komponent a předdefinovaných JavaScript souborů usnadňující vývoj aplikace. Tyto komponenty (components) jsou definovány svými vlastnostmi (props) a pracují se svým vnitřním stavem (state). Tyto jednotlivé komponenty se pak propojí a sestaví se z nich výsledný vzhled a ovládací prvky aplikace [10]. Na Reactu jsou postavené aplikace Facebook, Instagram, Netflix a další.

Vue

Vue je další z momentálně populárních a rychle se rozvíjejících frameworků pro front-end webových aplikací. Jedná se opět o JavaScriptovou knihovnu plnou dynamických uživatelských komponent, které pomohou uživateli rychle, efektivně a kvalitně sestavit plně hodnotnou aplikaci. Jde o progresivní framework, jelikož dovoluje vytvořit aplikaci s minimální snahou. Pozadí aplikace sám připraví a vývojář se tak může čistě zaměřit pouze na vizuální vrstvu. Pokud vývojář bude postupně přicházet s novými a složitějšími požadavky na aplikaci, může si přizpůsobit a připojit další externí knihovny. [11]

Tento framework využívá DOM (Document Object Model) webových prohlížečů. Jedná se o objektově orientovanou reprezentaci XML a HTML dokumentů. Takový dokument si framework vytvoří při aktualizaci jakýchkoli dat na stránce, porovná ho s tím aktuálním a poté aktualizuje pouze komponenty, které se v dokumentech liší, aby nemuselo docházet ke kompletnímu načtení celé stránky. Stejných principů využívá i zmíněný React. Vue využívají aplikace Gitlab, Adobe, Trivago a mnoho dalších.

Vybrané back-end frameworky

Express

Jedním z back-end JavaScript frameworků je Express. Jde o minimální a flexibilní Node.js webový aplikační framework, který disponuje velkou sadou funkcí pro webové a mobilní aplikace. Minimální proto, že při jeho používání vývojář začíná skoro z nuly a veškeré části si sám postupně připojuje dle potřeby, aniž by byl před něj postaven již nějaký rozsáhlý předem vytvořený projekt. [12]

Flexibilní je pak díky tomu, že ke komunikaci s klientskou aplikací právě používá čisté HTTP dotazy a skrze HTTP na ně odpovídá. Slouží tedy převážně jako framework pro Web API [12]. Express využívají PayPal, Uber, IBM a jiné.

ASP.NET / ASP.NET Core

Momentálně nejpopulárnější a nejpoužívanější framework dle statistik z [9] je ASP.NET od společnosti Microsoft, který je založen na platformě .NET. Je to nesmírně užitečný framework při vytváření živých dynamických webových aplikací a služeb pro počítače a mobilní telefony. ASP.NET Core je nová verze a je známá svou rychlostí, produktivitou, multiplatformním pokrytím a výkonem. Kromě kvalitního back-end frameworku lze využít k vytvoření kompletní plnohodnotné webové aplikace obsahující back-end i front-end částí. K tvorbě interaktivního uživatelského rozhraní na straně klienta využívá vlastní rozhraní Blazor a jazyk C#. Framework využívají například stránky StackOverflow, Microsoft, TacoBell, Dell a další.

2.1.3 Multiplatformní aplikace

Speciální kategorií jsou tzv. multiplatformní aplikace. Stejně jako webové aplikace, tak i tyto mají tu vlastnost, že stačí jejich vývoj provádět pouze jednou a aplikace je schopna fungovat na většině zařízeních. Aplikace se ale liší tím, že je uživatel schopen si ji nainstalovat jako desktopovou aplikaci na téměř jakékoli zařízení s různými operačními systémy. Některé frameworky umožňují aplikaci rovněž převést na webovou a spouštět ji skrze webový prohlížeč, stejně jako čistě webové aplikace.

Výhodou takovýchto aplikací je především ušetření si práce a financí, jelikož back-end a front-end není třeba vyvíjet na každou platformu zvlášť. O převod na ostatní platformy se postará samotný framework, nejčastěji převodem zdrojových souborů do strojového kódu. Jde tak o úctyhodný balanc mezi kvalitou a náklady. Pro firmy je tím také zajištěn vyšší zisk, jelikož je tak možné oslovit více uživatelů s různými platformami. Další výhodu je i jednoduchá následná správa celé aplikace. Veškeré aktualizace a opravy stačí provést na jednom kódu a změny se následně propíší do všech zařízení. Plusem je i jednotný vzhled aplikace na všech platformách a uživatel se tak rychle zorientuje i na zařízeních s jiným systémem. [13]

Ne všechny frameworky však umí dostat aplikace na veškerá zařízení a operační systémy. Nejčastěji se jedná minimálně o základní složení Windows, MacOS/iOS a Android. Je proto důležité se před samotným vývojem rozhodnout, pro jaké platformy by měla být aplikace určena a v jakém programovacím jazyce by měl vývoj probíhat. I ten je totiž pro každý framework odlišný a využívá rozdílné technologie.

Vybrané multiplatformní frameworky

React Native

React Native je nadstavba nad již zmíněném front-end frameworku React od společnosti Meta Platforms. Jedná se spíše o multiplatformní framework pro mobilní aplikace, avšak je možné v něm vytvořit aplikace pro operační systémy Android, iOS, MacOS, Windows, web a také pro televize se systémy Android TV či tvOS. Framework využívají například aplikace Tesla, Skype, Pinterest, aplikace společnosti Meta a jiné.

Vývoj s tímto frameworkem probíhá v jazyce JavaScript a dokáže ho převést na ostatní platformy, kde aplikace vypadají jako by byly napsané v jazyce určeném pro danou platformu. Jedná se tak na každé platformě o plnohodnotnou aplikaci, což je rozdíl oproti jiným multiplatformním JavaScript frameworkům, které pouze využijí obal aplikace, aby v něm zobrazily aplikaci jako webovou stránku. [14]

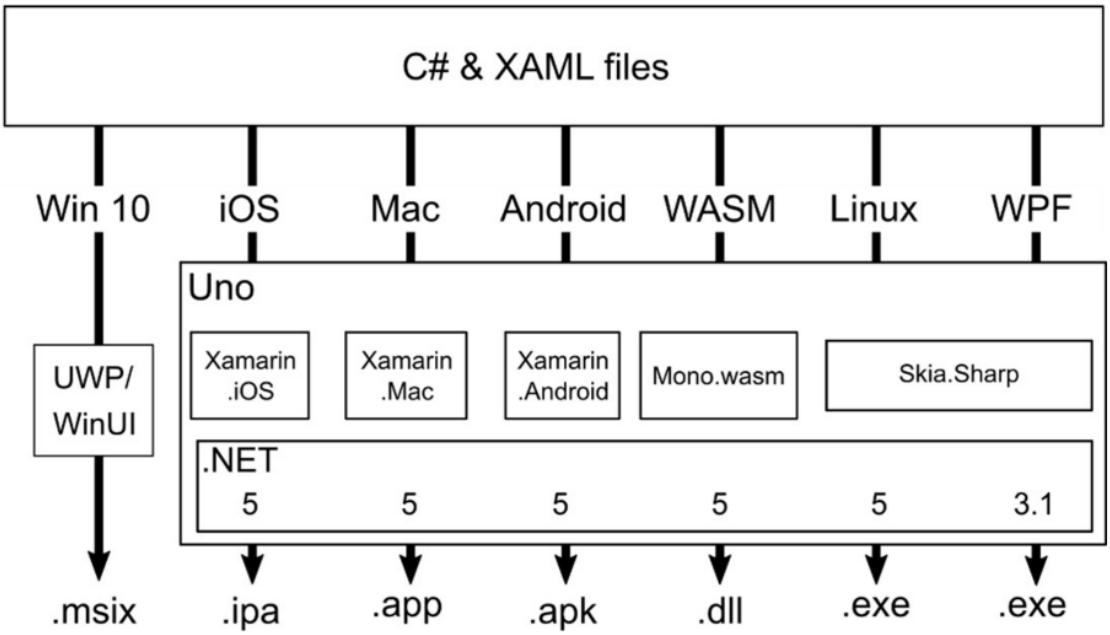
Xamarin a UNO Platform

Další možností, jak tvořit multiplatformní aplikace, je framework Xamarin. Tento framework se objevil již v roce 2011 a v roce 2016 byl získán společností Microsoft. Používá jazyk C# a framework .NET pro vytvoření aplikací na Android, iOS, MacOS a Windows. Zajišťuje komunikaci a převod C# kódu a navržených XAML front-end částí do originálního jazyka konkrétních platforem.

Xamarin je například využíván i jiným multiplatformním frameworkem, a to UNO Platform vyvinutý v roce 2018. Ten využívá Xamarin pro zmíněné platformy a pomocí jiných technologií je rozšiřuje i na několik dalších. To vše je umožněné díky rozsáhlému využití několika Windows API a jejich nástroji pro vývoj [15]. Základem je jednotný kód psaný v jazyce C# s grafickým rozhraním WinUI založeným na nezávislém vývojovém prostředí Universal Windows Platform (UWP) pro Windows 10. Front-end je opět tvořen jazykem XAML. Jelikož je tedy UNO založeno stejně jako Xamarin na C# a XAML, je v něm možné Xamarin využít pro převod na platformy Android, iOS a MacOS.

Operační systém Windows 10 je pokryt již od základu, protože jeho vývojové prostředí UWP je základním kamenem UNO a je následně převáděno na další platformy. Díky UNO má tedy aplikace vyvinutá v prostředí UWP možnost instalace a spuštění na platformách Windows 7 a 10, iOS, macOS, Android, Linux a WebAssembly. To vše je možné díky kompilaci[5] stejného kódu pro každou zmíněnou platformu (Obr. 2.2). Liší se tím například od jiných platforem, které před kompilací překládají kód do programovacího jazyka cílové platformy nebo převádí kód do HTML a Javascriptu, za účelem spouštět aplikaci ve webovém prohlížeči. [15]

[5] Převod algoritmu zapsaného v nějakém programovacím jazyce do strojového kódu



Obrázek 2.2: Vysokoúrovňová architektura platformy UNO. Převzato z [15]

UNO je tedy oproti Xamarinu rozšířeno o WebAssembly, Linux a WPF (Windows 7). WebAssembly zde nahrazuje klasickou čistě webovou aplikaci vytvořenou JavaScriptem a HTML. Využívá k tomu webový standard WebAssembly, který pomocí definovaného binárního kódu přenáší strojový kód do webového prohlížeče a následně ho v něm umí spustit. Pro Linux a WPF využívá UNO technologie SkiaSharp od společnosti Google. Skia je open-source[6] 2D grafická knihovna poskytující několik API dostupných napříč systémy. Zde je využita k překreslování každého pixelu navrženého front-endu do front-endu ostatních systémů, aby aplikace vypadal na všech zařízeních zcela totožně. Konkrétně SkiaSharp se zaměřuje na převod z .NET platforem. [16]

Xamarin využívají například aplikace UPS a BBC Goodfood. UNO pak využívají třeba aplikace Ch9, kalkulačka Windows 10 převedena na všechny systémy, Universal Azure DevOps App a další.

Electron

Zajímavým multiplatformním frameworkem je také Electron. Electron byl vyvinut a spravován GitHubem, původně jen pro textový editor Atom. Ke svému fungování využívá Node.js a vykreslovací jádro Chromium webového prohlížeče Chrome. Pomocí webových technologií vytváří čistě desktopové aplikace [14]. Na Electronu jsou postavené například velmi známé aplikace Visual Studio Code, Discord a Microsoft Teams.

[6] Otevřený software - počítačový software s veřejným zdrojovým kódem s volným použitím

Flutter

Za zmínku stojí i momentálně velice populární multiplatformní framework Flutter vytvořený společností Google. Jedná se především o framework pro mobilní aplikace na Android a iOS, avšak byl rozšířen i na desktop aplikace, web a vestavěné systémy[7]. Lze v něm tedy vytvořit aplikace i na Windows, MacOS, Linux, web a Google Fuchsia (nově vyvíjený operační systém od Google).

Vývoj ve Flutter probíhá pomocí vysokoúrovňového moderního programovacího objektově orientovaného jazyka Dart, který byl rovněž vyvinut Googlem. Flutter je znám svým vysokým výkonem a plnou kontrolou nad uživatelským rozhraním. Obsahuje plno grafických balíčků a k vytvoření uživatelského rozhraní na různé platformy využívá překreslení pixelů pomocí Skia, stejně jako UNO. Pro běh kódu na jiných platformách pak využívá Dart AOT kompilátor, která převádí kód do hlavního jazyk pro danou platformu. Tím se liší od jiných frameworků, které kód převádí do JavaScript a HTML nebo strojového kódu [17]. Mezi aplikace využívající Flutter patří například Google Ads, BMW, Google Pay a eBay.

2.2 Současná situace na klinice

Jak bylo již popsáno v úvodu, celá práce je zaměřena na konkrétní situaci a postupy plánování služeb na oční klinice Fakultní nemocnice Královské Vinohrady. Proto je tedy tato část věnována především aktuálnímu stavu procesu jejich tvorby rozpisů služeb, jaké byly pokusy to zlepšit a v čem jim tato práce pomůže.

2.2.1 Dříve

Stejně jako v jiných odvětví, tak i zde před nástupem jakékoli modernizace fungovala pouze metoda tužka-papír. Pro plánování služeb sloužili zapisovací knihy, dle kterých se všichni zaměstnanci řídili a dodržovali svou docházku. Postupem času se žalo modernizovat a již několik let se pro tyto účely využívá software Microsoft Excel s předdefinovanou strukturou tabulky. Za tu dobu prošla tabulka pár úpravami, avšak pouze co se týče vzhledu a ne funkčnosti.

2.2.2 Aktuálně

Aktuální plánování služeb tedy stále probíhá ve zmiňované tabulce v Excelu, jejíž vyplňování není nijak zautomatizované, nepoužívají se žádná předdefinovaná makra a rozmisťování zaměstnanců do jednotlivých kolonek je prováděno ručně. Vytvoření rozpisu tímto způsobem trvá až 3hod týdně a je náchylné na chyby. Je potřeba nahlížet

[7] Jednoúčelový počítač, ve kterém je řídicí systém zcela zabudován do zařízení, které ovládá.

do několika dalších tabulek, zda je ten den vůbec daná osoba přítomna, zda není nějaká akce nebo statní svátek a jiné nepříjemnosti.

Jedná se o jednoduchou excel tabulku (Obr. 2.3), ve které jsou sloupce dny v týdnu a řádky jednotlivá oddělení, která jsou ještě rozdělena na jednotlivé ambulance. Do jednotlivých políček, kde se protíná den s ambulancí jsou následně vyplňovány zkratky jmen jednotlivých zaměstnanců. Po vyplnění a několika kontrolách je tabulka rozeslána ostatním zaměstnancům a v případě nějakých nedostatků či žádostí o změnu je pak tabulka ještě několikrát přepracována.

BŘEZEN 2020		**pondělí 8.3.**	**úterý 9.3.**	**středa 10.3.**	**čtvrtek 11.3.**	**pátek 12.3.**
Lůžkové oddělení	lékař	Rozs do 12	Rozs do 12	Rozs do 12	*Nov* / Rozs od 10	Kraj + sál
MUDr.Penčák	lékař	*Klem*	*Klem*	*Klem*	*Klem*	*Klem*
Operační sály	1.sál	Pen/Str	Pen / *Komb*	Bru/Kap / *Kop*	Str	
MUDr.Straňák	zadní segment	PPV akutní	PPV	dětské	PPV	
	2.sál	Dítě/Ham	Klim/Kuch	Net ?	Net	Stu/Kraj
	přední segment	kat+roh	kat	kat+roh	kat+roh	kat
	3.sál				---	---
					dětské	malé výkony
Amb. všeobecná	amb. 1	**Ham do 12**	**Kon**	**Mag**	**Sok**	**Ham**
MUDr.Rejholcová	amb. 2	Kon	Šen	Kon	Kon	Ves
7,30 - 15,30h	amb.3	Kraj + aplikace	NemM	Šen	Kraj	Kon
	amb.4	*Nov*	*Baš*	*Nov*	*Baš*	*Baš*
odpo		YAG	YAG	ZAM		
Zadní segment	amb.2	Str	Str	Dítě	Dítě	Dítě
	amb.4	Dítě	Ern	Ern / *Baš*	Ern	---
	amb.5	Pen / *Kop*	Dítě / Kraj	Pen	Pen	Pen

Obrázek 2.3: Ukázka aktuálního rozpisu služeb na oční klinice
Ukázka obdržena od zaměstnanců kliniky

2.2.3 Snahy o zlepšení

Tuto situaci se klinika snažila už několikrát řešit. Nejprve byla snaha nalézt již existující aplikaci, která tyto funkce nabízí, jelikož takovýchto plánovačů je nespočet. Nepodařilo se však objevit aplikaci, která by se dokázala úspěšně přizpůsobit všem jejich požadavkům a funkcím, které si klinika představovala. Jednou z variant byla například aplikace DaySwaps, která se právě plánováním a docházkou zabývá [18]. Pan inženýr Pluhovský, který tuto modernizaci iniciuje, měl se zástupci aplikace v roce 2019 schůzku, na které probírali jejich možnosti a potřeby kliniky, ale bohužel se jim nepodařilo nalézt společné řešení, jelikož na klinice je mnoho podmínek a variability, které nebylo možné v DaySwaps zohlednit.

Podobně dopadly i jiné snahy o nasazení existujících aplikací, a tak přišla řada na vytvoření svého vlastního systému. Již několik dřívějších studentských prací se pokusilo tuto aplikaci vytvořit, avšak bohužel zatím neúspěšně bez jakéhokoli funkčního výsledku. K úspěšnému vývoji takové aplikace slouží tedy tato práce, která by měla danou problematiku vyřešit.

2.3 Požadavky na aplikaci

Hlavním požadavkem na vznikající aplikaci je tedy jednoduché a přehledné plánování služeb, které urychlí aktuální proces. Pro budoucí rozvoj by měla být aplikace připravena na dva přístupy, a to roli správce a veřejnou roli v podobě jakéhokoli zaměstnance.

2.3.1 Požadavky na role

Role lékař

Přístup role lékaře bude veřejný a možnost nahlédnout do aplikace bude mít jakýkoli zaměstnanec. Na domovské stránce se bude nacházet pro lékaře možnost zažádat o dovolenou, a to i za jiného zaměstnance, kdy vyplnění jména bude zcela volné a nezávislé na zaměstnancích v databázi aplikace. Součástí domovské stránky bude také přehled absence pro daný den.

Aby předávání vytvořených rozpisů nemuselo probíhat pouze externě (např. skrze email), měl by mít lékař přístup ke kompletnímu kalendáři v podobě zmíněné tabulky, díky které bude schopen nahlédnout do kompletního rozpisu služeb. Náhled bude čistě informativní a nebude možné se službami jakkoli operovat.

Role správce

Role správce by měla být schována za přihlašovacími údaji a sloužit pouze pro konkrétní osobu, která se stará o kompletní plánování služeb ostatních zaměstnanců na klinice. Proto je základní funkcí správce především možnost tvorby rozpisů služeb.

Samotné plánování by se mělo principiálně držet stávajícího rozložení rozvrhů, avšak přehlednější formou a s částečnou automatizací. Správce by měl mít možnost do rozpisu služby přidávat, mazat a upravovat. Během tvorby by měl rozvrh automaticky upozorňovat na případné kolize a nedostatky barevnou indikací. Na vyobrazených službách by mělo být na první pohled poznat, o koho se jedná, jaké kategorie zaměstnanec je, kterého dne a jaké části dne se služba týká a v jaké konkrétní ambulanci slouží. Ke každé službě by mělo být možné také přidat poznámku.

Rozpis by také mělo být možné nechat automaticky vygenerovat z předešlých týdnů nebo dle preferencí jednotlivých zaměstnanců a výsledný rozvrh pro každý týden exportovat do formátu PDF pro následné externí šíření mezi zaměstnanci či pro tisk.

Aby bylo možné rozpis sestavit a aplikace si zachovala svou variabilitu, je nutné, aby správce mohl spravovat veškerá data, kterými bude aplikace naplněna. Proto by aplikace měla obsahovat několik záložek, a to především záložku správy zaměstnanců, ve které půjde přidávat, odebírat a upravovat jednotlivé zaměstnance kliniky, filtrovat je podle kategorií, vyhledávat v seznamu podle jména a u každého zaměstnance bude možné zobrazit detailnější informace, jako jsou např. kontaktní údaje. V aplikaci bude

možné jednotlivým zaměstnancům přiřadit kategorie typu „Doktor“, „Sestra“ a další, případně vytvořit svou vlastní novou kategorii a nastavit i časové rozmezí působení dané osoby na klinice (např. využitelné pro kategorii „Stážista“).

Společně s tím je potřeba také záložka se správou jednotlivých oddělení a ambulancí, kdy opět půjde jednotlivá oddělení a k nim přiřazené ambulance přidávat, odebírat a upravovat. U oddělení i ambulancí bude možné nastavit minimální, optimální a maximální kapacitu, která se pak následně zohlední při vytváření rozpisu.

Další funkcí správce by měla být i kontrola nad dovolenými zaměstnanců. Jelikož má lékař možnost zažádat o dovolenou, je třeba, aby správce měl možnost tyto žádosti nějak zpracovávat. Z toho důvodu by měla aplikace obsahovat také správu dovolených pro jejich zakládání, odebírání a upravování s přehlednou filtrací podle časového rozmezí, stavu dovolené (schválená, zamítnutá, čeká na schválení, upravena) a jména zaměstnance, kterého se dovolená týká.

Po přihlášení do aplikace se zobrazí domovská stránka, na které se budou pro snadnější potvrzování dovolených zobrazovat jednotlivé žádosti s možností jejich rychlého schválení či zamítnutí. Současně se zde bude také nacházet absence zaměstnanců pro konkrétní den, stejně jako v případě role lékaře.

2.3.2 Technické požadavky

Rozhodování a domlouvání požadavků na technickou stránku aplikace nebylo vůbec jednoznačné, a nakonec i celkem komplikované. Prvotní technické a provozní požadavky na aplikaci byly, aby se jednalo o desktopovou verzi, která by případně fungovala i na bázi přenosné portable aplikace. Jednalo by se tak o aplikaci, která je uložena a spouštěna na přenosném disku (např. USB flash) a neukládá žádná data do počítače. Po pár prvních diskusí však z této varianty sešlo, jelikož se ukázalo jako výhodnější, aby do aplikace mohlo přistupovat více lidí naráz.

Z toho důvodu byl vznesen požadavek, aby aplikace zůstala desktopová, avšak data byla uložena v nějakém souboru na sdíleném disku, do kterého vidí více zařízení. Vývoj aplikace se tedy začal ubírat tímto směrem, ale při dalších diskusích se ukázalo, že jsou na klinice i počítače se staršími operačními systémy a bylo by dobré, aby i ty měly možnost aplikaci spustit. K tomu navíc byl následně vznesen požadavek, že přeci jen by se mohlo jednat případně i o webovou aplikaci s databázovým serverem.

Jelikož byl již vývoj aplikace v procesu a směřován na desktopovou verzi pro operační systém Windows 10, byly zvolené takové technologie, díky kterým byla rozpracovaná aplikace schopna tyto podmínky splnit a připravit aplikaci na další případná rozšíření.

3 Cíle práce

Finálním výsledkem této práce má být plně funkční aplikace, která převezme roli po softwaru Excel na oční klinice Fakultní nemocnice Královské Vinohrady a umožní kompletní plánování služeb zaměstnanců. To bude mít za následek ulehčení, zpřehlednění a urychlení celého procesu tvorby rozvrhů a rozsáhlé možnosti pro budoucí rozvoj systému.

3.1 Plánování služeb

Základním a nejdůležitějším cílem je zajistit možnost tvorby rozpisu služeb zaměstnanců. Aby však seznámení s novým procesem plánování bylo co nejjednodušší, měl by se postup plánování aspoň z části podobat momentálnímu postupu (viz 2 Přehled současného stavu). Proto je cílem vytvořit multifunkční interaktivní tabulku prolínající jednotlivé dny v týdnu, oddělení a ambulance, kdy pouhým klikáním bude možné jednotlivé služby vytvářet, rušit a editovat.

Pro příjemnější a rychlejší práci s plánovačem, je také jedním z cílů tabulku rozšířit o několik funkcí. Jedná se o různé barevné indikace, možnost rychlého přepínání mezi týdny, export vytvořených rozpisů do souboru PDF (pro následné sdílení mezi zaměstnanci) a automatická generace rozvrhů dle preferencí či dříve vytvořených rozpisů.

Součástí tabulky by měl být také prostor pro vytvoření nějaké události nebo svátku pro konkrétní den či rozmezí dnů, což umožní opět lepší orientaci v rozvrhu a vše je hned vidět na jednom místě.

3.2 Správa dat aplikace

Aby však plánování bylo ještě více interaktivní a kontrolovatelné, je dalším z cílů přidat do aplikace možnost správy zaměstnanců, dovolených, oddělení a ambulancí. Jedná se opět o základní operace typu přidávání, mazání a editaci s možností nastavení různých vyhledávacích filtrů. Zaměstnance bude možné vyhledávat pomocí jména nebo zkratky a filtrovat podle kategorie. Správa dovolených zas umožní vyhledávání zaměstnanců a filtraci podle stavu dovolené a rozmezí datumů, do kterých dovolené zasahují.

Dalším rozšířením bude domovská stránka aplikace, která umožní náhled nad absencí určitých zaměstnanců pro daný den a zobrazí se zde případné žádosti o dovolené, které čekají na zpracování.

3.3 Rozšiřitelnost aplikace

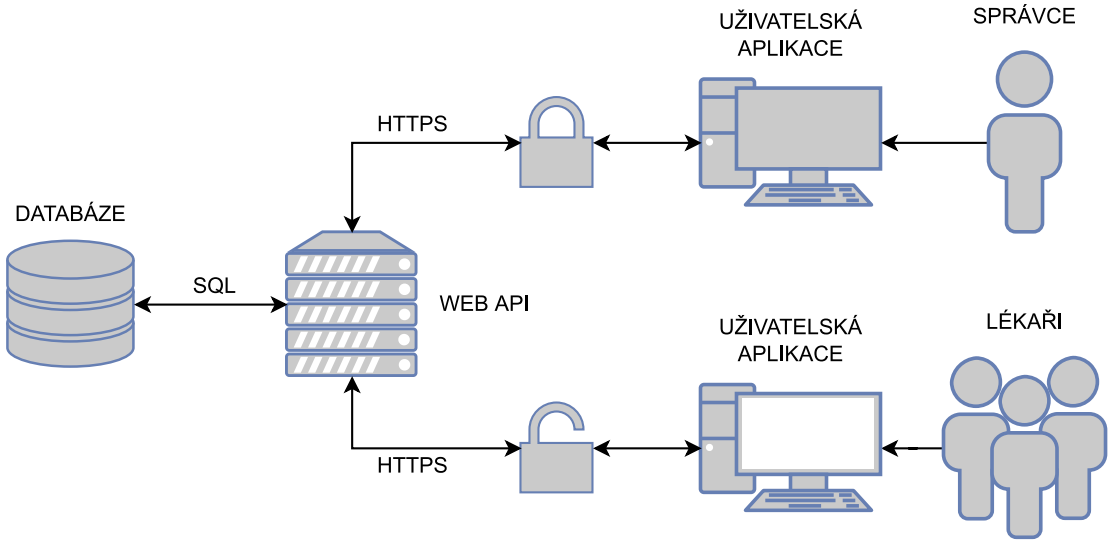
Dalším nemalým cílem je aplikaci připravit pro následné multiplatformní rozšíření a přístup z více zařízení naráz. Proto je cílem zvolit takové technologie, které tato rozšíření umožní.

Aplikace je zatím primárně určena pro jednoho člověka tvořící rozpisy služeb, avšak pro budoucí použití je cílem připravit aplikaci i pro základní dva přístupy, a to veřejný přístup v podobě lékaře a správce, který tvoří rozvrhy a spravuje veškerá data. Lékař má přístup pouze k náhledu vytvořených rozpisů, absenci ostatních zaměstnanců a možnost zažádat o dovolenou.

Dle požadavků kliniky by měl být přístup v podobě lékaře veřejný a možnost zažádat o dovolenou zcela univerzální, aby žádost mohla být po domluvě provedena i za jiné lékaře. Správce je pak schován za přístupovými údaji a má kontrolu nad celou aplikací.

4 Návrh aplikace

Tato kapitola se zaměřuje na kompletní návrh jednotlivých částí aplikace a jejich finální propojení. Návrh obsahuje popis požadavků na funkčnost aplikace, její možnosti a odůvodnění zvolených technologií pro vývoj. Aby bylo zajištěno větší multiplatformní pokrytí aplikace a sdílená práce s daty, byl po analýze požadavků zvolen vývojový model složený z databáze, Web API serveru a uživatelské aplikace, který je popsaný v kapitole 2.1.2 Webové aplikace a obrázkem 2.1. Tři hlavní podtémata této kapitoly tak představují jednotlivé části modelu (Obr. 4.1) a obsahují svou funkční a technickou specifikaci.



Obrázek 4.1: Návrh modelu aplikace (autor)

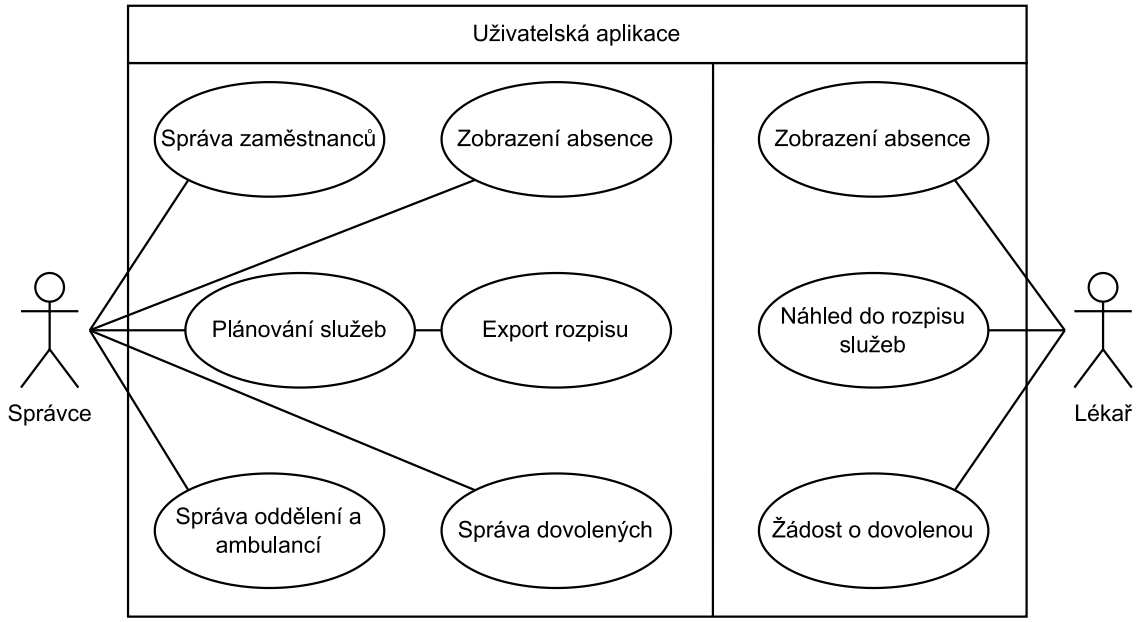
4.1 Uživatelská aplikace

Uživatelská aplikace je hlavní částí z pohledu uživatele, jelikož skrze ni celou aplikaci ovládá a prezentuje veškerá potřebná data. Během vývoje uživatelské aplikace došlo k několika náhlým změnám, díky kterým se návrh několikrát měnil (viz 4.1.2 Technická specifikace) a které nakonec vyústily v její finální podobu a rozsah.

4.1.1 Funkční specifikace

Vzhledem k požadavkům byl sestaven datový model, se kterým bude uživatelská aplikace dle návrhu pracovat. Ten odpovídá modelu databáze (Obr. 4.6) a datovému modelu Web API, aby dokázala na základě svých dotazů zpracovávat veškerá přijímaná data a byla schopna je prezentovat. Pro každý objekt z modelu bude tak vytvořena konkrétní třída s jejími atributy.

Hlavním cílem aplikace je tedy plánování služeb zaměstnanců na klinice. Jelikož však ideálním případem budoucnosti je, aby do aplikace měl přistup kdokoli a mohl se podívat, jak jsou služby naplánované, musí být samotné plánování a manipulace s daty v aplikaci nějak zabezpečeno pouze pro správce, který má tvorbu rozvrhů na starost. Z tohoto důvodu bude aplikace rozdělena na dvě základní role, „lékař“ a „správce“, a na základě nich bude následně kontrolován přístup k jednotlivým možnostem a funkcím aplikace (Obr. 4.2). Pod jakou rolí chce uživatel vstoupit si zvolí hned po spuštění.



Obrázek 4.2: Případy užití podle uživatelských rolí (autor)

Role lékař

V případě, že uživatel vybere v úvodním okně roli lékaře (viz 6 Uživatelská dokumentace), automaticky se dostane na její domovskou stránku (nástěnku). Tato role slouží totiž převážně jako veřejný přistup pro zaměstnance, aby mohli nahlédnout do vytvořených rozvrhů, a tak vstup pod touto rolí není nijak chráněn.

Pro lékaře bude menu aplikace (Obr. 4.5) obsahovat pouze dvě záložky. První z nich je domovská stránka, na které je k dispozici absence pro aktuální datum (rozdělená dle kategorií zaměstnanců) a možnost zažádat o dovolenou. Pro odeslání žádosti stačí vybrat její rozmezí a jméno zaměstnance z nabídky nebo vyplnit ručně jméno, které se z nějakého důvodu v databázi nenachází. Žádost se pak následně objeví správci, který s ní pak dále nakládá dle svých potřeb. Druhou a hlavní záložkou je přehled služeb, kde má zaměstnanec možnost nahlédnou na naplánované služby pro jakýkoli datum.

Role správce

Role správce má kontrolu nad celou aplikací, a tak bude po zvolení této role uživatel požádán o přihlášení. Po úspěšném přihlášení bude uživatel vpuštěn na domovskou stránku a menu aplikace bude obsahovat větší množství záložek. Domovská stránka obsahuje stejně jako u lékaře absenci zaměstnanců, avšak místo vytvoření žádosti o dovolenou jsou zde vypsány žádosti od zaměstnanců s možností okamžitého schválení či zamítnutí.

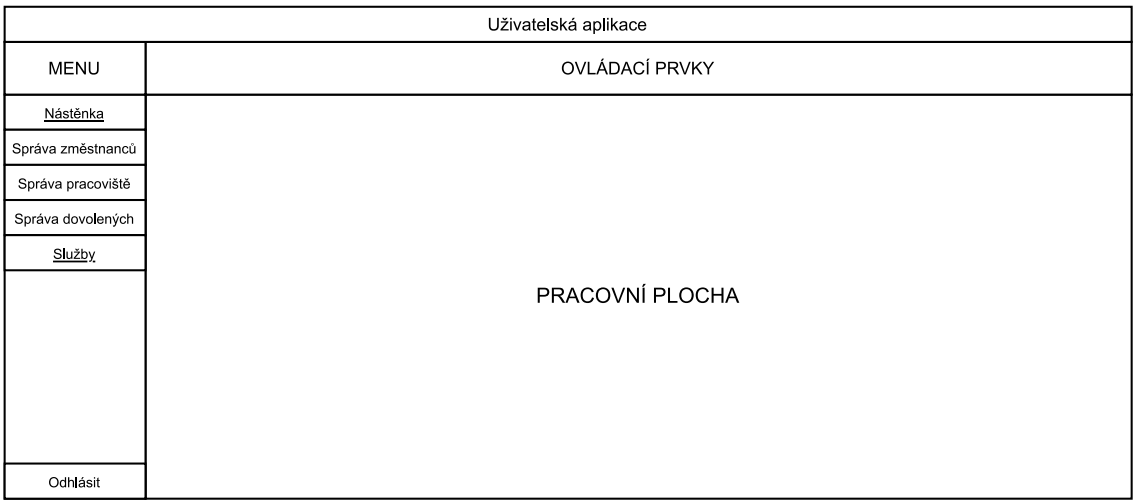
Následují záložky pro správu zaměstnanců, správu pracoviště a správu dovolených. Správa zaměstnanců slouží pro zobrazení, vytváření, mazání a úpravu zaměstnanců. Správa pracoviště slouží ke stejným operacím, avšak se týká oddělení a jejich ambulancí. Ve správě dovolených je pak možné dovolené vytvářet, mazat, editovat a prohledávat pomocí filtrů.

Hlavní záložkou jsou služby, kde má správce možnost sestavovat rozpis služeb pro konkrétní týden. Do vygenerované tabulky z oddělení a ambulancí prolínajících dny v týdnu má možnost vytvářet nové služby, mazat a měnit. Rozpis služeb je i možné kopírovat z předchozích týdnů a následně je případně pouze poupravit. Do rozvrhu lze také pro konkrétní dny přidat nějakou událost, která se ten den koná a je dobré na ni upozornit ostatní zaměstnance či jako upozornění pro správce při vytváření rozpisu. Pro případné šíření rozpisu mezi zaměstnanci (jinou cestou než skrze aplikaci) je rozpis možné exportovat ve formátu PDF.

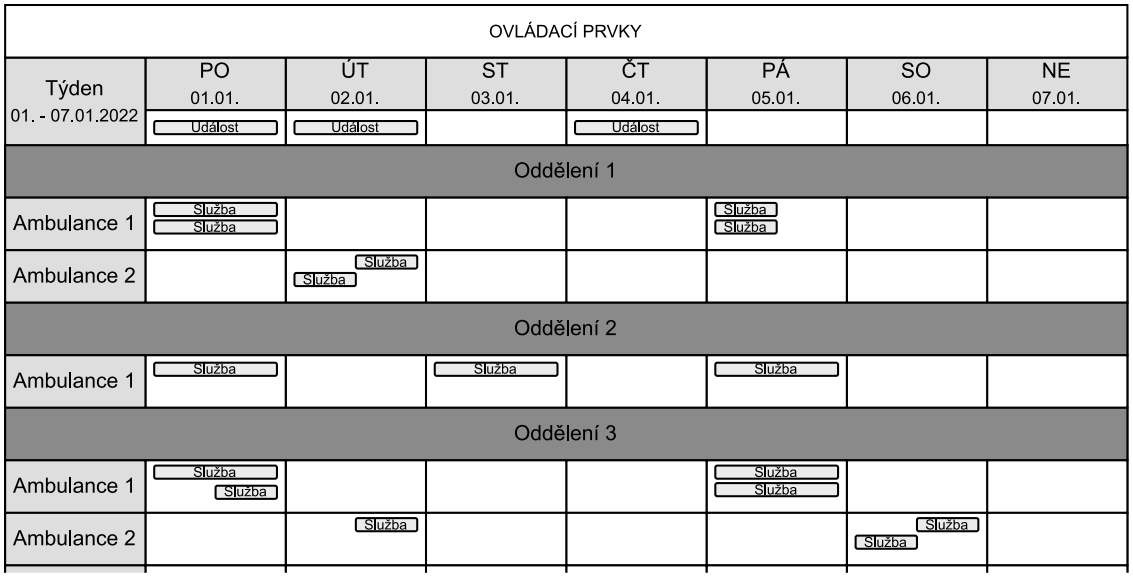
Rozložení uživatelské aplikace

Aplikace byla navržena jako jednostránková (Obr. 4.3), kdy se její obsah pracovní plochy mění na základě výběru položky v menu. Každá pracovní plocha odpovídající své záložce má pak svůj ovládací panel pro práci s daty, kterých se týká.

Pracovní plochu hlavní záložky „Služby“ bude tvořit tabulka, která by měla dle požadavků držet podobný základ jako aktuální tabulka z Excelu (Obr. 2.1), avšak lépe přehledná a obohacená o události. Byla tak navržena následující tabulka, která tyto požadavky splňuje (Obr. 4.4). Pro lepší přehled budou nápisy „Služba“ nahrazeny konkrétním jménem nebo zkratkou jména zaměstnance a barva služby bude odpovídat barvě kategorie zaměstnance. K rychlému určení, zda jde o celodenní, dopolední či odpolední službu, pak slouží odsazení služby vlevo (dopolední), vpravo (odpolední) nebo žádné (celodenní). Pro náhled do rozpisu v roli lékaře je tato tabulka zcela totožná, avšak má zablokované veškeré ovládací funkce, kromě listování dle datumů. Tabulka se stejným rozložením i barevnou anotací je taktéž vygenerována při exportu rozpisu do dokumentu PDF. Pracovní plochu ostatních záložek pak představují jednoduché seznamy či dlaždicové zobrazení, jejichž detailnější popis lze nalézt v kapitole 6 Uživatelská dokumentace.



Obrázek 4.3: Návrh grafického rozložení aplikace pro roli správce (autor)
(Role lékař má v menu dostupné pouze podtržené záložky)



Obrázek 4.4: Návrh rozložení plánovací tabulky služeb (autor)

4.1.2 Technická specifikace

Návrh a vývoj aplikace se za běhu několikrát měnil, což silně ovlivnilo finální výběr technických prostředků. Primární platforma, na které měla být aplikace provozována, byl operační systém Windows 10, jelikož ho využívá osoba, která bude primárně aplikaci spravovat a plánovat služby. Z toho důvodu byl zahájen vývoj uživatelské aplikace pomocí grafického .NET frameworku UWP s přímým napojením na databázi a základní jazyk byl zvolen C#.

Bohužel jsem se až později dozvěděl, že některé počítače na klinice, které by později mohli aplikaci využívat, fungují na operačním systému Windows 7 a aplikace by mohla být případně i webová. Jelikož framework UWP není na systémech Windows 7

podporován a vývoj byl již v průběhu, musela by se aplikace začít vyvíjet od nuly pod jiným frameworkem, aby bylo zajištěno multiplatformní pokrytí.

Tato komplikace byla však nakonec vyřešena díky multiplatformnímu frameworku UNO Platform (viz 2.1.3 Multiplatformní aplikace). Jeho základní kámen je právě UWP framework, který je následně pomocí UNO převáděn na ostatní platformy. Existující aplikaci UWP lze tak převést do frameworku UNO a není třeba zahajovat vývoj znovu, ale díky malým úpravám v něm pokračovat. Vývoj byl pak zaměřen primárně na platformy UWP (Windows 10), WPF (Windows 7) a WebAssembly (web), s možným budoucím rozšířením na další platformy, které UNO Platform podporuje. V době vzniku této práce, byl vydán nový operační systém Windows 11, se kterým UNO Platform také počítá a již umožňuje budoucí převod aplikace i do tohoto operačního systému.

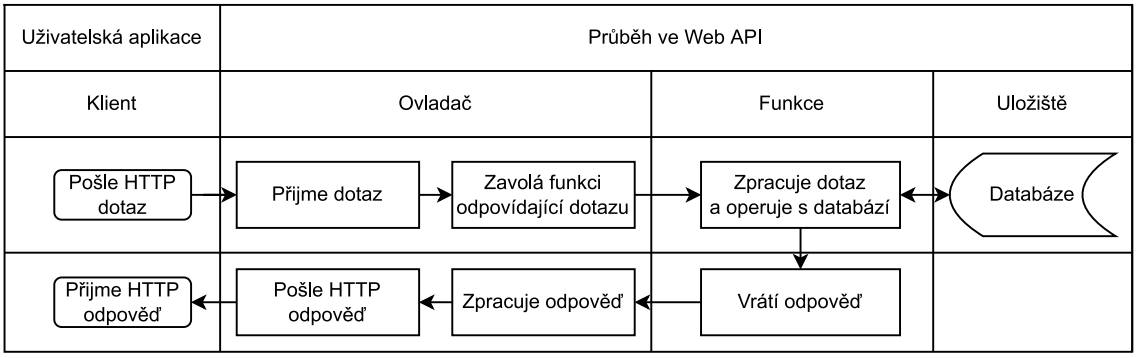
Přímé napojení na databázi bylo nahrazeno již zmíněným mezikrokem v podobě Web API. Aplikace s ním komunikuje skrze HTTP protokol na základě specifických dotazů a zpracovává přijaté odpovědi (viz 5.1 Web API).

4.2 Web API

Pro logický back-end a způsob práce s daty bylo dle vývojového modelu zvoleno Web API, které řídí komunikaci mezi databází a uživatelskou aplikací. Jak bylo již popsáno ve vývojovém modelu v kapitole 2.1.2 Webové aplikace, slouží Web API jako spojka mezi databázovým serverem a uživatelskou (klientskou) aplikací. Tento přístup byl do návrhu vybrán jakožto nejlepší varianta pro zajištění rozsáhlého multiplatformního pokrytí dané aplikace. Web API je tak jako jediné propojené přímo s databází a veškeré uživatelské aplikace mohou skrze něj přistupovat úplně stejně k datům pomocí HTTP protokolu. Databáze tak nemusí být napřímo svázána konkrétně s každou klientskou aplikací, což bývá při vývoji multiplatformních aplikací velice komplikované.

4.2.1 Funkční specifikace

Datový model Web API by měl být totožný s uživatelskou aplikací a s modelem databáze (Obr. 4.6), jelikož s nimi bude propojen a měl by tak obsahovat stejné objekty a jejich atributy, které jsou následně svázány s těmi z databáze. Každý objekt bude poté obohacen o svou sadu funkcí a svůj ovladač, který dané funkce řídí na základě přijatých HTTP dotazů z klientských aplikací (Obr. 4.5). Navržené sady funkcí jednotlivých datových objektů ve Web API jsou vypsány v tabulce 4.1.



Obrázek 4.5: Schéma obecného průběhu ve Web API (autor)

Tabulka 4.1: Seznam a popis funkcí objektů ve Web API (autor)

Služba (shift)	CreateShift() DeleteShift() GetShiftById() GetAllShifts() UpdateShift() GetShiftsByDate() **GetShiftsForWeek()** **GetShiftsByAmbulanceAndDate()** CopyWeek() GenFromPref()	Vytvoření nové služby Odstranění služby Vrátí službu podle ID Vrátí všechny služby Upraví existující službu Vrátí služby pro konkrétní datum Vrátí služby pro konkrétní týden Vrátí služby pro konkrétní ambulanci a datum Překopíruje služby z jednoho týdne na druhý Generuje rozvrh z preferencí zaměstnanců
Zaměstnanec (employee)	CreateEmployee() DeleteEmployee() GetEmployeeById() **GetAllEmployees()** UpdateEmployee() **GetEmployeesWithNoShiftsForDate()** GetEmployeesByCategory()	Vytvoření nového zaměstnance Odstranění zaměstnance Vrátí zaměstnance podle ID Vrátí všechny zaměstnance Upraví zaměstnance Vrátí zaměstnance bez služeb pro daný datum Vrátí zaměstnance určité kategorie
Kategorie (empcategory)	CreateEmpCategory() DeleteEmpCategory() GetEmpCategoryById() GetAllEmpCategories() UpdateEmpCategory()	Vytvoří kategorii Odstraní kategorii Vrátí kategorii podle ID Vrátí všechny kategorie Upraví existující kategorii
Oddělení (department)	CreateDepartment() DeleteDepartment() GetDepartmentById() GetAllDepartments() UpdateDepartment() **GetAllDepartmentsWithAmbulances()**	Vytvoří oddělení Odstraní oddělení Vrátí oddělení podle ID Vrátí všechny oddělení Upraví existující oddělení Vrátí všechny oddělení včetně jejich ambulancí
Ambulance (ambulance)	CreateAmbulance() DeleteAmbulance() GetAmbulanceById() GetAllAmbulances() UpdateAmbulance() **GetAllAmbulancesWithDepartment()** AmbulanceMaxLoadForWeek()	Vytvoření nové ambulance Odstranění ambulance Vrátí ambulanci podle ID Vrátí všechny ambulance Upraví existující ambulanci Vrátí všechny ambulance včetně jejich oddělení Vrátí ambulance plné pro daný datum
Dovolená (vacation)	CreateVacation() DeleteVacation() GetVacationById()	Vytvoření nové dovolené Odstranění dovolené Vrátí dovolenou podle ID

	GetAllVacations() UpdateVacation() GetVacationsByFilter() **RequestVacation()**	Vrátí všechny dovolené Upraví existující dovolenou Vrátí dovolené podle filtru Vytvoření žádosti o dovolenou
Událost (calevent)	CreateCalEvent() DeleteCalEvent() GetCalEventById() GetAllCalEvents() UpdateCalEvent() **GetCalEventsForWeek()**	Vytvoření události Odstranění události Vrátí událost podle ID Vrátí všechny události Upraví existující událost Vrátí všechny události pro daný týden
Uživatel (user)	CreateUser() DeleteUser() GetUserById() GetAllUsers() UpdateUser() **UserExist()** **CheckConnection()**	Vytvoření nového uživatele Odstranění uživatele Vrátí uživatele podle ID Vrátí všechny uživatele Upraví existujícího uživatele Vrátí uživatele, pokud existuje v databázi Kontrola, zda je aplikace připojena k serveru

Jednotlivé funkce budou však dále rozděleny na ty, které jsou veřejně dostupné, a na ty, které jsou zabezpečené a může je využívat pouze role správce. Zabezpečené budou především funkce, které jednotlivé objekty vytváří, mažou a upravují. Funkce, ke kterým má přístup nepřihlášený uživatel, jsou v tabulce 4.1 zvýrazněny tučně a jedná se především o funkce potřebné pro nahlédnutí do rozvrhu, žádosti o dovolenou či přihlášení uživatele. Autorizace bude probíhat při každém dotazu od klienta a ovladač pak bude řídit, ke kterým funkcím umožní přístup. Tento krok se nachází v schématu u ovladače mezi „Přijme dotaz“ a „Zavolá funkci odpovídající dotazu“. Pokud ovladač nepovolí funkci zavolat, přeskočí rovnou na odeslání negativní HTTP odpovědi.

Web API bude též opatřeno o reportovací schopnost, kdy jsou veškeré případné problémy zaznamenávány dle datumu a času do textového souboru pro případnou kontrolu správného běhu serveru. Neklade také žádné vysoké podmínky na výkon aplikace/serveru, jelikož nepracuje s nijak enormním počtem dat a neprovádí žádné časově náročné algoritmy.

4.2.2 Technická specifikace

Jako framework pro Web API byl zvolen ASP.NET. Byl vybrán na základě dřívějších zkušeností s tímto frameworkem, a také proto, že jeho hlavní programovací jazyk je objektově orientovaný C#, ve kterém probíhal i vývoj uživatelské aplikace a opět s ním spojené dříve získané zkušenosti. Plusem je také to, že finální uživatelská aplikace je psaná primárně na operační systém Windows, konkrétně Windows 10 s technologií UWP, která funguje stejně jako ASP.NET na platformě .NET od společnosti Microsoft a tím je zajištěna lepší kompatibilita a totožné principy při vývoji.

Ke komunikaci Web API s databází byl využit open-source framework objektově relačního mapování (ORM) Entity Framework. ORM zajišťuje převod dat mezi relační

databází a objektově orientovaným jazykem, kdy jednotlivé tabulky v databází převádí na objektové třídy a s jednotlivými entitami v databázi pracuje jako s objekty. S databází se tak nekomunikuje skrze čisté SQL příkazy (lze je v Entity Framework však také použít), ale pomocí definovaných C# funkcí.

Veškeré třídy odpovídající tabulkám v databázi byli vytvořeny pomocí ADO.NET (.NET knihovna poskytující přístup k databázím), Entity Frameworku a ovladačů pro MySQL metodou Database First. Jedná se o metodu, kdy jsou třídy jednotlivých objektů automaticky sestaveny z předem vytvořené a napojené databáze a v ní definovaných tabulek.

V rámci vývoje bylo Web API vystaveno opět na lokálním serveru, avšak může být vystaveno přímo na serveru kliniky či na některých z nabízených online služeb.

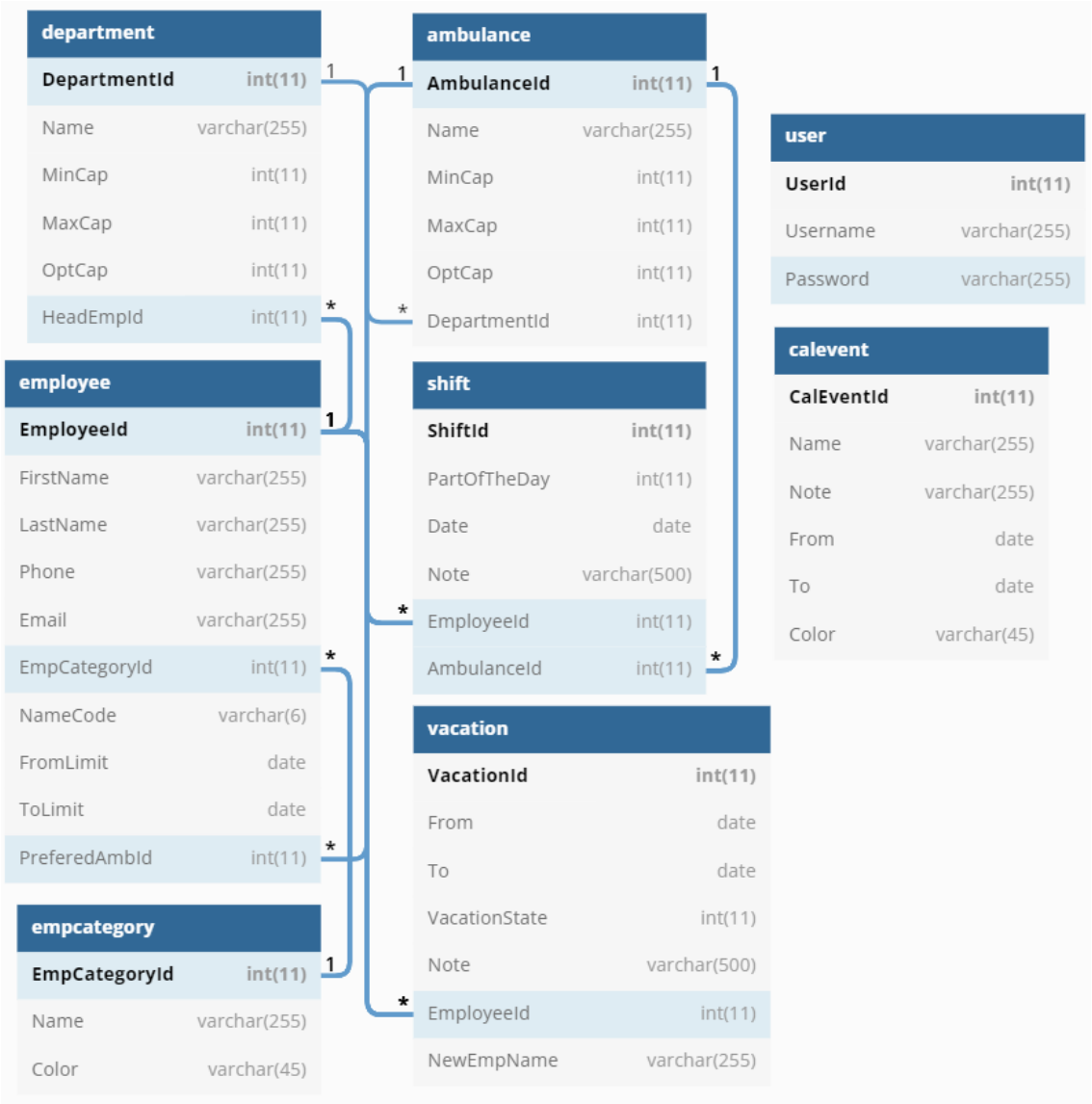
4.3 Databáze

V kapitole 2.3.2 byly popsány technické požadavky, ze kterých vyplynulo finální rozhodnutí pro založení samostatného databázového serveru, ke kterému bude přistupovat Web API.

4.3.1 Funkční specifikace

Datový model (Obr 4.6), se kterým by měla aplikace pracovat, se skládá z několika vzájemně propojených objektů. Hlavním objektem aplikace je služba (shift) zaměstnance. K definování služby je především zapotřebí objekt zaměstnanec (employee), kterého se služba týká a ambulance (ambulance), ve které má zrovna službu. Zaměstnanec je následně rozšířen o objekt kategorie zaměstnance (empcategory) a ambulance o objekt oddělení (department), ve kterém se nachází.

Součástí modelu jsou pak dále objekty uživatele (user), dovolené (vacation) a události (calevent). Objekt uživatele definuje účty, díky kterým se lze do aplikace přihlásit pod rolí správce. Objekt dovolené je propojen se zaměstnancem, kterého se dovolená týká, a definuje tak jeho pracovní volno. Událost je pak objekt definující nějakou akci, meeting, svátek či jiné události a umožňuje tak správci je do kalendáře zadat a upozornit na ně. Všechny objekty jsou dále rozšířeny o několik parametrů, které je více upřesňují a jejich přehled lze taktéž vyčíst z datového modelu (Obr 4.6).



Obrázek 4.6: Datový model aplikace (autor)

Jedná se o relační datový model, kdy jsou jednotlivá data v databázi rozdělena do jednotlivých tabulek (tzv. relací), které obsahují několik sloupců (tzv. atributy). Jde o jakési vlastnosti dané tabulky a každý atribut je dán názvem, typem a rozsahem dat (na obrázku 4.1 jsou atributy tabulek znázorněny řádky). Takto definované tabulky se pak plní jednotlivými daty, kdy každý řádek představuje jeden zápis a vytváří tzv. entitu, která je právě definována svými atributy (Obr. 4.7). [19]

EmployeeId	FirstName	LastName	Phone	Email	EmpCategoryId	NameCode	FromLimit	ToLimit	PreferedAmbId
1	David	Aron	606520975	aron.david@seznam.cz	1	DavA	0001-01-01	0001-01-01	1
2	Kateřina	Soumarová	123456789	katsou@gmail.com	2	KatS	0001-01-01	0001-01-01	3
3	Petr	Omáčka	456785259	pet.om@seznam.cz	1	Peom	0001-01-01	0001-01-01	4
4	Romana	Horká	656897542	roho@gmail.com	2	RomH	0001-01-01	0001-01-01	4

Obrázek 4.7: Ukázka zápisů do databáze (autor)
Tabulka zaměstnanců „*employee*“, jedná se o nereálné osoby (mimo autora)

4.3.2 Technická specifikace

Pro správu databáze byl zvolen open-source databázový řídící systém MySQL. Výhodou tohoto řídícího systému je především jeho open-source použití a také například oproti řídícímu systému SQL Server, který se používá pouze na operačních systémech Windows a Linux, umí fungovat na systémech Unix, Linux, Windows, MacOS a z/OS (operační systém společnosti IBM). [19]

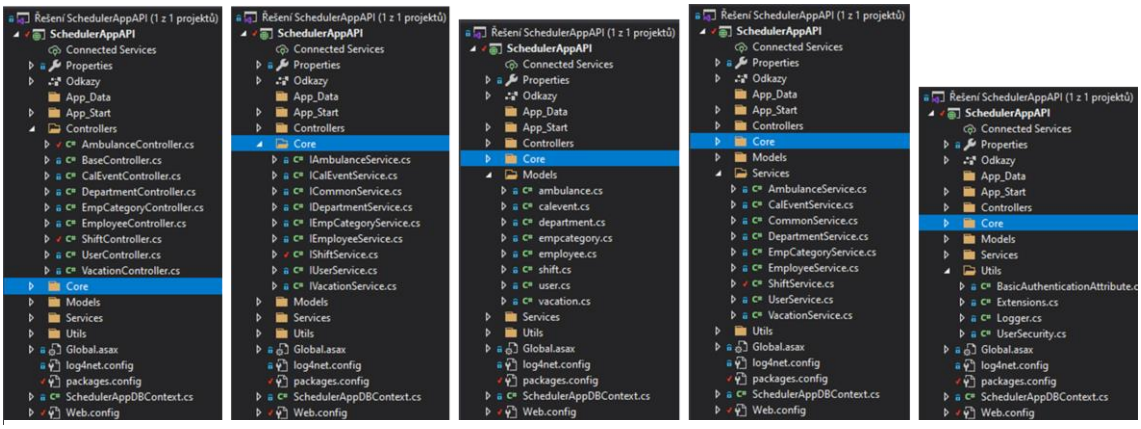
Ke komunikaci se serverovým řídícím systémem MySQL pracujícím s daty se využívá strukturovaný dotazovací jazyk SQL, stejně jako u většiny relačních databázových systémů. Tento jazyk byl využit také k založení a návrhu databáze v návrhovém nástroji MySQL Workbench. Během vývoje byl databázový MySQL server provozován na lokálním zařízení. Databázi pak lze však nasadit přímo na serverech kliniky nebo využít některou z online služeb, jako je například Microsoft Azure.

5 Implementace

Tato část práce je zaměřena na implementační popis dvou projektů, a to Web API a uživatelské aplikace. Součástí popisu každého z nich je rozdělení adresáře projektu, popis jeho částí a jak spolu souvisí, architektura projektu a některé zajímavé prvky a kusy kódů. Databáze v této části není zmíněna, jelikož se jedná pouze o založení databázových tabulek pomocí MySQL podle navrženého modelu (Obr. 4.6). Skript pro založení takové databáze se nachází na přiloženém CD.

5.1 Web API

Web API bylo vyvíjeno v prostředí Visual Studio pomocí frameworku ASP.NET a je šiřitelné pod licencí MIT. Adresář projektu (Obr. 5.1) se kromě složek, které byly vytvořeny automaticky, skládá ze složek *Controllers*, *Core*, *Models*, *Services* a *Utils*.



Obrázek 5.1: Adresář projektu Web API

Složka *Models* obsahuje veškeré modelové třídy odpovídající jednotlivým objektům z modelu databáze (Obr. 4.6). Tyto třídy byly vygenerovány po připojení databáze pomocí ADO.NET (viz 4.2.2 Technická specifikace). Třída propojující databázi a jednotlivé objekty se nazývá *SchedulerAppDBContext.cs* a nachází se v kořenovém adresáři.

Každá modelová třída je rozšířena svou sadou funkcí, které lze nad daným objektem provádět. O které funkce jde je sepsané ve složce *Core*, která obsahuje ke každé modelové třídě vlastní interface obsahující seznam funkcí pro konkrétní objekt. Dle těchto interface jsou následně sestavené jednotlivé service nacházející se ve složce *Services*, které následně strukturu těchto funkcí definují.

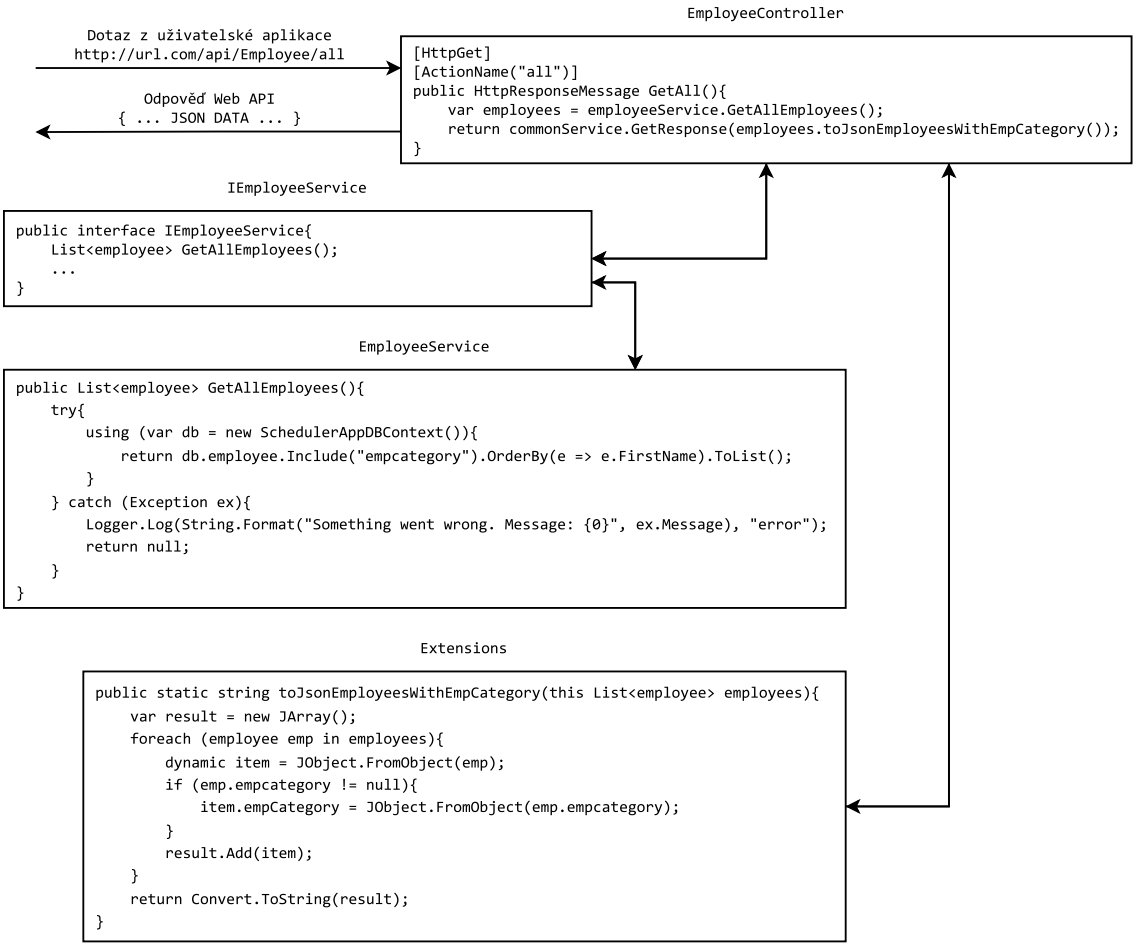
Funkce jsou následně volány v závislosti na svém controlleru, který je rovněž definován pro každou modelovou třídu a nachází se ve složce *Controllers*. Tento controller přijímá pomocí protokolu HTTP požadavky z klienta a podle nich pak volá požadované funkce nad modelovými třídami.

Poslední složkou jsou *Utils*. Ta obsahuje pomocné třídy pro zabezpečení (*BasicAuthentificationAttribute.cs, UserSecurity.cs*), třídy obsahující pomocné funkce na převody typů proměnných a validaci (*Extensions.cs*) a *Logger.cs* pro report případných chyb na serveru.

Na obrázku 5.2 je popsán průběh ve Web API po přijetí dotazu z uživatelské aplikace na získání všech zaměstnanců z databáze. Dotaz dorazí skrz síťový protokol HTTP v podobě URL adresy, kterou Web API zpracuje a zavolá odpovídající controller *EmployeeController*. Ten následně z odpovídajícího interface *IEmployeeService* funkci, která požadovaná data vrací, a zavolá ji z přidružené *EmployeeService*.

Ta z databáze získá všechny zaměstnance a vrátí je do controlleru. Controller však musí nejprve získaná data převést z objektů do jazyka JSON, který je zde využíván pro přenos dat skrze HTTP protokol. Proto nad daty zavolá funkci z Extensions, která tento převod provádí, a následně data odešle zpět na uživatelskou aplikaci.

Na obrázku 5.2 ve funkci z EmployeeService ke taktéž vidět využití reportovacího Loggeru, který v případě nějaké chyby zapíše do textového souboru, kdy a jaká chyba nastala.



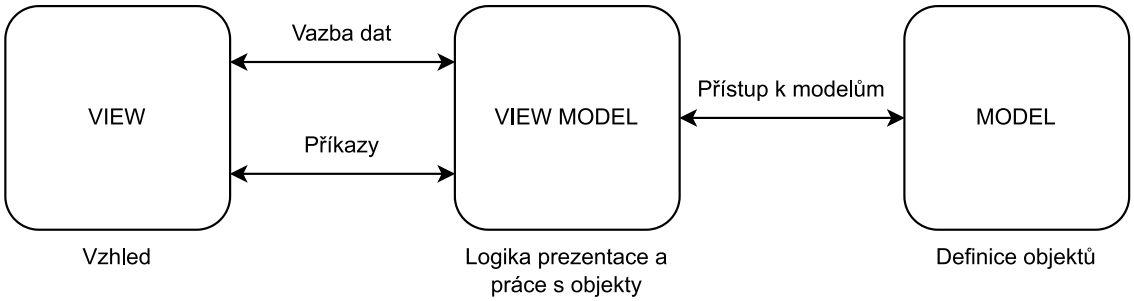
Obrázek 5.2: Průběh ve Web API po přijetí dotazu na získání všech zaměstnanců z databáze (autor)

K propojení controllerů, interface a service využívá Web API princip dependency injection pomocí knihovny Ninject. Jednotlivé třídy spolu nejsou napřímo svázány, ale dotazují se tzv. kontejneru na jednotlivé komponenty, které potřebují využít.

5.2 Uživatelská aplikace

Vývoj uživatelské aplikace opět probíhal ve vývojovém prostředí Visual Studio a finálním použitým frameworkem byl zvolen UNO Platform. Adresář projektu je rozdělen na jednotlivé spustitelné projekty pro jednotlivé platformy, které UNO Platform podporuje, a projekt s koncovkou „*Shared*“, ve kterém probíhá celý vývoj aplikace a ostatní projekty jednotlivých platforem z něj čerpají stejné zdrojové kódy a jednotlivé soubory, které si následně převádí pro svou platformu.

Mimo automaticky vygenerovaných souborů se v adresáři (Obr. 5.6) sdíleného projektu nachází složky *Models*, *Views*, *ViewModels*, *Controls*, *Converters*, *Dialogs* a *WebServices*. Hlavní složky *Models*, *Views* a *ViewModels* odpovídají základům vývojové architektury MVVM (Obr. 5.3), která byla pro tento projekt zvolena. Ten odděluje vzhled aplikace od back-end logiky a umožňuje tak jednodušší záměnu vzhledu bez závislosti na konkrétním modelu. Pro zjednodušení byla využita knihovna Microsoft MVVM Toolkit[8], která obsahuje předpřipravené třídy a funkce pro práci s MVVM. V této aplikaci je brán vzor spíše jako inspirace, a tak striktně nedodržuje všechny jeho náležitosti.



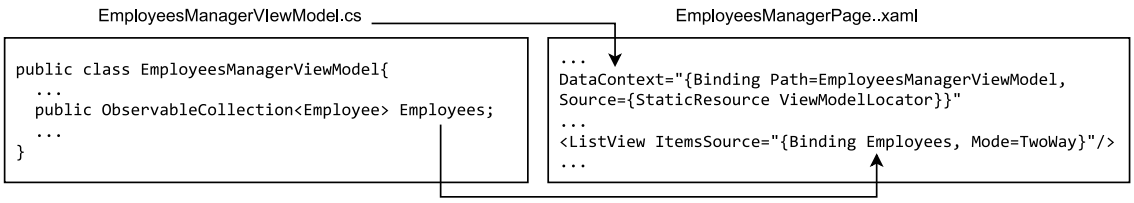
Obrázek 5.3: Schéma architektury MVVM (autor)

Složka *Models* obsahuje definice modelových objektů, se kterými aplikace pracuje a které odpovídají modelu databáze (Obr. 4.6). Je však rozšířena o dva výčtové typy, které definují, jaké číselné hodnoty určují konkrétní část dne a stav dovolené. V adresáři se nachází i složka *Controls* s modely některých ovládacích prvků, které rozšiřují základní modely a slouží pro snazší sestavování plánovací tabulky.

Složka *Views* následně obsahuje veškerý vzhled aplikace. Jsou zde pomocí jazyka XAML navrženy všechny záložky úvodní obrazovky aplikace. Vzhled jednotlivých vyskakovacích oken a formulářů je stejným způsobem definován ve složce *Dialogs*.

[8] Dostupné na https://docs.microsoft.com/en-us/windows/communitytoolkit/mvvm/introduction

Ve složce *ViewModels* se pak nachází všechny třídy zajišťující logiku a vazební data pro jednotlivé views. Součásti složky jsou také view modely jednotlivých dialogů ze složky *Dialogs*. Pro data z view modelu, která se mají na view zobrazit, je vytvořená tzv. vazba dat - Data Binding (Obr. 5.4). Jde o propojení proměnných kódu view modelu a grafického prvku ve view. Pro komunikaci se pak používají příkazy (Commands), které jsou opět pomocí vazby dat přiřazeny ovládacímu prvku, konkrétně tlačítku, a po je jejich zavolání je provedena nad daty odpovídající funkce.



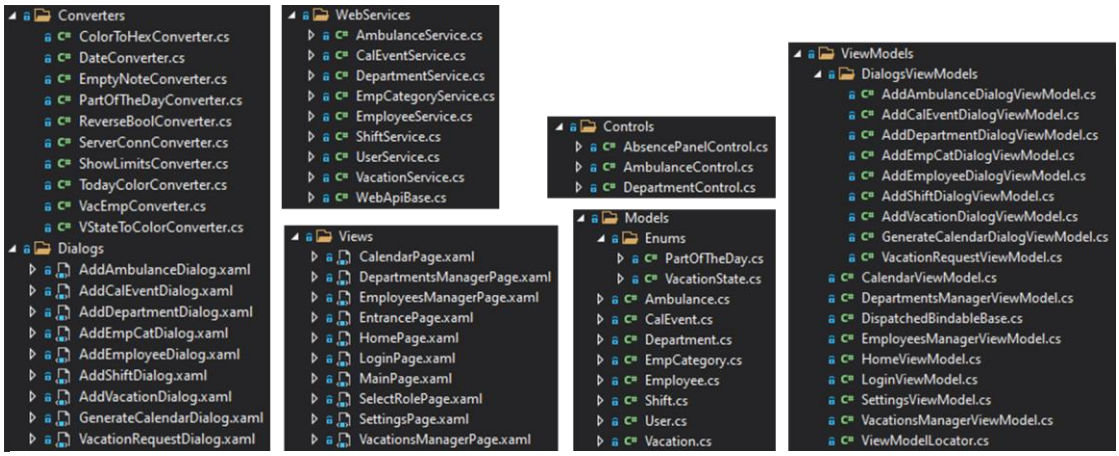
Obrázek 5.4: Ukázka napojení seznamu zaměstnanců pomocí vazby dat (autor)

K propojení vizuální stránky aplikace a view modelu napomáhají ještě tzv. konvertory, které se nachází ve složce *Converters*. Využívají se pro převod navázaných proměnných na jiné v závislosti na hodnotě dané proměnné. Například do barvy políčka je navázaná hodnota datumu dne, a pokud je shodná s dnešním datumem, konvertor ji vrátí růžovou a v opačném případě šedou.

Poslední složkou jsou *WebServices*. Třídy v této složce zajišťují komunikaci skrze HTTP protokol a posílají tak dotazy do Web API. Jedná se o třídu se základními funkce, které následně dědí ostatní třídy odpovídající objektům z modelu. V kódu 5.1 je ukázka funkce odeslání dotazu pro vrácení seznamu všech zaměstnanců, které předchází následnému průběhu ve Web API popsaném na obrázku 5.2.

```
...
public async Task<IEnumerable<Employee>> GetAllEmployees(){
    var result = await this.GetAsync(
                 url + "/Employee/all",
                 await GetHeaders());
    if (result != null){
        return JsonSerializer.Deserialize<IEnumerable<Employee>>(result);
    }
    return new List<Employee>();
}
...
```

Kód 5.1: Ukázka funkce pro odeslání dotazu na Web API
Zdrojové kódy pro komunikaci s Web API převzaty a upraveny z [20]



Obrázek 5.6: Adresář projektu uživatelské aplikace (autor)

Uno Platform umožňuje při psaní kódu rozlišit, jaký kód se má vykonat v závislosti na platformě, na které je aplikace spuštěna. Lze tak definovat, jaká funkce se má provést, pokud je aplikace spuštěna na Windows nebo jiné platformě. V této aplikace toho bylo využito například při definování klienta (Kód 5.2) pro HTTP komunikaci, který je pro webovou platformu rozdílný od ostatních. Toho podobného způsobu lze využít i při navrhování vzhledu aplikace pomocí XAML, kde mohu definovat, jak mají prvky vypadat na rozdílných platformách. To však nebylo v této aplikaci použito.

```
...
protected static HttpClient _client;

static WebApiBase(){
    #if __WASM__
        var innerHandler = new Uno.UI.Wasm.WasmHttpHandler();
    #else
        var innerHandler = new HttpClientHandler();
    #endif
    _client = new HttpClient(innerHandler);
}
...
```

Kód 5.2: Definování kódu pro specifickou platformu
Zdrojové kódy pro komunikaci s Web API převzaty a upraveny z [20]

WebAssembly je bohužel neideální variantou pro tuto aplikaci. Aplikace je založena na více vláknovém volání a častém obnovování komponent, což jedno vláknová a jednostránková WebAssembly aplikace zrovna dobře nesnáší. Je to také dáno tím, že WebAssembly neobsahuje garbage collector, který by uvolňoval paměť aplikace, a tudíž se brzy zahltí. Z toho důvodu není momentálně použitelná v reálném prostředí a pro web by mohla vzniknou čistě webová aplikace viz 7 Diskuse.

Export rozpisu

Pro export naplánovaného rozpisu do PDF využívá aplikace knihovnu iTetxt[9]. Jedná se o knihovnu pro jazyky Java a .NET, která slouží k vytváření a práci s PDF soubory. Obsahuje několik definovaných objektů, např. *Document*, *Paragraph*, *Table* a další, díky kterým je z dat následně sestavován finální PDF dokument (Obr. B.2). Tato knihovna je pod licencí AGPL, pod kterou musí být taktéž šířena, tudíž je pod touto licencí vedena i tato uživatelská aplikace.

Zabezpečení

Pro zabezpečený vstup do aplikace pod rolí správce se využívá autorizační hlavičky protokolu HTTP. Po zadání přihlašovacích údajů je na Web API poslán dotaz, zda tento uživatel existuje v databázi, a pokud ano, vpustí ho dále. V aplikaci se následně z uživatelského jména a hesla vytvoří tzv. token, který se přikládá do autorizační hlavičky ke každému dotazu na Web API, aby správce mohl využívat jeho zabezpečené funkce. Až do odhlášení či vypnutí aplikace je uchováván přihlášený uživatel v paměti a vzhled aplikace včetně přístupu k vybraným ovládacím prvkům se liší v závislosti na tom, zda je uživatel přihlášen či nikoli.

[9] Dostupné na https://itextpdf.com/en

6 Uživatelská dokumentace

V této kapitole je popsán postup zprovoznění všech částí aplikace a následná instalace uživatelské aplikace na rozdílné platformy. Součástí je také popis jednotlivých záložek uživatelské aplikace a jejich možnosti.

6.1 Zprovoznění databáze a Web API

Pro plnohodnotné fungování uživatelské aplikace je potřeba zprovoznit navrženou databázi a Web API, aby měla uživatelská aplikace s kým komunikovat a odkud brát veškerá data. Pro tuto práci byly pro prvotní testování a jednoduchost instalace založeny servery pro databázi a Web API na cloudové platformě Microsoft Azure a aplikace byla nastavena pro komunikaci s těmito servery. Stačí tedy pouze nainstalovat uživatelskou aplikaci (viz 6.2 Instalace uživatelské aplikace) a není potřeba žádná vlastní instalace databáze nebo Web API.

Lokální zprovoznění

V případě, že se aplikace k serverům nepřipojí, je třeba aplikaci otestovat přímo na lokálním zařízení. Pro databázi je nutné na zařízení nainstalovat MySQL[10] a pomocí skriptu *Database.sql* (na přiloženém CD) založit tuto databázi. Web API je pak nutné otevřít ve Visual Studiu[11] a v souboru *Web.config* změnit připojovací řetězec pro nově založenou databází na lokálním zařízení. Web API lze spustit přímo z Visual Studia nebo publikovat na nějaký založený server. Uživatelskou aplikaci není však následně možné pouze nainstalovat. Je nutné ji opět otevřít v prostředí Visual Studio a ve složce *WebServices* v souboru WebApiBase.cs změnit hodnotu proměnné „url“ na URL adresu, na které je vystaveno Web API. Aplikaci lze pak spustit přímo ve Visual Studio nebo ji publikovat jako instalační balíček (viz 6.2 Instalace uživatelské aplikace).

6.2 Instalace uživatelské aplikace

Jelikož UNO převádí UWP aplikaci na ostatní platformy, má to za následek i změnu v následné instalaci uživatelské aplikace pro konkrétní systémy. Pokud otevřete kompletní projekt uživatelské aplikace (složka „Zdrojové kódy\FNKVScheduler“) ve vývojové prostředí Visual Studio, je možné pak vybrat v seznamu projekt pro konkrétní platformu a provést vlastní publikaci (kliknutí pravým tlačítkem na vybraný projekt → „Publikovat“), avšak to je doporučeno pouze osobám, které mají s těmito všemi prostředky zkušenosti.

[10] Dostupné na https://dev.mysql.com/downloads/installer/

[11] Dostupné na https://visualstudio.microsoft.com/cs/

Pro každou platformu existuje několik možností, jak aplikaci nasadit a připravit do provozu. Níže jsou popsané základní a předpřipravené instalace. Doporučenou verzí je verze pro systém Windows10, která je nejlépe optimalizovaná. Pro testovací provoz byl v databázi založen provizorní účet s přihlašovacím jménem „admin“ a heslem „admin“.

6.2.1 Windows 10

Pro instalaci aplikace na operační systém Windows 10 slouží instalační složka s názvem „FNKVScheduler_W10“, která se nachází na přiloženém CD ve složce „Instalační balíčky“. Ve složce se nachází veškeré potřebné soubory pro instalaci a PowerShell soubor s názvem „Install.ps1“. Ten je třeba spustit pomocí konzole PowerShell (kliknutí pravým tlačítkem myši na soubor → „Otevřít v programu“ → „Windows PowerShell“). Následně postupujte dle zobrazených pokynů v konzoli. Pro instalaci je potřeba mít povolené instalování aplikací z neznámých zdrojů, k čemuž Vás spuštěný skript navede. Zeptá se také na heslo k certifikátu aplikace, které však žádné není, tudíž stačí pouze potvrdit klávesou ENTER. Po dokončení instalace lze aplikaci nalézt v nainstalovaných aplikacích nebo pomocí vyhledávače na liště pod názvem „FNKVScheduler“.

6.2.2 Windows 7

K instalování aplikace na operační systém Windows 7 stačí pouze ze složky „Instalační balíčky“ (na přiloženém CD) zkopírovat složku „FNKVScheduler_W7“ a vložit si ji někam do svého počítače. Aplikace má veškeré potřebné soubory v této složce, která obsahuje i přímo spustitelný soubor „FNKVScheduler.exe“. Tímto souborem se aplikace přímo spustí a není třeba žádná složitá instalace. Lze tuto složku mít tím pádem i na přenositelném úložném zařízení (např. USB) a spouštět ji přímo z něj. Tento postup lze rovněž provést i na Windows 10. To je dáno tím, že framework WPF, do kterého byla aplikace převedena pro běh na Windows 7, má rovněž podporu i na Windows 10.

6.2.3 WebAssembly

Jak bylo již popsáno v 5.2 Uživatelská aplikace, není WebAssembly zcela optimalizované pro reálné použití. Je možné si ho však vyzkoušet pomocí otevření projektu ve Visual Studiu a následném spuštění podprojektu WASM. Aplikace lze případně publikovat pro nějaký externí server nebo na některou z nabízených online služeb. Výkon aplikace je také rozdílný na různých webových prohlížečích, jelikož každý nakládá s WebAssembly trochu po svém.

6.3 Práce s uživatelskou aplikací

Následující body se zabývají jednotlivými záložkami aplikace a jejich využitím. Jak bylo zmínění v 4.1 Uživatelská aplikace, ne všechny záložky jsou přístupné pro obě přístupové role a společné záložky také obsahují rozdílné funkce. Pod jakou rolí chce uživatel do aplikace vstoupit se rozhodne na úvodní stránce (Obr. 6.1), jejíž součástí je také prvotní kontrola, zda je aplikace připojena k serverům. Pokud je vybraná role lékař, uživatel je automaticky vpuštěn. Pokud však vybere roli správce, je uživatel vyzván k přihlášení (Obr. 6.2). Po úspěšném přihlášení je vpuštěn do aplikace.

Vzhled aplikace má dva režimy – světlý a tmavý. V jakém režimu je aplikace spuštěna se odvíjí od nastavení barevného motivu zařízení, na kterém je spuštěna. Ve Windows lze motivy přepnout v *Nastavení* → *Přizpůsobení* → *Barvy*. Doporučený a lépe optimalizovaný je však tmavý režim.

Obrázek 6.1: Výběr role po spuštění aplikace (autor)

Obrázek 6.2: Přihlašovací formulář pro správce (autor)

6.3.1 Domovská stránka (Nástěnka)

Nástěnka (Obr. 6.3) slouží k zobrazení absence zaměstnanců, kteří pro daný den nemají zapsané žádné služby. Náhled je zobrazen v jednotlivých blocích dle kategorií zaměstnanců. Dalším využitím nástěnky je základní práce s dovolenými, která se liší v závislosti na přihlášené roli. V případě lékaře je zde pomocí tlačítka „Zažádat o dovolenou“ možnost o dovolenou zažádat. V případě role správce tento prostor

nahrazuje zobrazení všech žádostí o dovolenou, které čekají na zpracování. Lze zde žádost ihned schválit nebo zamítnout a pro jiné operace je třeba zavítat do záložky „Správa dovolených“.

Obrázek 6.3: Domovská stránka/Nástěnka (autor)
(Kromě autora této práce se jedná o smyšlené osoby, případná shoda jmen je čistě náhodná.)

6.3.2 Správa zaměstnanců

Záložka správa zaměstnanců (Obr. 6.4) je určena ke správě osob, kterých se plánování služeb týká a je třeba je začlenit do rozvrhu. Zároveň tato záložka slouží jako přehled informací o těchto osobách včetně jejich kontaktů. V seznamu je možné vyhledávat podle jmen nebo filtrovat osoby dle jejich kategorie. Zaměstnance lze přidat pomocí tlačítka „Přidat zaměstnance“ na horní liště a následném vyplnění potřebných údajů v zobrazeném formuláři. Po vybrání některého zaměstnance ze seznamu se zobrazí jeho detail, ve kterém lze osobu také odstranit nebo upravit její údaje. V pravém dolním rohu se pak nachází správa kategorií zaměstnanců.

Obrázek 6.4: Záložka správy zaměstnanců (autor)
(Kromě autora této práce se jedná o smyšlené osoby, případná shoda jmen je čistě náhodná.)

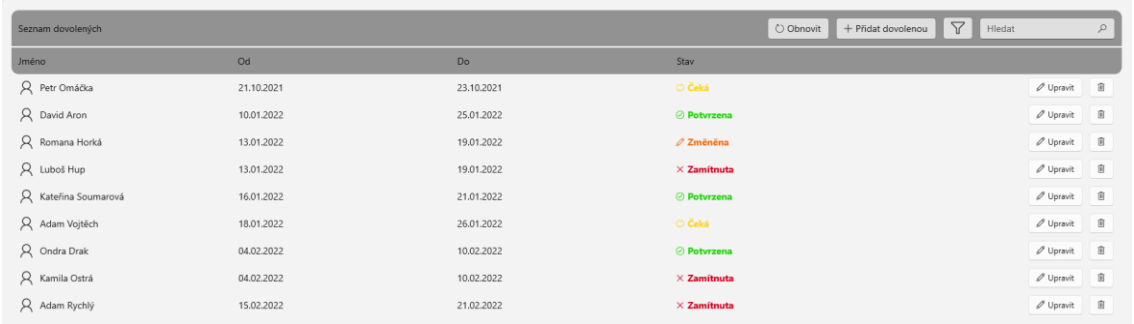
6.3.3 Správa pracoviště

Správa pracoviště (Obr. 6.5) se týká přehledu oddělení a ambulancí, které se na klinice nachází a ze kterých je následně tvořena plánovací tabulka. Oddělení graficky představují jednotlivé bloky, které obsahují další mešní bloky představující ambulance, které pod dané oddělení spadají. Pomocí tlačítka „Přidat oddělení“ na horní liště a vyplnění požadovaných údajů lze založit nové oddělení. Tlačítka u každého oddělení pak představují možnost do oddělení přidat ambulanci, upravit ho nebo ho odstranit. Stejná tlačítka pro úpravu či odstranění se nacházejí i na každé ambulanci.

Obrázek 6.5: Záložka správy pracoviště (autor)

6.3.4 Správa dovolených

Pro manipulaci s dovolenými je určena záložka správy dovolených. Nachází se zde přehled všech dovolených, které jsou dané jménem zaměstnance, časovým rozmezím a svým stavem. V dovolených lze hledat podle jména zaměstnance nebo pomocí hromadného filtru, který se skládá z časového rozmezí, stavu dovolené a kategorie zaměstnanců. Novou dovolenou lze přidat pomocí tlačítka „Přidat dovolenou“ na horní liště a následném vyplnění požadovaných údajů. K editaci nebo odstranění konkrétní dovolené jsou určena tlačítka na pravé straně stejného řádku.



Obrázek 6.6: Záložka správy dovolených (autor)
(Kromě autora této práce se jedná o smyšlené osoby, případná shoda jmen je čistě náhodná.)

6.3.5 Služby

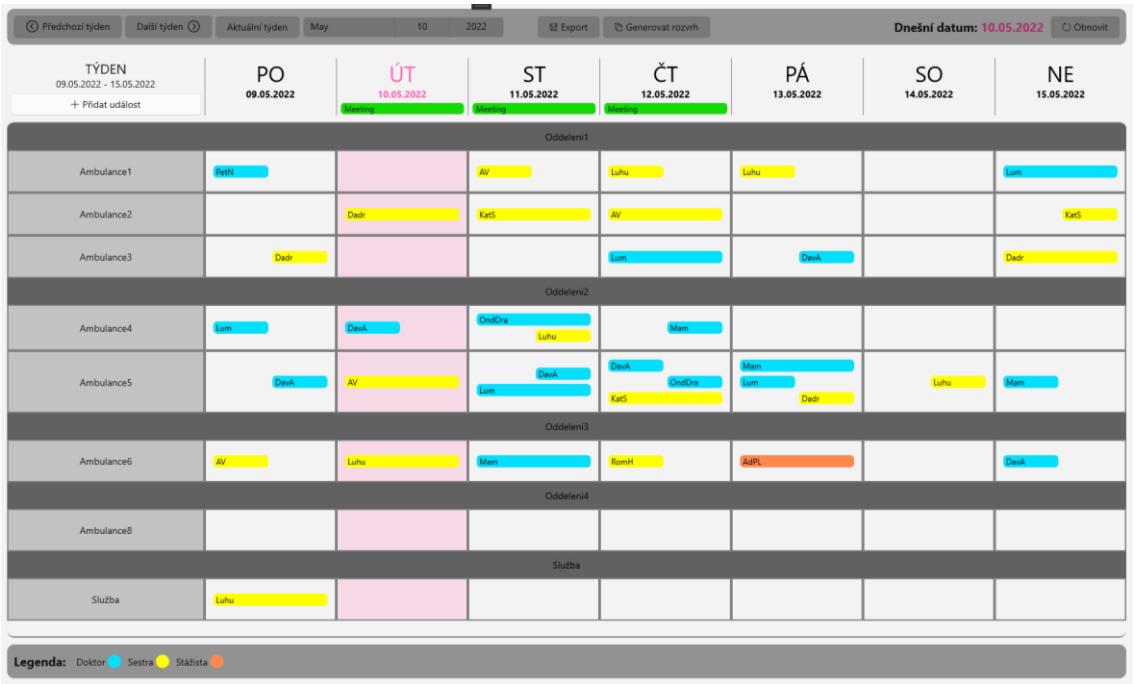
Hlavní záložkou jsou služby (Obr. 6.7), kde se nachází vygenerovaná tabulka protínají dny v týdnu a ambulance jednotlivých oddělení. V tabulce lze pomocí ovládacích prvku na horní liště přepínat mezi jednotlivými týdny, nalézt konkrétní datum nebo se vrátit na aktuální týden. Pro rychlejší orientaci je aktuální datum v tabulce barevně zvýrazněn.

Službu do rozpisu přiřadí správce pomocí kliknutí na určitě políčko v tabulce, kde se protíná požadovaný den a ambulance služby, a následném vyplnění potřebných údajů. Služba se tak zapíše do tabulky a po případném kliknutí na tuto službu se zobrazí její detail a možnost editace či odstranění. Název služby v tabulce je dán kódem jména zaměstnance, barva služby je závislá na kategorii zaměstnance a poloha služby v políčku je dána typem služby. Typ služby se rozděluje na celodenní (služba je přes celou šířku políčka), dopolední (služba je v levé polovině šířky políčka) a odpolední (služba je v pravé polovině šířky políčka).

Součástí horní lišty jsou také tlačítka pro případný export zobrazeného týdenního rozpisu do dokumentu ve formátu PDF nebo pro automatické vygenerování nového rozpisu. To je umožněno pouze pro prázdný týden a lze poté zvolit mezi kopírováním některého z předešlých týdnů nebo z preferencí zaměstnanců. Na spodní liště pod tabulkou se pak nachází legenda, která správci připomíná, jaké barvy jsou přiřazeny k jednotlivým kategoriím zaměstnanců.

Tabulka taktéž slouží k přidávání nových událostí k jednotlivým dnům. Události jsou zobrazovány těsně pod datumem každého dne. Jejich vzhled je dán názvem a zvolenou barvou při jejím vytváření. Práce s událostmi je vesměs stejná jako se službami. Po kliknutí na konkrétní událost se zobrazí její detail a tlačítka pro editaci nebo odstranění. Rozdíl je však v jejich vytváření, ke kterému slouží tlačítko „Přidat událost“ nacházející se v levém horním rohu tabulky a následném vyplnění požadovaných údajů. Způsob se liší proto, že událost může být nastavena oproti službě pro více dní.

Všechny záložky obsahují na horní liště tlačítko „Obnovit“, které slouží k znovu načtení záložky a aktualizaci zobrazených dat. Při práci se záložkami nebo při přepínání mezi nimi však k automatickým obnovám průběžně dochází.



Obrázek 6.7: Záložka služeb (autor)

7 Diskuse

Cílem této práce bylo navrhnout a vytvořit multiplatformní aplikaci na plánování služeb pro oční kliniku Fakultní nemocnice Královské Vinohrady, dle jejich požadavků a představ, a připravit aplikaci na případný budoucí rozvoj. Všechny tyto cíle byly splněny a aplikaci je možné nasadit do zkušebního provozu a díky poznatkům během používání ji případně doladit. Nasazení však bude nejspíš trochu komplikované z důvodu ne úplně ideální komunikace s IT oddělením kliniky a domluvě na samotném provedení. Jednou z možností je tak nasazení databáze a webového API na některé z nabízených online služeb a aplikaci by správce využíval k plánování soukromě.

Aplikace zajišťuje multiplatformní spuštění na systémech Windows 10, Windows 7 a WebAssembly. Díky vývoji pomocí frameworku UNO Platform má aplikace možnost jednoduššího budoucího rozšíření i na další platformy. WebAssembly je z těchto tří vybraných systémů bohužel výkonnostně nejslabší z důvodů popsaných v kapitole 5 Implementace. Aplikace by si tak zasloužila v tomto směru více optimalizace.

Funkčnost aplikace se nakonec vyšplhala do vyšších rozměrů, než se původně předpokládalo, a tak v některých směrech není zcela ideální. Vylepšení aplikace by se mohlo týkat například jejího grafického designu, zlepšení negativní zpětné vazby od Web API, zvýšené automatizace či optimalizace, zjednodušení kódu některých funkcí nebo využití dependency injection. Funkce, o které by aplikace mohla být rozšířena, jsou například uživatelské účty s přehledem jen svých služeb a dovolených nebo rozesílání různých upozornění skrze email. Web API by pak mohlo být vylepšeno o podrobnější a konkrétnější report událostí, vylepšení negativní odpovědi aplikacím a silnější zabezpečení, které by bylo v případě využití aplikace pro více účtů nutností.

Pokud by již nezačal původní vývoj čistě pomocí frameworku UWP, dle primárních požadavků pro operační systém Windows 10 a čistě desktopovou aplikaci, které se následně změnily a nejjednodušší cestou pak bylo využít framework UNO Platform, zvolil bych nejspíše jiný multiplatformní framework nebo aplikaci převedl pouze na webovou. Z toho důvodu jsem také začal s bonusovým vývojem čistě webového klienta (Obr. B.1) pomocí frameworku Vue (viz 2.1.2 Webové aplikace - Vue), který využívá stejnou databázi i Web API a mohl by tak sloužit jako další nástroj pro práci s těmito daty.

8 Závěr

Oční klinika ve Fakultní nemocnici Královské Vinohrady se již delší dobu potýká s časově náročným, a ne zcela přehledným plánováním služeb pomocí tabulkového softwaru Excel. Nepodařilo se nalézt existující výhodné řešení, a tak by jim tento problém měla pomoci vyřešit aplikace, jejíž návrh a vývoj probíhal v rámci této práce.

Dnes nalezneme mnoho způsobů, jak k vývoji aplikací přistupovat a jaké proto využít prostředky. V rámci této práce byli vybrány takové prostředky, které zajistily splnění požadavků kliniky a vznikla tak částečně multiplatformní aplikace pro plánování služeb s rozsáhlými možnostmi pro další vývoj, jak ve smyslu funkčnosti, tak i po technické stránce.

Samotné plánování služeb se díky aplikaci stává přehlednějším a více kontrolovatelným, k čemuž napomáhá různá barevná odlišnost, kontroly a částečná automatizace. Dávají tak správci aplikace více informací při tvorbě rozpisu než klasická tabulka. Aplikace nenabízí pouze plánování služeb, ale také kompletní správu zaměstnanců, pracoviště a dovolených, což více zpřehlední a usnadní plánování služeb, jelikož se veškeré potřebné informace pro sestavení rozpisu směn nachází na jednom místě.

K doladění aplikace by bylo vhodné lépe optimalizovat některé funkce a případně více oživit celkový vzhled aplikace. Aplikace je připravena pro případný budoucí rozvoj, kdy by mohla být například její funkcionalita rozšířena o další kontrolní funkce, uživatelské účty umožňující náhled nad vlastními daty, zasílání upozornění emailem a další. Po technické stránce může dojít k případnému rozšíření na další platformy či napojení dalších klientů a modulů do celkového vývojového modelu.

Zdrojové kódy Web API jsou dostupné pod licencí MIT a zdrojové kódy uživatelské aplikace jsou šiřitelné pod licencí AGPL z důvodů zmíněných v 5 Implementace.

Seznam použité literatury

[1] KOĎOUSKOVÁ, Barbora. Co je webová a desktopová aplikace a jaký je mezi nimi rozdíl? [online]. [cit. 2022-03-31]. Dostupné z: https://www.rascasone.com/cs/blog/desktop-web-aplikace

[2] What's a Universal Windows Platform (UWP) app? [online]. [cit. 2022-02-23]. Dostupné z: https://docs.microsoft.com/en-us/windows/uwp/get-started/universal-application-platform-guide

[3] Windows UI Library (WinUI) [online]. [cit. 2022-02-23]. Dostupné z: https://docs.microsoft.com/en-us/windows/apps/winui

[4] What is Cocoa? [online]. [cit. 2022-04-01]. Dostupné z: https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/CocoaFundamentals/WhatIsCocoa/WhatIsCocoa.html

[5] SwiftUI [online]. [cit. 2022-04-01]. Dostupné z: https://developer.apple.com/xcode/swiftui/

[6] Platform Introduction [online]. [cit. 2022-04-07]. Dostupné z: https://developer.gnome.org/documentation/introduction.html

[7] What is Web API? [online]. [cit. 2022-03-02]. Dostupné z: https://www.tutorialsteacher.com/webapi/what-is-web-api

[8] 7 Tips for Designing Easy-to-Understand REST APIs [online]. [cit. 2022-02-23]. Dostupné z: https://stabene.net/designing-easy-to-understand-rest-apis/

[9] 10 Popular web Frameworks for web App Development in 2022 [online]. [cit. 2022-04-07]. Dostupné z: https://www.monocubed.com/blog/most-popular-web-frameworks/

[10] WIERUCH, Robin. The Road to React: Your journey to master plain yet pragmatic React.js. 2020 Edition. 2020. ISBN 9781720043997.

[11] NELSON, Brett. Getting to Know Vue.js: Learn to Build Single Page Applications in Vue from Scratch. Apress, 2018. ISBN 1484237803.

[12] BROWN, Ethan. web Development with Node and Express: Leveraging the JavaScript Stack. 2nd Edition. 2019. ISBN 1492053511.

[13] SMITH, Ryan. Top 10 Best Cross Platform App Development Frameworks in 2022 [online]. 23.1.2021 [cit. 2022-04-12]. Dostupné z: https://www.codenameone.com/blog/top-10-best-cross-platform-app-development-frameworks-in-2022.html

[14] D. SCOTT, Adam. JavaScript Everywhere: Building Cross-Platform Applications with GraphQL, React, React Native, and Electron. 2020. ISBN 1492046981.

[15] Matt Lacey, Marcel Alexander Wagner, Creating Cross-Platform C# Applications with Uno Platform, ed. 1, Packt, 2021, ISBN 9781801078498

[16] SkiaSharp [online]. [cit. 2022-04-12]. Dostupné z: https://github.com/mono/SkiaSharp

[17] BIESSEK, Alessandro. Flutter for Beginners: An introductory guide to building cross-platform mobile applications with Flutter and Dart 2. 2019. ISBN 1788996089.

[18] DaySwaps [online]. [cit. 2022-03-17]. Dostupné z: https://dayswaps.com/

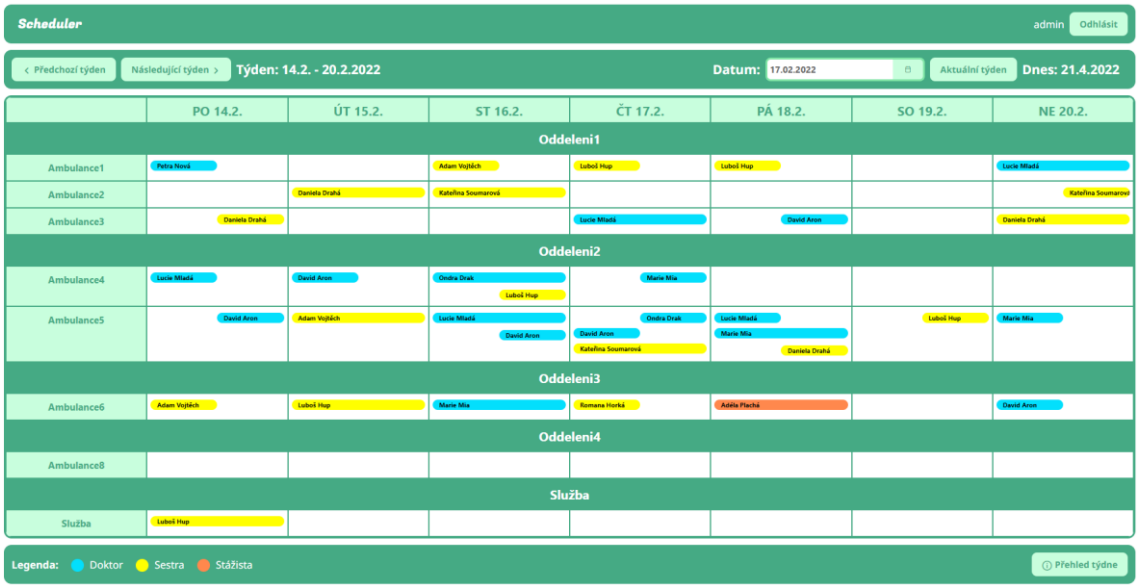
[19] MURACH, Joel. Murach's MySQL. 3rd Edition. 2019. ISBN 1943872368.

[20] UNO Platform: How to consume a web API [online]. [cit. 2022-04-28]. Dostupné z: https://platform.uno/docs/articles/howto-consume-webservices.html

Příloha A: Obsah přiloženého CD

- Repositář uživatelské aplikace
  - Zdrojové kódy
  - Licence (txt)
  - Uživatelský manuál (pdf)
- Repozitář Web API
  - Zdrojové kódy
  - Licence (txt)
- Skript na založení databáze
- Instalační balíčky pro jednotlivé platformy
- Repositář vznikající bonusové webové aplikace
- Klíčová slova (pdf)
- Abstrakt v českém jazyce (pdf)
- Abstrakt v anglickém jazyce (pdf)
- Zadání bakalářské práce (pdf)
- Bakalářská práce (pdf)

Příloha B: Dodatečné obrázky



Obrázek B.1: Design bonusového webového klienta (autor)

ROZPIS SLUŽEB

Týden: 14.02.2022 - 20.02.2022

	PO 14.02.2022	ÚT 15.02.2022	ST 16.02.2022	CT 17.02.2022	PÁ 18.02.2022	SO 19.02.2022	NE 20.02.2022
	Nová událost	Nová událost	Nová událost	Moje událost	Meeting2	Blabla	
Oddeleni1							
Ambulance2		Dadr	KatS	AV			KatS
Ambulance3	Dadr			Lum	DavA		Dadr
Ambulance1	PetN		AV	Luhu	Luhu		Lum
Oddeleni2							
Ambulance4	Lum	DavA	OndDra Luhu	Mam			
Ambulance5	DavA	AV	DavA Lum	DavA OndDra KatS	Mam Lum Dadr	Luhu	Mam
Oddeleni3							
Ambulance6	AV	Luhu	Mam	RomH	AdPL		DavA
Oddeleni4							
Ambulance8							
Služba							
Služba	Luhu						

Kategorie: Doktor Sestra Stážista

Obrázek B.2: Vygenerovaný PDF dokument po exportu rozpisu (autor)