České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra počítačů
Studijní program: Softwarové inženýrství a technologie

Mobilní aplikace pro plánování lidských zdrojů

Mobile App for Human Resource Planning

BAKALÁŘSKÁ PRÁCE

Vypracoval: Martina Kopecká
Vedoucí práce: Ing. Jiří Šebek
Rok: 2021

ČESKÉ
VYSOKÉ UČENÍ
TECHNICKÉ V PRAZE

Fakulta jaderná
a fyzikálně
inženýrská

Břehová 7 115 19 Praha 1

Název katedry

Akademický rok: rrrr/rrrr

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Jméno Příjmení**

Studijní program: Název programu

Obor: Název oboru

Název práce česky: **Název česky**

Název práce anglicky: Name in English language

Pokyny pro vypracování:

1. Úkol 1
2. Úkol 2
3. Úkol 3
4. Úkol 4

Insert your task scan here

Specimen only
Insert your task scan here

Doporučená literatura:

[1]

[2]

[3]

[4]

[5]

[6]

[7]

[8]

Literature

Jméno a pracoviště vedoucího práce: **Supervisor Name**
Supervisor institution

Jméno a pracoviště konzultanta: **Consultant Name**
Consultant institution

Datum zadání bakalářské práce: Date of issue

Datum odevzdání bakalářské práce: Deadline

Doba platnosti zadání je dva roky od data zadání.

V Praze dne Date

Speciaization supervisor Name

Signature

Department director Name

Signature

Stamp

Dean Name

Signature

Prohlášení

Prohlašuji, že jsem svou bakalářskou práci vypracovala samostatně a použila jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze dne .....

.....
Martina Kopecká

poděkování

Poděkování

Martina Kopecká

Název práce:
Mobilní aplikace pro plánování lidských zdrojů

Autor: Martina Kopecká

Studijní program: Softwarové inženýrství a technologie
Druh práce: Bakalářská práce

Vedoucí práce: Ing. Jiří Šebek

Abstrakt: Tato práce se věnuje návrhu mobilní aplikace pro plánování lidských zdrojů, především rozvrhování směn. V teoretické části jsou popsány základní rozhodující faktory a legislativní podmínky, které ovlivňují rozvrhování pracovních sil a způsob, jakým lze jejich rozvrhování automatizovat. Je stanoven předpoklad o cílové skupině organizací, pro které by byla aplikace na rozvrhování směn vhodná, a o jejich požadavcích. Na tomto základě jsou popsána vybraná softwarová řešení, je proveden rozbor možného nového řešení ve formě mobilní aplikace. V praktické části je představen vlastní triviální algoritmus na rozvrhování směn mezi zaměstnance, je popsán návrh aplikace včetně použitých technologií. Závěr je věnován uživatelskému testování aplikace a poznatkům z něj plynoucím.

Klíčová slova: Android, rozvrhování směn,obilní aplikace, Ruby on Rails

Title:
Mobile App for Human Resource Planning

Author: Martina Kopecká

Abstract: The objective of this bachelor thesis is to design a mobile application for human resource planning, mainly shift scheduling. In the theoretical part, main decision making factors and legislative standards that influence workforce scheduling and methods for scheduling automatization are described. The author forms a hypothesis about target group of organizations that might use the application for shift scheduling and their requirements. According to this assumption, there are described selected sofware solutions and an analysis of new solution is carried out. In the practical part, a trivial algorithm for shift scheduling is introduced and design of application is described, including description of used technologies. The final chapter of this thesis deals with user testing and its conclusions.
Key words: Android, shift scheduling, mobile app, Ruby on Rails

Obsah

Seznam použitých zkratek

UI	User Interface
OS	Operační systém
DPP	Dohoda o provedení práce
DPČ	Dohoda o pracovní činnosti
MVC	Model-View-Controller
MVVM	Model-View-ViewModel
REST	Representational State Transfer
API	Application Programming Interface
PaaS	Platform as a Service
DBMS	Database Management System
JSON	JavaScript Object Notation
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
XML	Extensible Markup Language
SQL	Structured Query Language
HTTP	Hypertext Transfer Protocol
MVP	Model-View-Presenter
RPC	Remote Procedure Call
ERP	Enterprise Resource Planning

Seznam obrázků

Úvod

Řízení lidských zdrojů je jednou z kritických součástí života každé organizace – úspěch stojí a padá na tom, že má organizace správné lidi na správném místě ve správném čase. Je tak potřeba, aby daná společnost měla dostatek dobře motivovaných a vhodně kvalifikovaných zaměstanců, které rozdělí tak, aby jí co nejvíce pomáhali. Tato práce si klade za cíl navrhnout uživatelsky přívětivou aplikaci, která by usnadnila jednu část tohoto rozsáhlého procesu – konkrétně rozvrhování směn mezi zaměstnance.

Tato práce v sobě kombinuje jak návrh softwarového systému pro rozvrhování směn, tak návrh rozvrhovacího algoritmu s cílem nalézt řešení pro alespoň malou část reálného problému, kterému může čelit jakýkoli manažer. Vycházelo se přitom ze zjednodušené představy menší společnosti (kavárny, restaurace nebo obchodu), jejíž interní procesy pro rozvrhování práce jsou nevhodně nastaveny. Rozvrh například sestavuje zaměstnavatel, jehož podpůrnými nástroji pro vícekriteriální manažerské rozhodování jsou papír a tužka, a prostředkem distribuce rozvrhu je nástěnka na pracovišti. Své procesy by chtěla organizace vylepšit nasazením software, aby ušetřila čas a zdroje a navíc i rozvrhla směny lépe. Cílem je navrhnout řešení ve formě mobilní aplikace, kterou by měli jak zaměstnanci, tak vedoucí neustále po ruce – vzhledem k rozvoji mobilní platformy a trendu růstu její oblíbenosti lze očekávat, že by taková aplikace mohla mít široké uplatnění.

Teoretická část této práce je věnována problematice plánování lidských zdrojů, procesu plánování pracovních sil a tomu, jaká kritéria jsou důležitá, a platné české legislativě, která tento proces ovlivňuje. Prostor je věnován problému rozvrhování směn jako vícekriteriální rozhodovací úloze, která je v literatuře nejčastěji popisována jako *problém rozvrhování zdravotních sester* (angl. nurse scheduling problem). Jsou představeny vybrané teoretické metody, které lze pro optimalizaci rozvrhu s řadou podmínek využívat. Je provedena analýza systému z pohledu organizace, nejprve je vytvořena hypotéza ohledně aktuálního a cílového stavu, poté jsou představena některá již existující softwarová řešení, která by problém, kterému daná firma čelí, mohla pomoci řešit. Na základě toho jsou sestaveny požadavky na nový systém, který by konkrétní potřeby uspokojil lépe, definováni jsou uživatelé (aktéři) a způsob jejich interakce se systémem (uživatelské příběhy).

V praktické části je popsána výsledná aplikace a návrh vlastního algoritmu. Je přiblížen výběr technologií pro vývoj aplikace a různé varianty, které připadají v úvahu, a také způsob, jakým jsou využity pro implementaci jak backendu, tak mobilní aplikace. Také je popsán způsob, jakým se sestavují rozvrhy směn v tomto projektu, včetně popisu a vyhodnocení vlastního algoritmu, který byl na základě teoretických poznatků navržen. Závěrečná kapitola je věnována uživatelskému hodnocení aplikace, tedy zda vůbec navrhované řešení dává uživatelům smysl, jak aplikaci po vyzkoušení hodnotí, případně jaká navrhují zlepšení.

Část I

Teoretická část

Kapitola 1

Analýza problematiky

1.1 Definice a terminologie

Řízení lidských zdrojů je definováno jako komplexní přístup k zaměstnávání a rozvíjení osob. Pojem v sobě obecně zahrnuje všechny aspekty toho, jak jsou osoby zaměstnány a řízeny v rámci organizace [5, s. 1], může se jednat jak o plánování pracovních sil a nábor nových zaměstnanců, tak i o jejich odměňování, školení nebo interní komunikaci mezi zaměstnanci. [5, s. 37]

Z hlediska této práce jsou důležité především pracovní síly a jejich plánování, tedy základní proces řízení lidských zdrojů, jehož smyslem je zajistit, aby společnosti pomáhal dosáhnout jejích cílů správný počet lidí se správnými schopnostmi, na správném místě, ve správném čase, za správnou cenu a ve správném pracovněprávním poměru. Mezi kroky tohoto procesu dle CIPD [11] patří:

- analýza aktuální personální situace;
- stanovení budoucích potřeb;
- identifikace současných nedostatků vzhledem k plánu do budoucna;
- provedení akcí k odstranění nedostatků;
- sledování a vyhodnocení dopadů akcí.

1.2 Rozhodující kritéria při plánování

1.2.1 Pracovní síly

V rámci organizace mohou existovat rozdíly mezi jednotlivými pracovníky. Jen část zaměstnanců tak bude pracovat na plný úvazek, jiní mohou pracovat méně hodin, například pouze v nejvytíženějších dnech [26]. V případě těchto zaměstnanců, kteří pracují méně hodin (a ne vždy pravidelně), je třeba zohlednit různé typy pracovněprávních vztahů, čemuž bude věnován prostor dále v podkapitole 1.3. Individualitu zaměstnanců je třeba zohlednit i z důvodu rozdílů mezi jejich schopnostmi, případně kvůli odlišným preferencím.

1.2.2 Směny

Jednotlivé organizace se od sebe mohou lišit způsobem, jakým vypisují směny. Mohou tak existovat podniky, kde je rozvrh práce pravidelný a stejný pro všechny zaměstnance, ale i

ty, kde je provoz dvousměnný, nebo i vícesměnný (tuto možnost připouští § 78 zákoníku práce [51]). Ve vícesměnném provozu může rozvrh být rotační (např. všichni zaměstnanci mají stejný základní rozvrh, pouze je začátek cyklu pro různé zaměstnance různě posunutý) nebo naopak necyklický (žádná pravidelnost neexistuje, sestavování rozvrhu je flexibilnější). [33]

Mezi organizace s pravidelně sestavovaným rozvrhem se nejčastěji řadí provozy, které fungují v nepřetržitém režimu (nemocnice, vězení, policejní služebny) nebo také obchody, restaurace a pobočky řetězců rychlého občerstvení. [7]

Nepravidelný rozvrh směn může mít pro zaměstnance nežádoucí zdravotní účinky, např. Flo [17] uvádí, že zaměstnanci ve vícesměnném provozu trpí nespavostí více než zbytek populace, a to především v případě, že je mezi směnami kratší než 11hodinová přestávka.

1.2.3 Předpověď poptávky po personálu

Zaměstnanci musí plnit úkoly podle toho, jaké události nastanou. Tyto události se musí organizace snažit předpovídat, očekávanou poptávku (přibližný počet zaměstnanců, jejich očekávané kompetence [5, s. 219]) je třeba modelovat. [16].

Příkladem může být prodejce hraček, který na základě historických dat ví, že nejvíce zákazníků přichází před Vánoci, a předpokládá, že tomu tak bude i v následujícím roce. Na tuto událost bude reagovat tím, že se rozhodne rozšířit otevírací dobu. Tím celkově zvýší poptávku po personálu v daném období.

1.2.4 Cena

Dalším důležitým kritériem při plánování pracovních sil je celková cena lidské práce a otázka její minimalizace při naplnění poptávky.

1.3 Legislativní podmínky

Hlavním právním předpisem, který upravuje problematiku pracovních sil, je v českém prostředí zákon č. 262/2006 Sb., zákoník práce, který upravuje vztahy vznikající při výkonu závislé práce mezi zaměstnanci a zaměstnavateli, tedy pracovněprávní vztahy [51].

Zaměstnavatel je dle § 38 zákoníku práce povinen přidělovat zaměstnanci práci podle pracovní smlouvy. Zaměstnanec je pak povinen podle pokynů zaměstnavatele konat osobně práci v rozvržené týdenní pracovní době.

1.3.1 Pracovní doba

Pracovní dobou se rozumí doba, v níž je zaměstnanec povinen vykonávat práci pro zaměstnavatele nebo je k tomu na pracovišti připraven. Doba odpočinku není součástí pracovní doby (§ 78).

Stanovená týdenní pracovní doba činí mimo výjimky 40 hodin (§ 79). Pracovní dobu rozvrhuje zaměstnavatel, který určuje začátek a konec směn, a to zpravidla do pětidenního pracovního týdne (§ 81). Délka směny nesmí přesáhnout 12 hodin (§ 83). U nezletilých zaměstnanců pak nesmí délka směny v jednotlivém dni překročit 8 hodin a v jednom týdnu 40 hodin.

Mezi zaměstnancem a zaměstnavatelem může být sjednána kratší pracovní doba (§ 80).

1.3.2 Rozvrh pracovní doby

Zaměstnavatel je dle § 84 zákoníku práce povinen vypracovat rozvrh týdenní pracovní doby a seznámit s ním nebo s jeho změnou zaměstnance nejpozději 2 týdny předem (mimo výjimky stanovené zákonem nebo v případě existence jiné dohody mezi zaměstnancem a zaměstnavatelem). Tento rozvrh musí být v písemné formě. Pracovní dobu je podle § 90 třeba rozvrhovat s ohledem na nepřetržitý odpočinek mezi koncem jedné směny a začátkem následující (pro zletilé zaměstnance alespoň 11 hodin, pro nezletilé zaměstnance alespoň 12 hodin, v zákonem stanovených výjimkách lze za určitých podmínek odpočinek zkrátit).

1.3.3 Směna, směnný provoz

Směnou se dle § 78 písm. c) zákoníku práce rozumí část týdenní pracovní doby bez práce přesčas, kterou je zaměstnanec povinen na základě předem stanoveného rozvrhu pracovních směn odpracovat.

1.3.4 Pracovní poměr

Pracovní poměr mezi zaměstnancem a zaměstnavatelem se podle § 33 odst. 1 zákoníku práce zakládá pracovní smlouvou, není-li v tomto zákoně stanoveno jinak. Zaměstnavatel má zajišťovat plnění svých úkolů především zaměstnanci v pracovním poměru (§ 74 odst. 1).

1.3.5 Dohody o pracích konaných mimo pracovní poměr

Mimo závislou práci na základě pracovní smlouvy mohou být mezi zaměstnancem a zaměstnavatelem uzavřeny dohody o pracích konaných mimo pracovní poměr (dohoda o provedení práce, dohoda o pracovní činnosti), § 77 zákoníku práce stanoví, že na práci konanou na základě dohod se vztahuje úprava pro výkon práce v pracovním poměru, nestanoví-li zákon jinak (dle odst. 2 tohoto paragrafu jde např. o pracovní dobu a dobu odpočinku nebo dovolenou). Podle § 74 přitom zaměstavatel není zaměstnancům na dohody povinen rozvrhnout pracovní dobu.

Dohoda o provedení práce se dle § 75 uzavírá nejvýše na 300 hodin v kalendářním roce, doba se u jednoho zaměstnavatele sčítá v případě, že je dohod uzavřeno více.

Práci na základě dohody o pracovní činnosti není možné vykonávat v rozsahu překračujícím v průměru polovinu stanovené týdenní pracovní doby za celou dobu, po niž je uzavřena, nejdéle však za 52 týdnů. Musí být sjednán rozsah pracovní doby a doba, na niž se sjednává (§ 76).

Kapitola 2

Rozvrhování směn

2.1 Problém rozvrhování směn

Problém plánování směn je jedním z rozvrhovacích problémů podobně jako sestavování jízdních řádů nebo školních rozvrhů, obecně je považovaný za velmi komplexní, a to i v případě, že se řeší jen jeho zjednodušená verze (např. se vyhodnocuje jen jedno kritérium a schopnosti zaměstnanců jsou homogenní). V rámci velkých organizací lze složitost problému snížit např. tím, že na sobě nezávislé části mají samostatný rozvrh (např. v nemocnici by mohl být rozvrh ostrahy nezávislý na rozvrhu lékařů na patologii).

Teoretických problémů, které byly popsány v literatuře a které se týkají rozvrhování pracovních sil, existuje více druhů. Liší se dle podmínek a prostředí, jehož se bezprostředně týkají, nejčastěji je popisováno rozvrhování zdravotních sester.

Řešení totoho rozvrhovacího problému lze automatizovat více způsoby, které jsou vhodné v závislosti na jeho formulaci (pro více vizte podkapitolu 2.3), v jednoduchém případě může být možné jej interpretovat tak, aby odpovídal problému již známému a řešitelnému deterministickými metodami [8], především se ale výzkum tohoto problému se zaměřuje na vývoj efektivních stochastických metod řešení [1], které mohou tuto úlohu vyřešit v kratším čase, ale jejich výsledek je pouze přibližný.

Proces rozvrhování lze rozdělit na několik částí, které na sebe mohou, ale nemusí nutně navazovat. [16] Konkrétně se jedná o:

1. předpovídání poptávky po personálu,
2. rozvrhování volných dnů,
3. rozvrhování směn,
4. rozvrhování prací,
5. rozdělení úkolů,
6. přidělení osob.

2.2 Definice problému

Rozvrhování směn obecně řeší problém, že organizace má k dispozici N zaměstnanců, které je třeba rozdělit do S směn v D pracovních dnech, a to na základě sady podmínek, které se liší dle charakteru provozu.

2.3 Klasifikace

Každý jednotlivý problém rozvrhování směn se od sebe může lišit, De Causmaecker a Berghe [12] jako příklad uvádí:

Základní charakteristiky

- Na jaké úrovni detailu se definují typy směn, schopnosti nebo pokrytí?
- Jak flexibilní jsou parametry?
- Jsou rozvrhy cyklické?

Cíle

- Je hlavním cílem optimalizovat, nebo jenom nějak rozhodnout?
- Optimalizuje se dle podmínek, počtu personálu nebo něčeho jiného?

Podmínky

- Kolik je omezujících podmínek?
- Jakého jsou omezující podmínky typu?
- Které omezující podmínky jsou zbytné a které nezbytné[1]?

Velikost problému

- Na jak dlouho se plánuje?
- Pro kolik zaměstnanců se rozvrh plánuje? Kolik je typů směn?

2.4 Podmínky

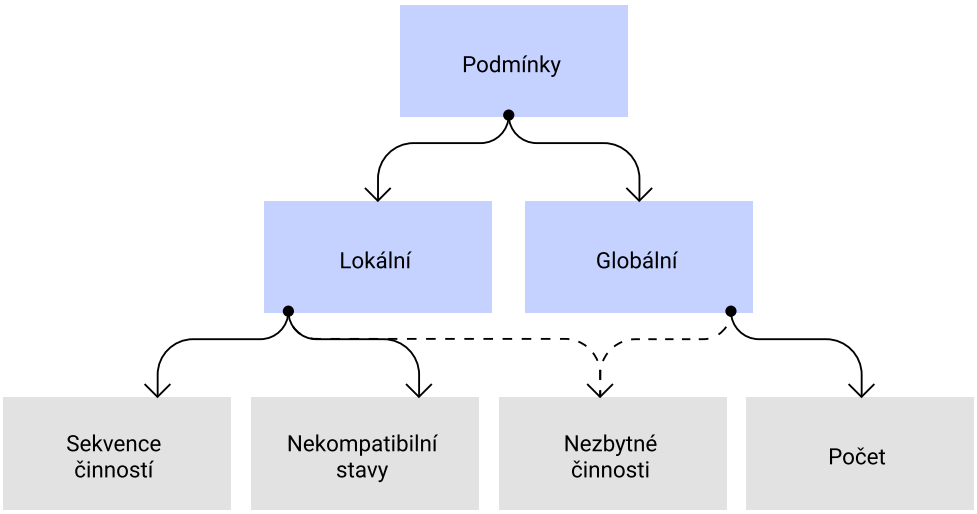
Rozvrhování směn závisí na takovém množství podmínek, že je obvykle není možné splnit všechny. Proto se také podmínky často musí před samotným řešením problému rozdělit na nezbytné (musí být splněny vždy) a zbytné (jejich splnění je žádoucí). [45] Pro každou zbytnou podmínku je vhodné stanovit její váhu. [10]

Podmínky se mohou týkat například:

- sekvence činností (např. po směně nemůže následovat 12 hodin další směna),
- počtu (např. zaměstnanec pracuje týdně 40 $\pm$ 8 h),
- nekompatibilních stavů (např. Alice nechce pracovat s Bobem),
- nezbytných činností (např. v daném dni je naplánována náročná operace srdce).

[1] K použité terminologii: anglická literatura používá pro omezující podmínky termíny *soft constraint* a *hard constraint*, lze nalézt i český překlad *měkká/tvrdá omezení*, autorka této práce však považuje termíny *zbytná/nezbytná podmínka* za lépe vyjadřující podstatu problému.

Podmínky lze rozdělit na globální, jejichž posouzení vyžaduje celkovou znalost řešení (Je na každé směně někdo?), a lokální, na jejichž verifikaci stačí pohled na vlastní podmnožinu řešení (Má tento zaměstnanec volný víkend?). Toto rozdělení je naznačeno na obr. 2.1, nezbytné činnosti však nelze rozdělit z tohoto pohledu jednoznačně (mohou se týkat jak celého rozvrhu, tak jeho části). [8]



Obrázek 2.1: Rozdělení podmínek dle jejich rozsahu

2.5 Matematická interpretace řešení

Řešení tohoto problému lze interpretovat jako trojrozměrnou binární matici $\boldsymbol{R}$, jejímiž parametry jsou zaměstnanec ($\forall i \in E$), den ($\forall j \in T$) a vypsaná směna ($\forall k \in S$), jednotlivé hodnoty R_{ijk} jsou určeny funkcí 2.1. [46] Takto má smysl řešení interpretovat spíše v případě, že se obsazují striktně určené sloty pro směny.

$$R_{ijk} = \begin{cases} 1, & \text{zaměstnanci } i \text{ je v den } j \text{ přiřazena směna } k, \\ 0, & \text{jinak.} \end{cases} \tag{2.1}$$

Den →	**1**			**2**			**3**			**4**			...
Směna → Zam. ↓	**1**	**2**	**3**	**1**	**2**	**3**	**1**	**2**	**3**	**1**	**2**	**3**	...
1	1	0	0	1	0	0	0	1	0	0	0	1	...
2	0	1	0	0	0	1	0	0	0	0	0	0	...
3	0	0	0	1	0	0	1	0	0	0	0	0	...
⋮		⋮			⋮			⋮			⋮		⋮

Tabulka 2.1: Příklad interpretace řešení dle funkce 2.1

2.6 Cílová funkce

Cílová funkce je taková funkce, jejíž hodnotu je cílem optimalizovat (minimalizovat nebo maximalizovat). V případě rozvrhování směn se může jednat o funkce vyjadřující rovnoměrnost obsazení směn, cenu lidské práce, porušené zbytné podmínky [8] či počet kontaktů mezi zaměstnanci (důležitý např. v případě epidemie) [52].

Například v případě optimalizace na základě vážených podmínek lze cílovou funkci formulovat jako rov. 2.2. Sankce za porušení nemusí být lineární, může jít i o konstantu či kvadratickou nebo libovolnou jinou funkci. [23]

$$f(\boldsymbol{x}) = \sum_{s=1}^{n} c_s \cdot g_s(\boldsymbol{x}), \tag{2.2}$$

kde $\boldsymbol{x}$ je dané řešení, n je počet zbytných podmínek, c_s je váha s-té podmínky (také lze tento údaj chápat jako sankci za její porušení) a $g_s(\boldsymbol{x})$ je počet porušení s-té podmínky v řešení $\boldsymbol{x}$. [6]

2.7 Deterministické optimalizační metody

Pro nalezení optimálního řešení mezi všemi možnostmi existuje celá řada metod – kromě vyčerpávajícího prohledání celého prostoru může jít o matematické programování (lineární, kvadratické, nelineární, ...).

2.7.1 Lineární programování

Úlohou lineárního programování je nalézt vektor $\boldsymbol{x}^* \in \mathbb{R}^n$ optimalizující hodnotu cílové funkce mezi všemi vektory, které splňují danou soustavu lineárních rovnic a nerovnic (kterým se zpravidla říká omezující podmínky nebo omezení). [31]

Celočíselné lineární programování je pro rozvrhování směn vhodné v jednoduchých případech, např. když jsou pro všechny zaměstnance stanoveny stejné počty po sobě jdoucích pracovních dnů a volna a pro každý den je navíc stanoven minimální počet personálu. Cílem je optimalizovat počet zaměstnanců tak, aby byla naplněna poptávka. [38]

2.7.2 Omezení deterministických optimalizačních metod

Tyto metody obecně můhou přinést optimální výsledky, avšak pro reálné použití je jejich model příliš jednoduchý [9], případně špatně škálovatelný.

2.8 Heuristika

Většina stochastických metod vyskytujících se v literatuře je rozšířením tzv. metaheuristik, abstraktních metod pro řešení optimalizačních problémů, upravených na problém rozvrhování směn. Narozdíl od deterministických metod řešení není zaručeno, že bude nalezeno optimální řešení, cílem je získat výstup, který je dost dobrý, a to v dostatečně krátkém čase (co konkrétně to znamená závisí na požadavcích). Neprobíhá tak vyčerpávající prohledávání všech kombinací. [20]

Metody se dají rozdělit do dvou základních skupin – jedny iterativně rozšiřují částečné řešení, dokud není kompletně dokončeno (lokální vyhledávání), druhé pracují s platným řešením, které iterativně vylepšují (vylepšovací metaheuristiky). [47]

Výměna sousedících struktur

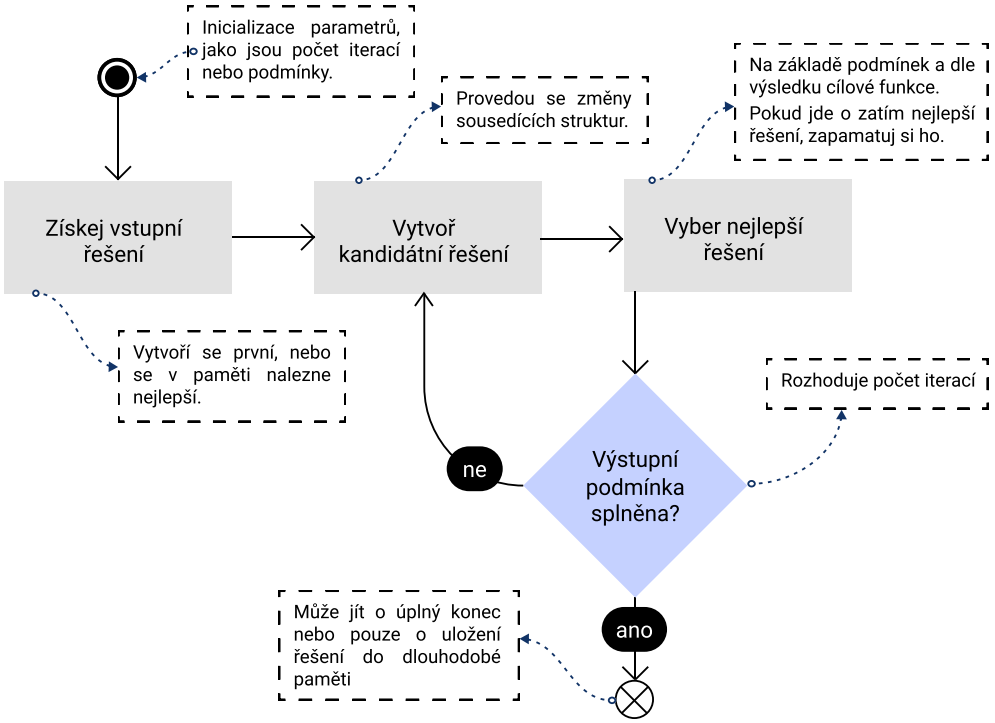
Za účelem vytvoření nebo vylepšení rozvrhu je obecně možné provádět řadu změn, dle Kletzandera a Musliu [23] jde například o:

- přidání či odebrání směny,
- výměnu směny za jinou,
- změnu typu směny,
- změnu či vytvoření posloupnosti směn,
- výměnu směn mezi zaměstnanci,
- zkrácení směny.

2.8.1 Tabu search

Tabu search je metaheuristika, jejímž principem je prohledávání okolí s ohledem na již nalezená řešení, která jsou zapovězená (tabu). Cílem je naopak prohledávat i nová řešení, která jsou horší, aby nedocházelo pouze k nalezení lokálního minima cílové funkce bez dalšího průzkumu. Základem je existence dvou druhů paměti – krátkodobé (obsahuje naposledy navštívená řešení) a dlouhodobé (obsahuje frekvenci návštěv). [25]

Vytváření a vylepšování řešení probíhá způsobem dle obr. 2.2 tak, aby byly splněny podmínky. Prohledávání prostoru probíhá v blízkém sousedství řešení, které je omezeno tak, aby se řešení nezacyklilo (jsou určeny zapovězené kroky – tabu – v krátkodobé paměti, aby nebyla procházena stále stejná, dále již cílovou funkci nezlepšující řešení – tabu lze použít pouze tehdy, když řešení splňuje aspirační kritéria, tedy je lepší). [19]



Obrázek 2.2: Metaheuristika tabu search

Algoritmus začíná inicializací řešení částečně náhodným způsobem tak, aby řešení neporušovalo nezbytné podmínky. Vylepšování referenčního řešení probíhá pomocí výměny

sousedících struktur (dle podmínek např. výměna posloupnosti směn mezi dvěma zaměstnanci). [36]

2.8.2 Genetický algoritmus

Genetický algoritmus je metaheuristika inspirovaná genetikou – používají se operace zvané selekce chromozomů (výběr těch, které budou v další generaci), křížení (náhodná kombinace částí dvou řešení) a mutace (náhodná změna).

Narozdíl od tabu search se v genetickém algoritmu prohledává větší množina řešení – začíná se s počáteční populací řešení, v níž se selektují nejlepší řešení, která se dále mezi sebou kříží (kombinace náhodných částí), případně nahodile mutují. Algoritmus končí, pokud řešení konverguje – už se dlouhou dobu nepodařilo ho vylepšit. [29]

Tento přístup lze aplikovat na rozvrhování směn například tak, že se jako chromozom kóduje kombinace dne, směny a zaměstnance. [28]

Kapitola 3

Analýza systému

3.1 Cílová skupina

Aby bylo možné stanovit požadavky na aplikaci, byl vytvořen předpoklad o cílové skupině pro tento systém. Jedná se o menší firmu (případně malý tým v rámci větší firmy) o 10 až 20 zaměstnancích, která podniká v nekritickém vícesměnném provozu (např. jedna pobočka řetězce rychlého občerstvení, restaurace nebo obchod). Firma zaměstnává jak pracovníky s pracovní smlouvou, tak brigádníky na DPP/DPČ. Vzhledem k tomu řeší problém, jak rozvrhovat směny, stávající způsob je z pohledu vedení firmy i samotných zaměstnanců neefektivní například z následujících důvodů:

- brigádníci si musí směny domlouvat osobně,
- rozvrh je pověšen pouze na nástěnce,
- rozvrh se sestavuje manuálně,
- nedodržuje se týdenní pracovní doba,
- rozvrh je sestaven dle osobních preferencí vedoucího.

3.2 Požadavky z pohledu organizace

Následující požadavky definují, jaké základní funkce by měl tento systém mít, a to především z pohledu organizace, která by jej využívala (jedná se o požadavky s vysokou mírou abstrakce). Vychází se z představy popsané výše, tedy že tento software je určen pro nasazení ve firmě, která momentálně není schopna efektivně připravovat rozvrhy směn a chtěla by ušetřit čas a zdroje, které na sestavení rozvrhu vynakládá.

B1. Systém umožní distribuci rozvrhů směn k zaměstnancům.

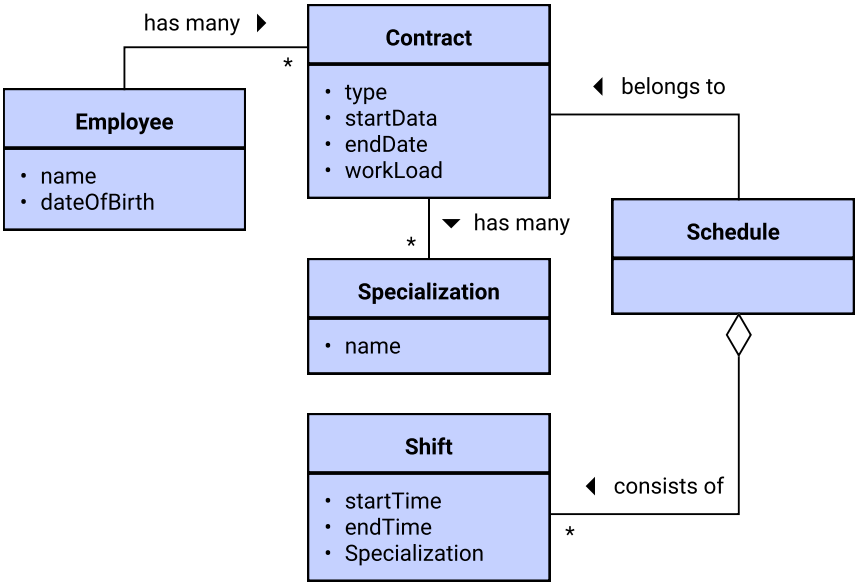
B2. Systém umožní zápis zaměstnance na DPP/DPČ na směnu.

B3. Systém vytvoří rozvrh směn na základě požadavků organizace.

B4. Systém bude automatizovat dohled nad dodržováním legislativy.

3.3 Doménový model

V diagramu 3.1 je definován předpokládaný doménový model firem cílové skupiny, jde o znázornění skutečnosti, že ve firmě pracují zaměstnanci, kteří mají smlouvy a kterým jsou do jejich rozvrhu přiřazeny směny. Bližší popis doménového modelu je součástí přílohy A.



Obrázek 3.1: Doménový model

3.4 Analýza existujících řešení

Tato podkapitola se věnuje popisu již existujících nástrojů pro plánování lidských zdrojů. Jedná se o komerční software, který byl otestován v demoverzi, vzhledem k tématu této práce byly hlavními sledovanými aspekty mobilní aplikace pro zaměstnance a způsob rozvrhování směn.

Tyto systémy lze zařadit do kategorie tzv. informačních systémů pro řízení lidských zdrojů, jejichž úkolem je obecně jak automatizovat, tak podporovat strategické rozhodování na základě dat. [24]

Byly vybrány informační systémy pro plánování lidských zdrojů, které alespoň částečně odpovídají uvedeným požadavkům z pohledu organizací – podporují samoobslužné sestavování rozvrhů a případně i tvorbu rozvrhů nějakým způsobem automatizují.

3.4.1 Tamigo

Aplikace Tamigo[1] je aplikace pro plánování lidských zdrojů, je určena pro použití v pohostinství, maloobchodě, zdravotnictví aj. [43]

Aplikace je organizacím nabízena v několika variantách, které jsou zpoplatněny podle množství podporovaných funkcí (jedná se např. o rozpisy směn všech zaměstnanců, výkazy práce, správu absencí, správu mezd a smluv zaměstnanců, evidenci příchodů a odchodů) a počtu zapojených zaměstnanců. Mezi součásti tohoto produktu patří webové rozhraní i

[1] `https://www.tamigo.cz/`

mobilní aplikace (přestože jde o nativní aplikaci, v testované verzi fungovala pouze v případě, že je uživatel připojen k síti, jinak skončila chybou). Uživatelské role jsou v tomto systému administrátor, manažer a zaměstnanec.

Systém umožňuje manažerům vytvářet rozvrh, částečně je tento proces manuální, částečně lze rozvrh vygenerovat na základě šablon, které si manažer předem připraví (např. obvyklá pracovní doba pro jednotlivé zaměstnance).

Zaměstnanec si může aktualizovat osobní data, zobrazit osobní rozvrh směn, žádat o dovolenou a sledovat stav jejího čerpání, zobrazit výkaz práce pro výplatní období, aj.

3.4.2 Tanda

Na podobném principu jako aplikace Tamigo pracuje i aplikace Tanda[2]. Automatické rozvrhování funguje na principu šablon – rozvrh se tedy vytvoří jednou a následně se opětovně používá [44].

I tento systém má jak webové, tak mobilní rozhraní, obojí pro zaměstnance i vedoucí. Mobilní rozhraní opět nepodporuje ani čtení dat v případě, že je uživatel offline.

3.4.3 When I Work

Aplikace When I Work[3] má opět webové i mobilní rozhraní. Rozvrhování zde opět funguje manuálně nebo na základě šablon, aplikace v placené verzi [50] však podporuje i automatické přiřazení volných směn k vhodným zaměstnancům (v potaz při rozvrhování bere pracovní pozici, již existující směny, požadavky na volno a další filtry). [49] Mobilní aplikace umožňuje zaměstnancům zadat, jakou pracovní dobu preferují nebo kdy jsou naopak nedostupní. Ani tato aplikace nefunguje v případě, že uživatel není připojen k síti.

3.4.4 Shrnutí

Vyzkoušené informační systémy se v principu příliš neliší a nabízejí podobné základní prvky (mobilní a webové rozhraní; uživatelské role; komplexní správa mezd, docházky, rozvrhů). Společné mají i to, že žádná z aplikací neimplementuje ani částečný offline režim, což může být pro uživatele v některých situacích nepříjemné (lze ale předpokládat, že mobilní aplikace nebyla tvůrci považována za prioritní v porovnání s webovým rozhraním). Liší se však způsobem, jakým rozvrhují směny – největší automatizaci zde poskytuje aplikace When I Work. Dalším rozdílem je uživatelská přívětivost, její hodnocení by však bylo spíše subjektivní.

Alternativou systémů, které přímo odpovídají požadavkům, mohou být komplexnější systémy ERP a jejich modul pro plánování lidských zdrojů.

3.5 Požadavky na systém

Na základě požadavků ze strany organizace a po otestování podobných aplikací byly stanoveny požadavky na funkce nového systému, které byly rozděleny do několika kategorií podle cílového uživatele. Při implementaci systému na plánování směn je žádoucí, aby byl dohled nad dodržováním pracovněprávních předpisů (především těch, které uvádějí kvantitativní údaje), automatizován, proto jsou rovněž uvedeny požadavky vyplývající z analýzy legislativy (podkapitola 1.3).

[2] `https://www.tanda.co/`

[3] `https://wheniwork.com`

Uživatelé

U1. Systém bude personalizovaný podle potřeb daného uživatele.

Zaměstnanci

Z1. Systém umožní zaměstnancům náhled do jejich rozvrhu.

Z2. Systém umožní zaměstnancům na DPP/DPČ zápis na směnu.

Organizace

O1. Systém umožní registraci nových organizací.

O2. Systém umožní přidávání nových zaměstnanců.

O3. Systém umožní automatické přiřazení směn.

O4. Systém umožní plánování směn na následující týdny.

O5. Systém umožní náhled na rozvrh zaměstnanců.

Legislativa

L1. Systém umožní náhled do osobního rozvrhu nejméně 2 týdny předem.

L2. Systém umožní zaměstnancům na DPP odpracovat nejvýše 300 hodin v kalendářním roce.

L3. Systém umožní zaměstnancům na DPČ odpracovat nejvýše 20 hodin týdně v průměru 52 týdnů.

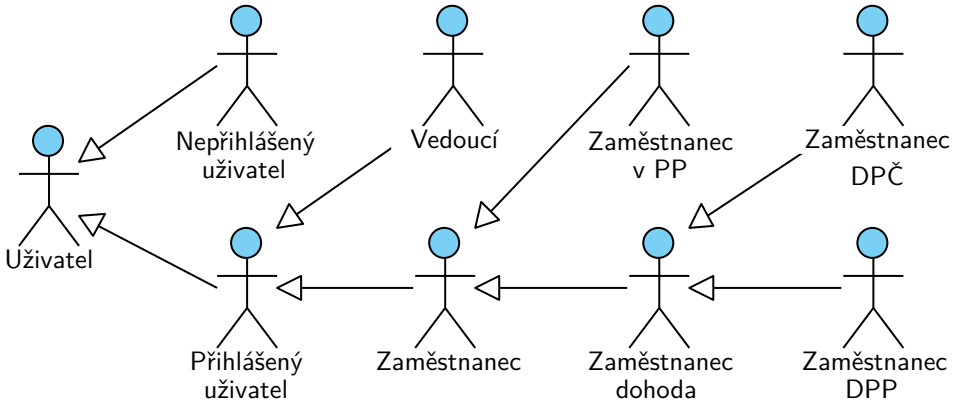
L4. Systém rozvrhne směny s ohledem na 40hodinovou týdenní pracovní dobu a úvazky jednotlivých zaměstnanců.

3.6 Aktéři, uživatelské role

Jedním z hlavních cílů systému na plánování směn je informovat každého zaměstnance o jeho směnách, proto se jeví vhodnou personalizace, které se docílí s pomocí uživatelských účtů. Vstupní operací bude přihlášení – z toho vyplývá nezbytnost existence role přihlášeného a nepřihlášeného uživatele.

Roli přihlášeného uživatele je nutné dále rozšířit, konkrétně na zaměstnance a vedoucího, který má zodpovědnost za rozvrhování pracovní doby a měl by mít komplexní přehled o rozpisu i možnost do něj zasáhnout.

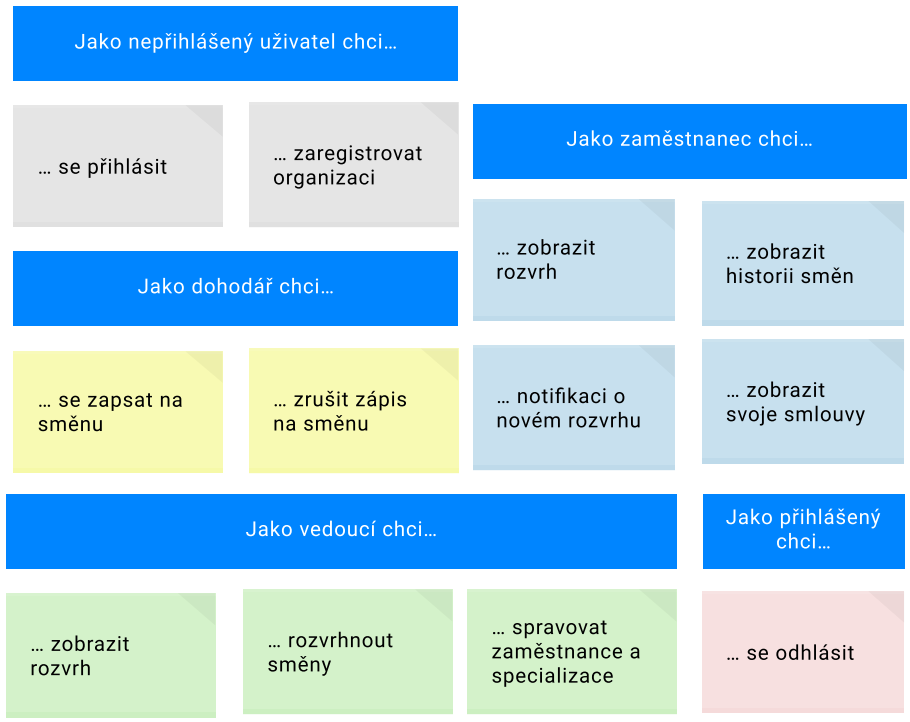
Jednotliví zaměstnanci pak v systému mohou být různými aktéry, především na základě pracovněprávního vztahu (tj. zaměstnanec v pracovním poměru, zaměstnanec na DPP a zaměstnanec na DPČ), vizte obr. 3.2. Důvodem je, že aktéři budou moci provádět specifické operace a systém s nimi bude interagovat odlišně. Rozhodujícím faktorem není to, zda má zaměstnanec plný nebo zkrácený úvazek, v obou případech platí stejné podmínky a liší se jen týdenní pracovní doba, stejně tak nezáleží na tom, zda je zaměstnanec nezletilý.



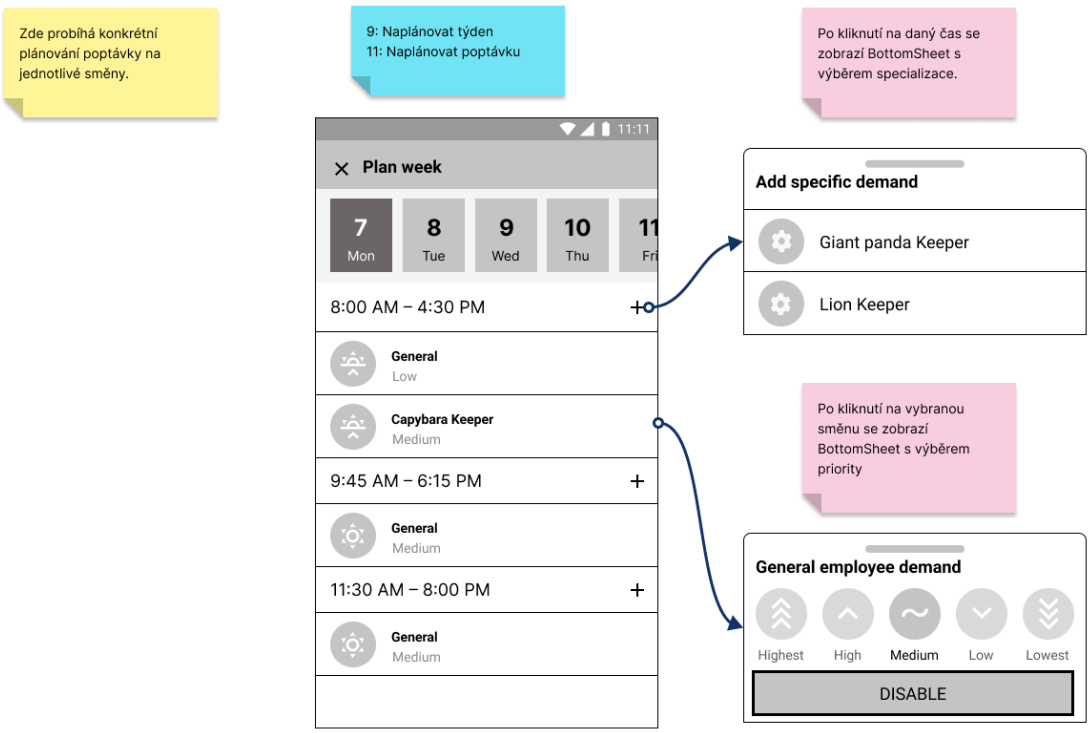
Obrázek 3.2: Diagram aktérů

3.7 Uživatelské příběhy

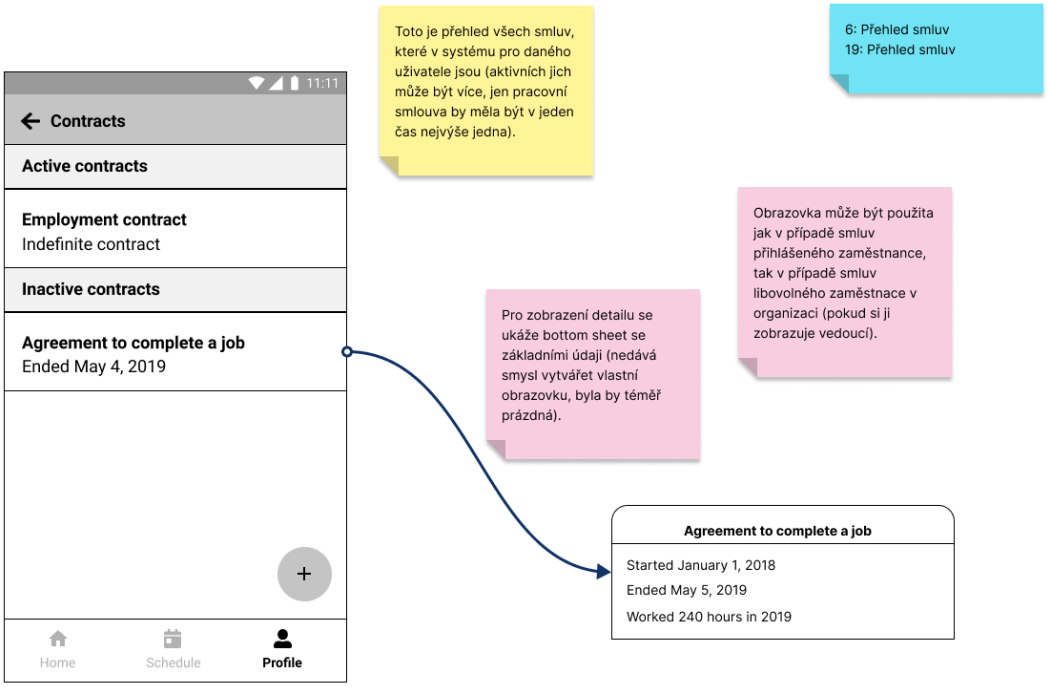
Pro analýzu systému z pohledu uživatelské interakce byl zvolen neformální způsob inspirovaný uživatelskými příběhy (user stories), hlavní výhodou je využití běžného jazyka. Přehled uživatelských příběhu ve zjednodušené podobě je na obr. 3.3, jejich podrobnější mapa je součástí přílohy A. Součástí přílohy A jsou i nákresy uživatelského rozhraní včetně popisu, pro ukázku vizte obr. 3.4



Obrázek 3.3: Zjednodušený přehled uživatelských příběhů



(a) Obrazovka pro úpravu poptávky



(b) Obrazovka se seznamem smluv

Obrázek 3.4: Ukázka nákresů UI

Část II

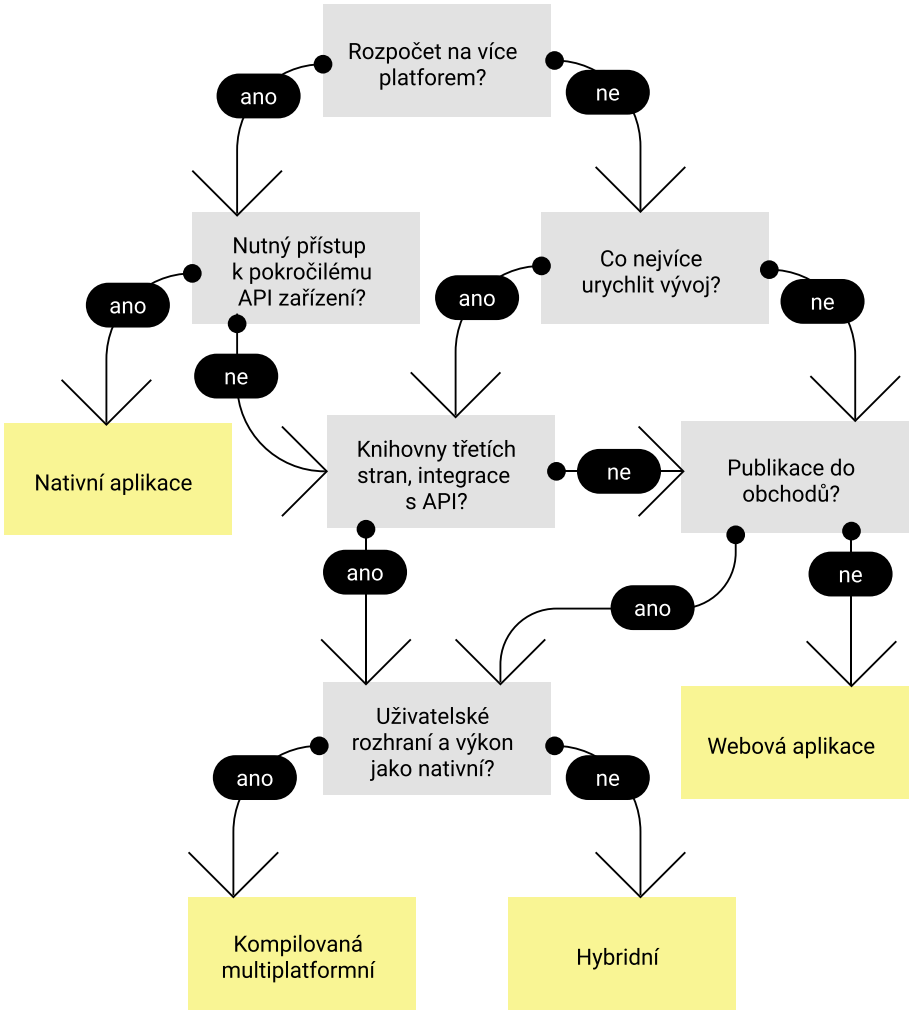
Praktická část, implementace

Kapitola 4

Výběr technologií

4.1 Technologie pro uživatelské rozhraní

V případě, že by byla aplikace určena k nasazení v produkčním prostředí, připadaly by v úvahu nejméně dva přístupy – mobilní nebo webová aplikace.



Obrázek 4.1: Rozhodovací strom pro vývoj na mobilní zařízení

4.1.1 Webová aplikace

Pro použití webové aplikace hovoří fakt, že je není třeba žádným způsobem instalovat a stačí prosté zadání adresy do internetového prohlížeče, z podstaty věci je multiplatformní, lze ji zobrazit jak na počítači, tak na mobilním zařízení, jedná-li se o responzivní aplikaci, může být i její použití na mobilním zařízení uživatelsky přívětivé. Nevýhodami jsou omezené možnosti při implementace offline režimu, i to, že z hlediska rychlosti se nativním mobilním aplikacím nevyrovnají. [41]

4.1.2 Mobilní aplikace

Vyplývá-li podpora mobilní platformy z požadavků, ilustruje další rozhodování o způsobu implementace strom na obrázku 4.1 (vychází z [32]), především ale záleží na prioritách toho, kdo danou aplikaci chce uvádět na trh.

Oproti responzivnímu webu mají nativní aplikace výhodu, že mohou jednoduše přistupovat k API, které poskytuje zařízení, na kterém jsou nainstalované, tedy mohou snadno získat přístup k fotoaparátu či poloze zařízení; také lze na zařízení jednoduše přijímat notifikace, i když je aplikace na pozadí. Nativní aplikace mohou být vhodné i pro použití v offline režimu, aplikace může mít lokální databázi.

Nevýhodou pro uživatele je, že se musí instalovat na zařízení. S tím souvisí i o něco složitější způsob distribuce – v případě, že je možné zveřejnit aplikaci v obchodech (Google Play, AppStore), musí projít schvalovacím procesem, což může trvat několik dní a nemusí být vždy úspěšné, např. v případě, že aplikace obsahuje nevhodný obsah, narušuje něčí autorská práva nebo nevhodným způsobem zpracovává osobní údaje uživatelů. Je třeba uvádět, jaká oprávnění pro přístup ke mobilním API (SMS, kontakty, poloha, soubory, aj.) aplikace požaduje. [21].

Při výběru konkrétního způsobu implementace mobilní aplikace pak existuje několik dalších rozhodovacích situací, především výběr podporované platformy.

Platforma

Na trhu s chytrými mobilními zařízeními jsou dvě dominantní platformy, Android (podíl na trhu celosvětově 72 %) a iOS (podíl na trhu 27,5 %) [40]. Nevýhodou rozšířenějšího Androidu oproti iOS je fragmentace, tzn. existence velké řady různých zařízení, která je třeba podporovat. Pro účely této práce bude podporována pouze platforma Android.

Způsob vývoje

Je-li nezbytné podporovat více platforem, je možné vyvinout aplikaci multiplatformně, hybridně, případně pro každou platformu zvlášť.

Multiplatformní vývoj Multiplatformní vývoj umožňuje nasazení jedné aplikace na více platformách, kód pro všechny platformy je sdílený, jádro aplikace tedy není třeba duplikovat, což je jejich nespornou výhodou, navíc to může i snížit náklady na vývoj.

Problémem je přístup k aplikačnímu rozhraní a hardwaru daného zařízení, omezené možnosti při používání knihoven specifických pro platformu nebo složitější vývoj uživatelského rozhraní tak, aby odpovídalo konvencím pro všechny platformy. [30] Používají se frameworky jako Flutter (jazyk Dart), React Native (JavaScript) nebo Xamarin (C#).

Vývoj pro danou platformu V případě vývoje na jednu platformu je možné používat všechny součásti zařízení naplno. Co se přístupu k hardwaru, výkonu a uživatelské přívětivosti týče, mají nativní aplikace nad multiplatformními jasnou převahu, jejich vývoj je však dražší. [13]

Hybridní vývoj Hybridní aplikace jsou kombinací webu a nativní mobilní aplikace – použijí se standardní technologie pro vývoj webu (HTML + CSS + JavaScript), toto se obalí do nativní aplikace, v níž běží webový prohlížeč. Jejich největší výhodou je cena, rychlost vývoje a jednodušší správa. Stinnou stránkou tohoto přístupu je menší uživatelská přívětivost, omezený přístup k API zařízení nebo nižší výkon v porovnání s plně nativní aplikací. [14]

Programovací jazyk

Pro nativní vývoj na platformu Android lze vybrat nejméně ze tří možností, a to C/C++, Javy a Kotlinu. Toto rozhodování může v této době u nových projektů probíhat spíše pouze v teoretické rovině, neboť oficiální podporu ze strany Googlu má nejmodernější jazyk Kotlin.

Mezi hlavní výhody Kotlinu oproti jiným variantám patří null-safety (bezpečné volání na objektech, které mohou nabývat hodnotu `null`), coroutines (odlehčená verze vláken pro asynchronní operace) nebo Kotlin Extensions (knihovny nejrůznějších funkcí usnadňující vývoj obecně i specificky pro Android). [4]

Obdobné rozhodování může probíhat i v případě platformy iOS, pro kterou se nabízí použití Swiftu nebo starého jazyka Objective-C. Ze strany Applu je doporučena modernější varianta (Swift), protože aplikace jsou rychlejší a vývoj snazší.

Architektonický vzor

Standardním architektonickým vzorem používaným při vývoji Android aplikací je MVVM. [2]. Aplikace má tři vrstvy, které spolu komunikují způsobem naznačeným na obr. 4.2. Hlavním cílem je oddělit prezentační vrstvu od business logiky. [39]

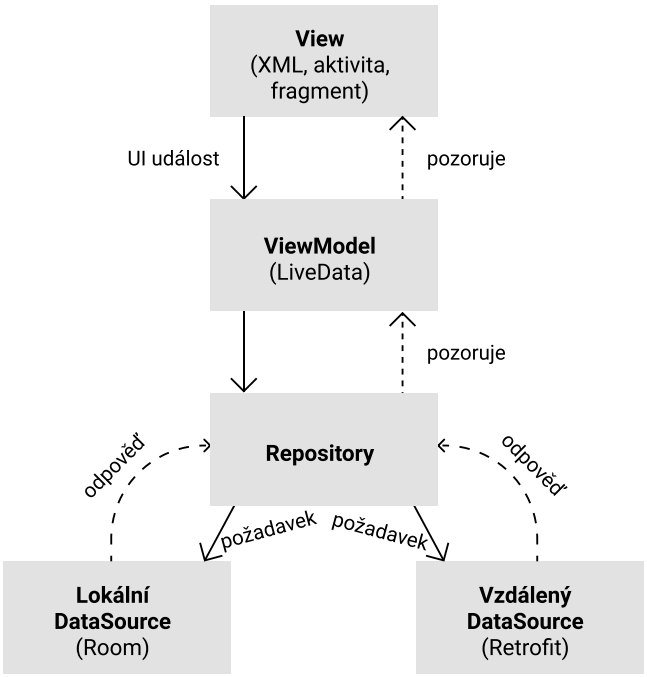
MVVM vychází ze staršího architektonického vzoru MVP, Presenter je podobný jako ViewModel, hlavním rozdílem je, že Presenter si drží referenci na View (relace mezi View a Presenterem je 1:1) a řídí, kdy se má View aktualizovat, kdežto ViewModel neví, jaký View ho pozoruje a relace mezi View a ViewModelem je 1:n. [48] Nadstavbou nad MVVM je tzv. Clean Architecture, která zaručuje ještě větší oddělení prezentační vrstvy a business logiky (Android aplikace se rozděluje do více modulů, nejčastěji nazvaných Presentation, Domain a Data). [22]

Vzhledem k předpokládané velikosti projektu byl zvolen doporučený architektonický vzor MVVM.

4.1.3 Shrnutí

Jako nejvhodnější by se v případě produkčního nasazení této aplikace jevilo zkombinovat dva přístupy, tedy mít uživatelské rozhraní jak ve formě webové aplikaci, tak mít i podpůrnou mobilní aplikaci. Tento přístup byl také zvolen v případě většiny podobných řešení (podkapitola 3.4). Zde bude implementována pouze jedna část, a to mobilní aplikace, neboť důraz bude kladen především na pohodlí zaměstnanců, kteří chtějí mít svůj rozvrh po ruce co nejrychleji.

Mobilní aplikace bude vyvinuta jako nativní Android aplikace, v produkčním prostředí by bylo žádoucí uživatele iOS nediskriminovat, a aplikaci jim také nabídnout, z tohoto



Obrázek 4.2: Komunikace mezi vrstvami v MVVM

důvodu by bylo vhodné zvážit použití nějakého multiplatformního frameworku, rozhodující by byly plány na budoucí vývoj, případně i finance. Pro vývoj se použije programovací jazyk Kotlin, protože se jedná o moderní programovací jazyk, který je pro vývoj na danou platformu nejvhodnější. Aplikace bude navržena dle zásad architektonického vzoru Model-View-ViewModel.

4.2 Backendová technologie

4.2.1 Webový framework

Webové frameworky, tj. knihovny zdrojových kódů připravené pro tvorbu webů (především jejich serverové strany), se standardně používají pro zjednodušení vývoje a nasazení aplikací. Ať už jsou napsány v libovolném programovacím jazyce (může jít např. o PHP, Javu, Ruby nebo Python), standardními součástmi jsou perzistence dat, autentizace uživatelů, správa session nebo šablony pro uživatelské rozhraní. Společnou mají také podporu pro architektonický vzor MVC. [15]

Z množství variant, které se nabízejí, byl zvolen framework Ruby on Rails (v jazyce Ruby), a to především z důvodu jeho filosofie Don't Repeat Yourself (kód by měl být znovupoužitelný, bez zbytečného opakování) a Convention Over Configuration (webová aplikace je nakonfigurována dle konvencí, a vývojář tak nemusí trávit čas psaním konfiguračních souborů) [35], která slibuje výrazné usnadnění vývoje.

4.2.2 Aplikační rozhraní

Pro komunikaci s dalšími aplikacemi (zde především pro komunikaci klienta se serverem) musí aplikace vnějšku poskytnout své aplikační rozhraní. Za tímto účelem existuje několik možností, které jsou vhodné pro různé případy (např. REST, SOAP, GraphQL, gRPC).

Tyto způsoby implementace se mohou od sebe odlišovat formátem přenosu (binární, textový) nebo protokolem (HTTP i jiné).

Pro účely tohoto projektu se jako optimální varianta jeví použití REST API, a to vzhledem k velmi dobré podpoře ze strany webového frameworku bez nutnosti rozšiřování o další knihovny.

V některých případech by bylo vhodnější použít pro komunikaci jiný prostředek (především v případě volání výpočtů na serveru), např. RPC (gRPC, XML-RPC, JSON-RPC či jinou konkrétní implementaci), neboť zde nedochází k přenosu zdrojů, ale pouze k výpočtům na vzdáleném serveru (provolávání akcí). [42] Toto by ale vytvořilo problémy při nasazení aplikace na PaaS službách (např. Heroku) [27], proto bylo zvoleno řešení, které sice není z hlediska architektury tou nejlepší volbou, ale jeho použití je flexibilnější.

4.2.3 Databáze

Rozhodování ohledně použité databáze se může týkat způsobu uložení dat (relační nebo NoSQL) a samotného DBMS. Je třeba se rozhodnout, zda v projektu bude použita relační či NoSQL databáze. Relační databáze přitom jsou vhodné na ukládání dat se statickou strukturou, NoSQL databáze mohou být flexibilnější, co se ukládání dat týče. [18]

Pro účely tohoto projektu byla zvolena relační databáze, a to z důvodu předem určené struktury dat a taktéž proto, že jde o technologii, která má dobrou podporu, a to i ze strany webových frameworků. Upřednostněna tedy byla tradiční technologie. Jako konkrétní DBMS byl zvolen Postgres.

4.2.4 Shrnutí

Aplikace bude vyvinuta ve webovém frameworku Ruby on Rails, a to vzhledem k tomu, že jde o rozšířený framework, který slibuje jednoduchou konfiguraci. Z volby frameworku vyplývá použit architektonického vzoru MVC. S vnějškem bude tato aplikace komunikovat prostřednictvím REST API vzhledem k jednoduché implementaci a silné podpoře pro jeho využití. Použita bude relační databáze, konkrétně Postgres.

4.3 Požadavky na kvalitu software

Z rozhodování v této kapitole vyplynuly následující požadavky na kvalitu software:

S1. Bude vyvinuta nativní Android aplikace v jazyce Kotlin.

S2. Backend bude vyvinut ve webovém frameworku Ruby on Rails.

S3. Mobilní aplikace bude se serverem komunikovat přes jeho REST API.

S4. Bude použita databáze Postgres.

Kapitola 5

Návrh vlastního algoritmu

5.1 Požadavky na rozvrh

Z hlediska klasifikace uvedené v podkapitole 2.3 půjde o neperiodický rozvrh s flexibilními parametry, cílem bude spíše rozhodnout než řešení co nejvíce optimalizovat.

Nezbytné podmínky budou dány legislativními požadavky na rozvrh směn, rozvrh bude sestaven na týden, cílem je netvořit zbytečné překážky v zaměstnání a rozvrhnout práci všem zaměstnancům, kterým nebudou směny rozvrženy jiným způsobem (manuálně vedoucím nebo samostatně zaměstnancem). Zbytné podmínky byly vybrány dle možných manažerských požadavků na rozvrh směn.

Konkrétní požadavky byly stanoveny následovně:

1. Algoritmus rozvrhne směny na jeden týden.
2. Algoritmus rozvrhne směny v celém dni na základě poptávky.
3. Algoritmus rozdělí směny mezi zaměstnance v pracovním poměru.
4. Algoritmus vezme v potaz legislativní požadavky (přestávka, úvazek, maximální délka směny).

5.2 Tvorba směn a poptávky

Při rozvrhování se pro zjednodušení předpokládá, že v každém týdnu budou vypsány směny pravidelným způsobem, to znamená, že všechny budou stejně dlouhé a budou každý den v týdnu začínat ve stejnou dobu, a budou v rámci jednoho pracovního dne rozloženy pravidelně dle počtu směn (pro realističtější modelování situace existuje možnost některé směny z rozvrhu vyjmout). Půjde tak flexibilně stanovit parametry rozvrhování, konkrétně celkovou pracovní dobu, délku jedné směny, délku přestávky a počet směn v jednom dni. Bude také možné naplánovat směny probíhající přes noc, případně i 24hodinový provoz (pak bude třeba stanovit začátek jedné ze směn, aby bylo možné jejich rovnoměrné rozdělení). Pro každou jednotlivou směnu lze dále určit její prioritu (číslo mezi 0 a 5 určující, zda má směna být rozvržena více nebo méně zaměstnancům vzhledem k celku).

5.3 Podmínky

Obdobně jako v podkapitole 2.4 se podmínky dají rozdělit do dvou skupin (nezbytné a zbytné) tak, že při porušení nezbytných podmínek vůbec nebude rozvrh možné uložit

do databáze (lze určovat, zda je řešení validní či nikoli), zbytné podmínky budou pouze „doplňkem“, který říká, jak moc je dané řešení kvalitní z hlediska požadavků, které si klade manažer.

5.3.1 Nezbytné podmínky

Pro určení nezbytných podmínek byly využity poznatky z oblasti legislativy (podkapitola 1.3), především maximální délka směny, minimální přestávka mezi koncem jedné směny a začátkem další, týdenní pracovní doba, dále také určení specializace.

Rozvhování probíhá v transakci a tyto podmínky jsou vyhodnocovány na úrovni modelů, v případě, že budou porušeny, bude vyvolána výjimka, dojde k rollbacku a zahození celého rozvrhu.

Konkrétně se jedná o následující podmínky:

1. Zaměstnanec v celém týdnu odpracuje nejvýše počet hodin úměrný svému úvazku.
2. Mezi koncem jedné směny a začátkem další bude nejméně 12hodinová přestávka.
3. Zaměstnanec může být přiřazen pouze na směnu, která nemá žádnou specializaci nebo má stejnou specializaci jako zaměstnanec.
4. Zaměstnanec může být v jednu chvíli pouze na jedné směně, směny se nesmějí překrývat.

5.3.2 Zbytné podmínky a cílová funkce

Byly zvoleny čtyři zbytné podmínky (obsazení všech směn, obsazení směn úměrně poptávce, obsazení specializovaných směn, více volna v kuse), podle kterých bude možné určovat, jak moc je dané validní řešení kvalitní. Pro každou z nich byl zvolen způsob, jak se bude počítat cílová funkce, dále byl stanoven pokud možno co nejjednodušší způsob, jak optimalizovat hodnotu cílové funkce. Váhy jednotlivých kritérií se určují při volání algoritmu, pokud je váha nulová, daná strategie pro vylepšování rozvrhu neproběhne.

Cílová funkce je stanovena obdobně jako v podkapitole 2.6 – jde o součet vážených porušení zbytných podmínek (sankce za porušení je lineární).

5.3.3 Obsazení všech směn

Smyslem této podmínky je určit, zda v každou chvíli bude na pracovišti alespoň jeden zaměstnanec. Vyhodnocení probíhá tak, že se naleznou všechny směny, které nemá do svého rozvrhu zapsán žádný zaměstnanec. Výsledkem cílové funkce je počet neobsazených směn násobený sankcí.

5.3.4 Obsazení směn úměrně poptávce

Jak již bylo uvedeno v podkapitole 5.2, každá ze směn může mít určenou svou prioritu v závislosti na tom, jak moc velké množství zaměstnanců je v daném čase na pracovišti potřeba. Penalizována je odchylka od požadovaného obsazení `D[i]` pro `i` = priorita dané směny v souhrnném rozvrhu (jde tak o globální podmínku).

Rozložení poptávky

Pro účely rozložení poptávky mezi směny s různou prioritou byl navržen následující postup: nejprve se spočítá průměrná priorita (`avg_priority`) ze všech směn v daném období

a průměrný počet zaměstnanců na jednu směnu (`avg_assignments`, počet přiřazení na směnu dělený počtem směn).

Z těchto údajů se vypočítá požadovaný počet zaměstnanců na směně o střední prioritě (`medium_demand = 3`). `const` je nějaká konstanta, která ovlivňuje rozložení poptávky (vizte tab. 5.1). Hodnota konstanty by měla být alespoň `medium_demand`, aby poptávka byla kladné číslo. S výsledkem rov. 5.1 a 5.2 se pracuje v zaokrouhlené formě (může tak docházet k chybě).

$$\texttt{D[3]} = \texttt{avg\_assignments} \cdot \frac{\texttt{1 + (3 - avg\_priority)}}{\texttt{const}} \tag{5.1}$$

$$\texttt{D[i]} = \texttt{D[3]} \cdot \frac{\texttt{1 - (3 - i)}}{\texttt{const}} \tag{5.2}$$

Tabulka 5.1: Rozložení poptávky

zam.	const	D[1]	D[2]	D[3]	D[4]	D[5]
10	2	0	1	2	3	4
10	7	1	2	2	2	3
10	10	2	2	2	2	2
30	2	0	4	7	11	14
30	7	5	6	7	8	9
30	10	6	6	7	8	8

5.3.5 Více volna v kuse

Tato lokální podmínka vychází z předpokladu, že je pro zaměstnance příjemnější mít více dní volna najednou. Je penalizováno, pokud nemá zaměstnanec alespoň 48 hodin volna v kuse (mezi směnami nebo vzhledem k začátku/konci týdne), pokud tedy jsou v rámci rozvržení pracovních dnů v daném týdnu volné víkendy (či jiné dva po sobě jdoucí dny), bude penalizace nulová u všech zaměstnanců.

5.3.6 Obsazení specializovaných směn

Penalizováno je obsazení směn, které nejsou specializované. Výsledek cílové funkce dává součin celkového počtu přiřazení na směny bez specializace a sankce.

5.4 Popis algoritmu

Algoritmus (vizte kód 5.1) začíná vytvořením nějakého platného řešení (jde o náhodné řešení při splnění nezbytných podmínek), o kterém lze říci, jak moc kvalitní je a které lze srovnávat s ostatními řešeními, co se hodnoty cílové funkce týče. Toto řešení slouží jako referenční pro další operace (pokud se řešení v dalších krocích podaří zlepšit, změní se referenční řešení na nové, lepší řešení).

```
Vytvoř první řešení
POKUD zbývají strategie
  Zvol další strategii
  Pokus se vylepšit řešení
  POKUD je nové řešení lepší
```

```
    Ulož nové řešení
JINAK
  Vrať nejlepší řešení
```

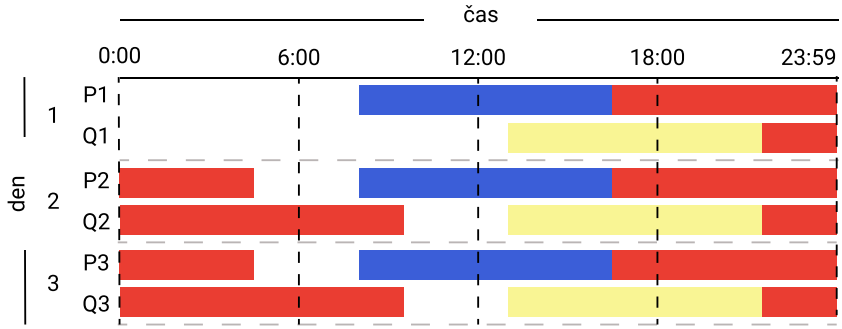
Výpis kódu 5.1: Pseudokód rozvrhovacího algoritmu

Referenční řešení se se znalostí informací relevantních pro danou zbytnou podmínku algoritmus dále pokouší zlepšovat (postupně z hlediska různých podmínek).

Tento algoritmus se inspiruje u výše zmíněných řešení nalezených v literatuře především v podmínkách a v zavedení náhodného referenčního řešení, které se postupně zlepšuje různými operacemi se sousedními strukturami, rovněž existencí vah a cílové funkce. Liší se především tím, že v jednu chvíli je pro zjednodušení vylepšována a vyhodnocována právě jedna podmínka dle vlastní strategie, a to jen v těch částech rozvrhu, kde dochází k jejímu výraznému porušování (pokud tedy některé části rozvrhu danou podmínku nezhoršují, nejsou pro danou strategii relevantní). Po konci strategie se vyhodnocuje celkové řešení s ohledem na váhy všech podmínek.

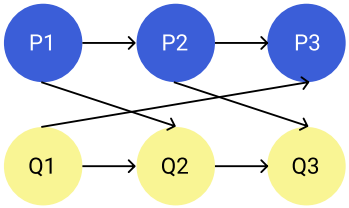
5.4.1 Nepřekrývání směn, přestávka

Ze zjevných důvodů nelze rozvrh vytvořit tak, aby byl jeden zaměstnanec přiřazen na více směn v jednu chvíli. Z legislativních požadavků navíc vyplývá nutnost mít mezi koncem jedné směny a začátkem další alespoň 11 hodin nepřetržitého odpočinku (u nezletilých zaměstnanců jde o 12 hodin), tímto tedy je rozšířena podmínka nepřekrývání směn. Pro jednoduchou ilustraci tohoto problému na třech dnech vizte obr. 5.1 (červená značí minimální přestávku).



Obrázek 5.1: Nepřekrývání směn

Pro zjednodušení jsou směny uspořádané do datové struktury naznačené na obr. 5.2, tak, aby bylo možné vyhledávat platné kombinace o nějaké délce (dané úvazkem zaměstnance), aby se směny nepřekrývaly.



Obrázek 5.2: Graf následnosti směn

5.5 Délka pracovního týdne

Předpoklad pro algoritmus je co se týče délky pracovního týdne jednoduchý, neřeší možnost plánování přesčasů. Aby byla dodržena délka pracovního týdne (standardně 40 hodin), případně méně (zkrácené úvazky), každému zaměstnanci se rozvrhne nejvýše `úvazek` $*$ `40 / délka směny` směn (zaokrouhleno nahoru). Když se směny ukládají, jedna náhodná z nich se, je-li to třeba, zkrátí, aby byla délka pracovního týdne dodržena.

5.6 Přiřazení na specializovanou směnu

Pro každou specializovanou směnu existuje směna bez specializace, která probíhá ve stejný čas. Zaměstnanec může být přiřazen na libovolnou nespecializovanou směnu nebo na směnu se specializací odpovídající jeho smlouvě.

5.7 Obsazení všech směn

Tato strategie se pokouší do rozvrhu náhodně vybraných zaměstnanců vložit směnu, která zatím nebyla obsazena, a to způsobem popsaným v pseudokódu 5.2. Tento algoritmus proběhne pro každou směnu nejvýše n-krát.

```
Vyber náhodného zaměstnance
POKUD lze vložit směnu do existujícího rozvrhu
  Vlož směnu do rozvrhu
  Smaž protínající nebo náhodnou směnu z rozvrhu
```

Výpis kódu 5.2: Strategie pro vylepšování obsazení všech směn

5.8 Obsazení specializovaných směn

Tato strategie změní přiřazenou směnu na specializovanou, která probíhá v daném čase, je-li to možné vzhledem ke specializaci zaměstnance, vizte pseudokód 5.3 (průběh pro jednu směnu v rozvrhu jednoho zaměstnance).

```
Vyber směnu bez specializace
POKUD probíhá ve stejný čas vhodně specializovaná směna
  Vyměň směnu za specializovanou
```

Výpis kódu 5.3: Strategie pro obsazení specializovaných směn

Úspěch této strategie je zaručen u všech směn, pro něž lze nalézt specializovanou směnu, která probíhá ve stejný čas – v případě, že všechny směny lze specializovat, je sankce za porušení této podmínky vždy nulová. To ale znamená, že některé směny zůstanou zcela prázdné – tedy že naopak tato strategie může zhoršit obsazení všech směn (je ale otázkou, jestli tento fakt má takovou váhu v reálném světě a zda vůbec dává možnost „na tuto směnu může přijít kdokoliv“ smysl v případě, že se směny mají rozepisovat s ohledem na specializaci zaměstnanců).

5.9 Více volna v kuse

Postup pro vylepšení, který se aplikuje na rozvrh každého zaměstnance, který zatím nemá 48hodinovou přestávku mezi směnami, je popsán v pseudokódu 5.4. Především tento algoritmus jednu směnu odstraní (tím vytvoří delší volno), a poté se pokusí umístit další směnu mezi dvě jiné.

```
Nalezni 48h přestávku mezi N-1 směnami
Odstraň směnu v tomto časovém období
POKUD lze směnu vložit do druhé největší přestávky
  Vlož směnu do této přestávky
```

Výpis kódu 5.4: Strategie pro více volna v kuse

5.10 Obsazení směn úměrně poptávce

Nalezne se zaměstnanec, který je zapsán na směnu, kde naplnění převyšuje poptávku a n náhodných směn, které poptávku nenaplňují – snahou je vyměnit přeplněnou směnu za prázdnější (pseudokód 5.5).

```
Vyber náhodného zaměstnance na směnu
POKUD lze tuto směnu vyměnit za méně obsazenou
  Vyměň směnu za méně obsazenou
```

Výpis kódu 5.5: Strategie pro obsazení směn dle poptávky

5.11 Testování

Při testování heuristických algoritmů nelze kontrolovat, zda byl vrácen jeden konkrétní výsledek (mimo jednoduché případy) a je možné pouze určovat, zda je řešení dost dobré, dost rychlé a validní. Může proběhnout sběr testovacích dat z místa reálného nasazení algoritmu, případně mohou být data generována náhodně. [37]

Cílem bylo ověřit na testovacích datech následující předpoklady:

1. Algoritmus vrací pro validní vstupy validní řešení.
2. Strategie snižuje sankci v daném kritériu.
3. Čím více strategií, tím je nižší sankce.
4. Čím více iterací zlepšování, tím lepší řešení.

Pro otestování bylo vytvořeno 8 různých organizací, které se odlišovaly v počtu zaměstnanců, počtu specializací, pracovní době, počtu směn nebo prioritě směn. Kritérium pro výběr testovacích dat bylo spíše náhodné, pouze byla snaha použít data, která se v daných parametrech od sebe odlišují. Pro každou organizaci následně proběhlo testování s různými parametry (počet iterací, váhy kritérií) ve více opakováních.

5.11.1 Definice pojmů

Co se týče prvního předpokladu, řešení bude považováno za validní, pokud algoritmus proběhne bez vyvolání výjimky a bez zacyklení, a pokud se řešení podaří uložit do databáze.

Pro vyhodncení zbývajících předpokladů budou porovnávány průměrné hodnoty ze všech testovacích organizací. Bude-li o řešení řečeno, že je lepší, znamená to, že je sankce

(v daném kritériu či celkově) nižší než sankce náhodného řešení, na kterém žádné další operace neproběhly (tj. proběhlo 0 iterací vylepšování nebo jsou váhy všech kritérií nulové). Sankce za jedno libovolné porušení dle definice cílové funkce je 1. To znamená, že pokud v rozvrhu např. zůstanou tři směny neobsazené, bude sankce 3.

5.11.2 Vyhodnocení předpokladů

Data, o která se následující vyhodnocování opírá, lze nalézt v příloze C. První předpoklad byl vyvrácen tím, že byla nalezena vstupní data (Organizace G, vizte přílohu C), pro která může být jak vyvolána výjimka, tak může algoritmus proběhnout korektně (vizte tab. 5.2). Z dat vyplývá, že tato chyba nastane, ať už jsou použity strategie na vylepšení rozvrhu či nikoli, to nasvědčuje tomu, že se jedná o chybu při implementaci vyhledávání směn v grafu, konkrétně v případě, že některé směny bez specializace mají nulovou prioritu. V případě jiných vstupních dat je algoritmus vždy z tohoto hlediska úspěšný.

Tabulka 5.2: Chybovost algoritmu pro organizaci G

Celkem	Korektní	Nekorektní
617	550	67

Co se druhého předpokladu týče, data ukazují, že strategie pro obsazení všech směn není příliš úspěšná v případě, že se rozvrhují specializované směny – data nasvědčují tomu, že rozvrh v tomto parametru spíše dokáže vylepšit strategie pro obsazení specializovaných směn, když proběhne samostatně, vizte tab. 5.3 udávající průměrné sankce ze všech organizací.

Tabulka 5.3: Sankce za obsazení všech směn

Typ	Průměr
Náhodné	15,87
Obs. všech	15,18
Obs. specializovaných	14,13

Samotná strategie pro obsazení specializovaných směn je úspěšná – sankci oproti náhodnému řešení snižuje ve všech případech výrazně, vizte tab. 5.4.

Tabulka 5.4: Sankce za obsazení specializovaných směn

Typ	Průměr
Náhodné	43,19
Obs. specializovaných	25,06

Strategie pro obsazení směn úměrně poptávce je mírně úspěšná, vizte tab. 5.5.

Strategie, která má zajišťovat více dní volna v kuse, je úspěšná, pokud probíhá izolovaně, nikoli však v kombinaci s dalšími strategiemi (vizte tab. 5.6), nebyla však dostatečně testována varianta, kdy je váha tohoto kritéria vyšší oproti ostatním.

Druhý předpoklad tak spíše platí v případě obsazení specializovaných směn, v případě strategie pro více volných dní v kuse a v případě obsazování směn úměrně poptávce. Bylo by třeba zrevidovat způsob, jakým je implementována strategie pro obsazení volných směn, aby správně obsazovala i směny se specializací.

Tabulka 5.5: Sankce za obsazení specializovaných směn

Typ	Průměr
Náhodné	54,11
Obs. úměrně poptávce	49,96

Tabulka 5.6: Sankce pro více volna v kuse

Typ	Průměr
Náhodné	10,38
Pouze volno v kuse	7,36
Všechny	9,38

Třetí předpoklad data spíše prokazují, sankce je u jednotlivých kritérií nižší, pokud dané vylepšování proběhne, výjimkou může být případ po sobě jdoucích dnů volna, zdá se ale, že jiné strategie se mezi sebou neovlivňují negativně, spíše se naopak podporují.

Tabulka 5.7: Průměrné sankce po proběhnutí všech strategií

Kritérium	Náhodné	Vylepšované
Obs. všech	15,87	11,99
Obs. úměrně poptávce	54,11	41,31
Obs. specializovaných	43,19	22,38
Volno v kuse	10,38	9,38

Co se týče čtvrtého předpokladu, data jednoznačně prokazují, že řešení po jedné iteraci je lepší než náhodné řešení, jiný jasný závěr by si ale vyžádal přesnější posouzení. Získaná data mírně nasvědčují tomu, že toto tvrzení z hlediska všech sankcí platí (je patrný velmi mírný pokles součtu průměrných sankcí všech řešení pro všechny organizace, vizte tab. 5.8), ale rozdíl mezi tím, zda proběhne jedna iterace či více iterací je nevýrazný. Lze předpokládat, že řešení začne velmi brzy konvergovat, tedy že už další zlepšování rozvrhu těmito operacemi nebude dávat smysl.

5.12 Vyhodnocení algoritmu

Byl navržen triviální algoritmus na rozdělení předem určených směn mezi zaměstnance s různě velkým úvazkem. V případě budoucího použití by bylo třeba dále jej vylepšit (především revidovat, které situace byly při implementaci opomenuty) a vzít v potaz, co by od rozvrhovacího algoritmu očekával ten, kdo by jej využival – skoro jistě by se jednalo o složitější optimalizační úlohu, která by vyžadovala odlišný model.

Z testování na několika vybraných „organizacích“ vyplynulo, že algoritmus není bezchybný, je však schopen ve většině případů generovat rozvrhy, které dávají smysl vzhledem k nezbytným podmínkám, a navíc je i lehce upravit, aby lépe naplňovaly zbytné podmínky, tedy aby řešení bylo kvalitnější.

Tabulka 5.8: Součet sankcí při různém počtu opakování

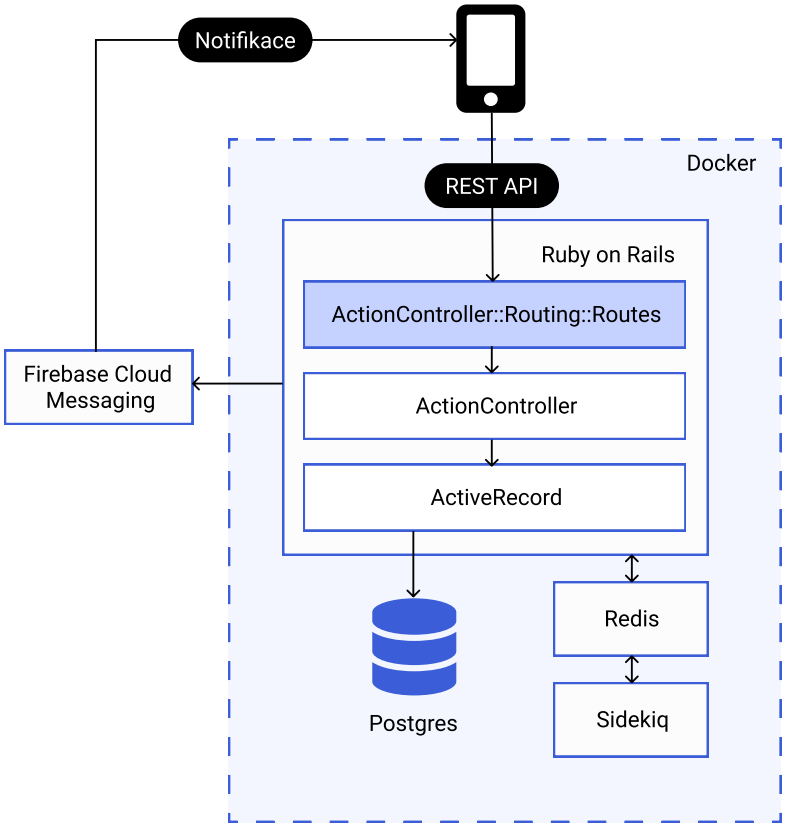
Iterací	0	1	2	3	4
Součet	988,54	830,86	812,83	807,90	803,22

Kapitola 6

Implementace

6.1 Architektura aplikace

Při implementaci byla použita třívrstvá architektura (klient, aplikační server, databáze), jak je naznačeno na obr. 6.1. Klientem zde je mobilní aplikace, která komunikuje se serverem přes jeho REST API.



Obrázek 6.1: Architektura aplikace

Serverová aplikace běží v Dockeru z důvodu snazšího nasazení na různá prostředí, kromě samotné Rails aplikace a databáze jsou v Dockeru se používají i služby Redis a Sidekiq, a to z důvodu zpracování operací na pozadí, v této aplikaci jde především o pravidelné vytvoření nového týdne pro rozvrhnutí nebo zveřejnění rozvrhu v případě, že to vedoucí neudělá v dostatečném předstihu.

Mimo tuto strukturu stojí Firebase Cloud Messaging, který se používá pro posílání notifikací (např. o zveřejnění nového rozvrhu nebo nového týdne na naplánování) do mobilní aplikace, když si toto server vyžádá.

Pro více informací o implementaci vizte přílohu A, která obsahuje diagram komponent a diagram nasazení.

6.2 Backend

Backendová aplikace využívá architektonický vzor MVC, přičemž uživatelské rozhraní není součástí webové aplikace.

Byla snaha implementovat aplikaci co nejjednodušším způsobem tak, aby v co největší míře byly dodrženy konvence a nebylo třeba ji příliš konfigurovat – controllery jsou implementovány na základě konvencí (vizte kód 6.1 – po zavolání cesty `GET /periods/:id` se automaticky provolá metoda `show`), není-li nutné to udělat jinak.

```
class SchedulingPeriodController < ActionController
  def show
      period = SchedulingPeriod.find(params[:id])
      render json: period
  end
end
```

Výpis kódu 6.1: Konvenční ActionController

Model je implementován s pomocí objektově relačního mapování, které se stará o přístup k databázi. Diagram tříd backendu je součástí přílohy A, Android aplikace přistupuje k obdobně strukturovaným datům.

Mimo strukturu specifickou pro webovou aplikaci stojí `Service` třídy, které se provolávají v případě, že je vhodné některé akce, které nejsou triviální, odsunout mimo controller, a také pomocné datové struktury a strategie pro rozvrhování směn.

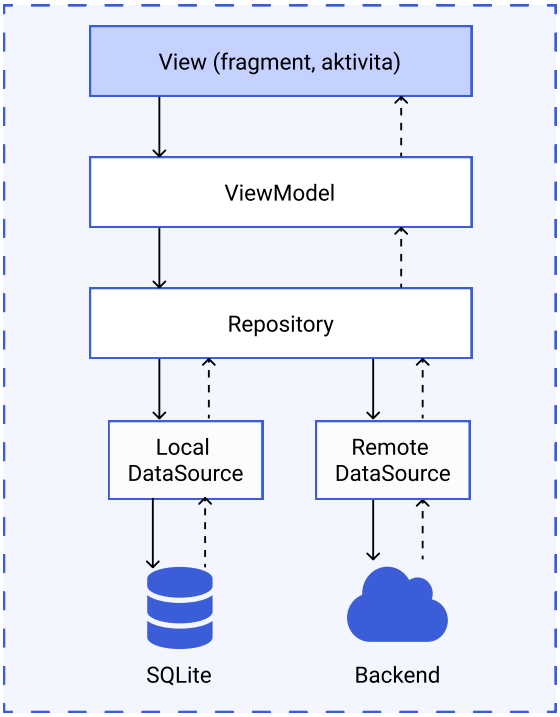
Používají se především gemy[1] Devise Token Auth (autentizace uživatele s pomocí tokenů), CanCanCan (zabezpečení operací se zdroji na základě uživatelské role) nebo will paginate (stránkování zdrojů).

6.3 Mobilní aplikace

Mobilní aplikace byla navrhnuta dle zásad nejpoužívanějšího a ze strany Googlu doporučovaného architektonického vzoru MVVM, vizte obr. 6.2. Při implementaci byly využity především knihovny Koin (pro dependency injection), Retrofit (HTTP klient) a dále některé součásti Android Jetpack (soubor knihoven pro platformu Android, které usnadňují vývoj udržitelných aplikací a poskytují kompatibilitu se staršími verzemi OS [3]), a to především LiveData (pozorovatelná data respektující životní cyklus Android UI komponent), Navigation (navigace mezi fragmenty aplikace), DataBinding (propojení UI komponent definovaných v XML layoutech s daty), Paging (stránkování seznamů) nebo Room (nadstavba nad databází SQLite pro perzistenci dat).

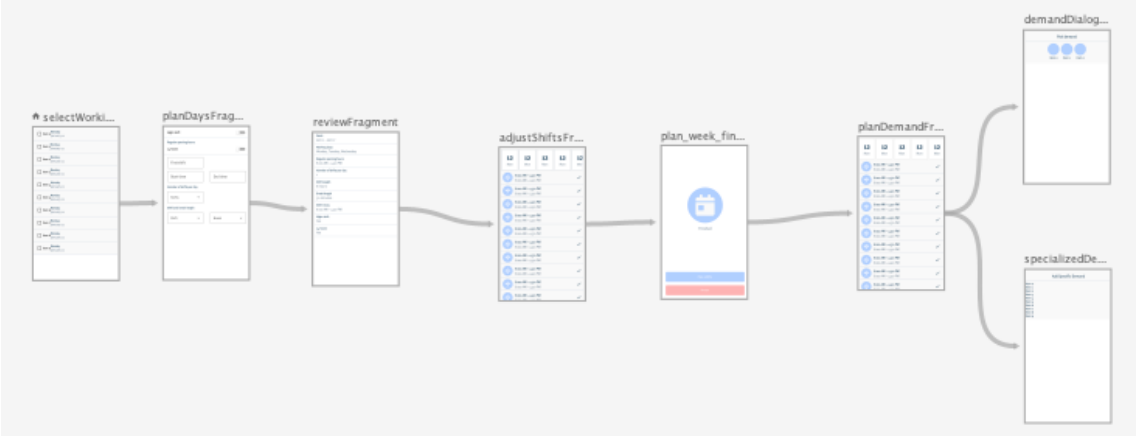
Jedná se o aplikaci s nižším počtem aktivit (samostatné aktivity se využívají v případě, že by nebylo vhodné zobrazovat v aplikaci spodní navigaci, tedy především tehdy, když uživatel vyplňuje formulář, např. přidává nového zaměstnance), v níž navigace probíhá s pomocí Jetpack Navigation (např. detaily směn tedy nejsou samostatnými aktivitami, ale jedná se o fragmenty v aktivitě `MainActivity`). Většina obrazovek má kromě fragmentu

[1] Gem je termín pro knihovny v Ruby.



Obrázek 6.2: Návrh mobilní aplikace

a patřičného layoutu také vlastní ViewModel, existují i sdílené ViewModely pro přenos dat v rámci jedné aktivity. Pro ukázku navigačního grafu vizte obr. 6.3, konkrétně je zde vyobrazena navigace v aktivitě pro přidávání pracovních dní. Obecně navigace v aplikaci probíhá v souladu s hierarchickým popisem, který je součástí přílohy A.



Obrázek 6.3: Ukázka navigačního grafu

6.3.1 View

View je část aplikace, která vykresluje uživatelské rozhraní, může se jednat o layouty ve formátu XML, aktivity (základní okno pro UI) nebo fragmenty (znovupoužitelné). Má referenci na ViewModel a pozoruje jeho změny, podle čehož aktualizuje UI; při událostech na uživatelském rozhraní (např. stisknutí tlačítka) může provolat ViewModel.

Pro zjednodušení se používá data binding, pro vytváření nových aktivit byla vytvořena abstraktní třída `BindingActivity<B: ViewDataBinding>`, pro vytvoření aktivity s layoutem bez dalších parametrů tak dostačuje kód 6.2. Abstraktní aktivita nastaví layout a binding, který je pro potomky přístupný jako proměnná `binding`, a nastaví mu vlastníka životního cyklu.

Třída `ViewModelActivity<V: BaseViewModel, B: ViewDataBinding>` je abstraktním potomkem `BindingActivity`, který je přípraven pro aktivity, které mají vazbu na ViewModel (parametry konstruktoru jsou layout a třída ViewModelu). Tato aktivita zajišťuje vytvoření ViewModelu (který se uloží do proměnné `viewModel` a navíc se nastaví jako parametr pro binding).

```
class SetupActivity :
  BindingActivity<ActivitySetupBinding>(R.layout.activity_setup)
```

Výpis kódu 6.2: Vytvoření jednoduché aktivity

Velmi podobně jako `BindingActivity` a `ViewModelActivity` fungují také abstraktní fragmenty, tj. `BindingFragment` a `ViewModelFragment`.

Vzhledem k použití knihovny LiveData (pozorovatelná data závislá na životním cyklu aktivity/fragmentu) zde větší míře použit návrhový vzor Observer, View pozoruje změny ViewModelu a podle toho se mění obrazovky.

6.3.2 ViewModel

ViewModel je prostředníkem mezi aktivitou a modelem – provolává nižší vrstvu a na základě dat, která mu vrátí, mění stav svých proměnných.

ViewModely jsou v této aplikaci potomkem abstraktní třídy `BaseViewModel`, která v sobě obsahuje odkazy na všechny repositáře (inicializace probíhá s pomocí dependency injection, vizte kód 6.3); také obsahuje LiveData pro zobrazování chyb a metody pro zpracování chyb, případně dat (to vše proto, aby se méně opakoval stejný kód). ViewModel ukládá odpovědi z repozitářů jako LiveData.

```
abstract class BaseViewModel : ViewModel() {
    protected val userRepository by inject(UserRepository::class.java)
    protected val shiftRepository by inject(ShiftRepository::class.java)
    [...]
}
```

Výpis kódu 6.3: Třída `BaseViewModel`

6.3.3 Model

Model se skládá z repozitářů, které mají zodpovědnost za provolávání datových zdrojů (lokálních či vzdálených) a za ukládání dat do databáze či do cache, takto poskytuje požadovaná data. Z repozitářů se data vracejí obalená ve třídě `ResponseModel<T>`, ta může nést zprávu o chybě (která se dále zpracují a zobrazí na UI) nebo požadovaná data.

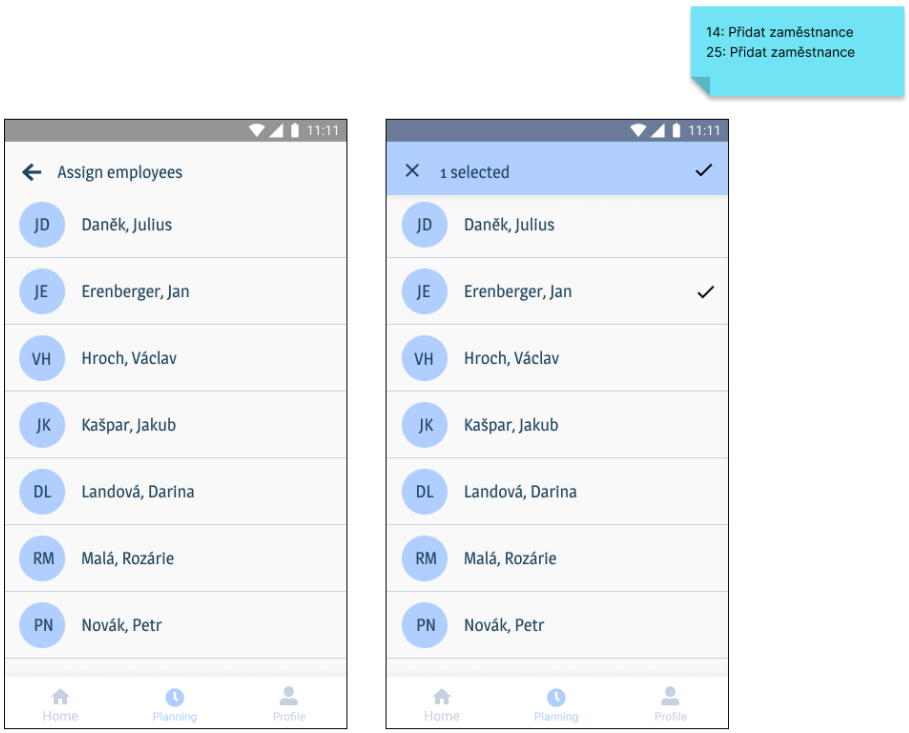
```
sealed class ResponseModel<T> {
  class SUCCESS<T>(var data: T? = null, val headers: Headers? = null):
    ResponseModel<T>()

  class ERROR<T>(val errorType: ErrorType? = null): ResponseModel<T>()
}
```

Výpis kódu 6.4: Třída `ResponseModel`

Většinu dat nedává smysl uchovávat v lokální databázi – ukládají se především rozvrhy směn pro zaměstnance, které jsou považovány za natolik důležité, že by měly být dostupné vždy (primárně se stahují aktuální data ze serveru, pokud to není možné, čte se z databáze).

6.3.4 Design

Design aplikace vznikl v souladu se základními pravidly pro tvorbu uživatelského rozhraní Android aplikací, jak jsou popsány v Material Design Guidelines[2], snahou bylo vytvořit minimalistické uživatelské rozhraní, v němž uživatelé snadno najdou to, co je pro ně důležité. Pro ukázku vizte obr. 6.4, konkrétní návrhy jednotlivých obrazovek jsou v příloze A.



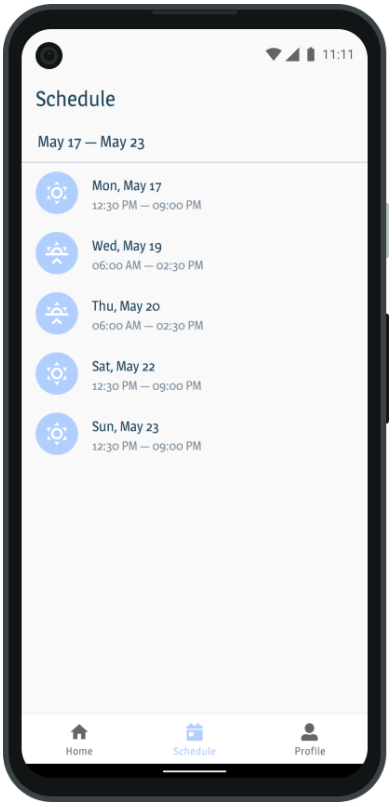
Obrázek 6.4: Ukázka designu

Návrh vznikl v nástroji Figma, ikonky byly převzaty z knihoven Font Awesome 5 a Material Design Icon.

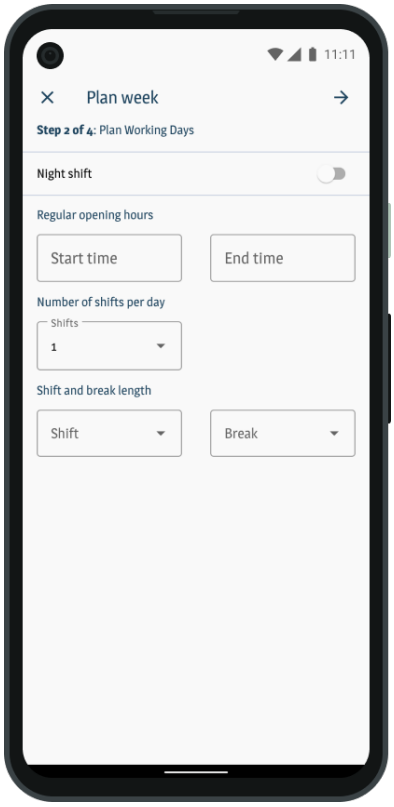
6.3.5 Uživatelské rozhraní aplikace

Konkrétní snímky obrazovky z mobilní aplikace jsou na obr. 6.5 (pro více snímků obrazovky vizte předpokládáný průchod aplikací v příloze B).

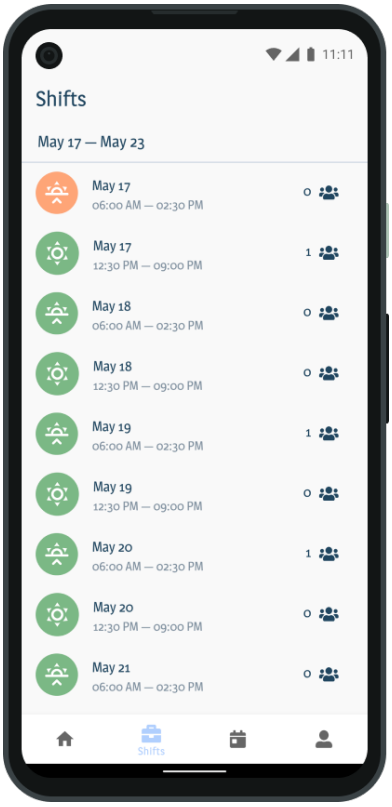
[2] https://material.io/design/guidelines-overview



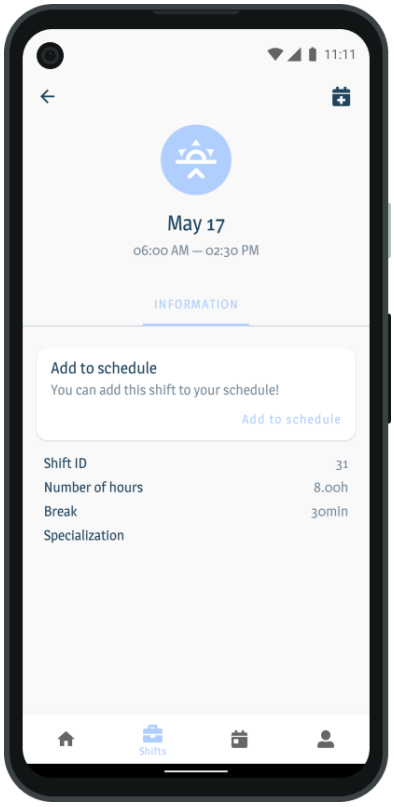
(a) Rozvrh směn



(b) Plánování týdne



(c) Volné směny



(d) Detail směny

Obrázek 6.5: Snímky obrazovky z Android aplikace

Kapitola 7

Uživatelské testování

7.1 Způsob testování

Testování proběhlo jako odlehčená verze testu použitelnosti, probíhalo vzdálenou formou. Účastníkům byl předem poskytnut popis jednotlivých pojmů (vizte přílohu A) a instrukce popisující, co se od nich chce a na co se mají konkrétně zaměřit, dále testovací scénáře, kterými měli projít (vizte přílohu B). Testování bylo kvalitativní, proběhlo na třech účastnících, což je dle [34] optimální počet, aby byly od uživatelů získávány nové podněty. Účastníci byli vybráni dle příslušnosti k cílové skupině či dle znalosti problematiky.

Před testováním byl sestaven předpoklad o průchodu aplikací ve formě po sobě následujících snímků obrazovky, který je součástí přílohy B. Zpráva z testování, obsahující podrobnosti o průchodu uživatelů aplikací, je součástí přílohy B.

7.2 Průchod aplikací

Všichni uživatelé aplikaci prošli podle doporučeného scénáře, před tím si přečetli instrukce a popis aplikace a potvrdili, že všemu rozumí a dává jim to smysl. Při průchodu většinou testovacích scénářů neměli zásadní problémy, aplikace jim připadala intuitivní.

Největším zádrhelem se ukázalo být přihlašování, především uživatelům chyběla možnost obnovit si heslo či si jej zkontrolovat při vyplňování, také jim scházela možnost upravit špatně odeslaná data (smluvy, specializace, uživatelské údaje). Jako matoucí působil průvodce rozvrhováním pracovní doby na daný týden – toto muselo být více vysvětleno, aby se dokázali zorientovat.

7.3 Návrhy na zlepšení

Z rozhovorů s potenciálními uživateli vyplynula celá řada návrhů na zlepšení aplikace. Byly rozděleny do dvou kategorií – připomínky k implementaci aktuálních funkcionalit a návrhy do budoucna (toto rozdělení návrhů je pouze orientační).

7.3.1 Aktuální aplikace

- Překlad aplikace do češtiny,
- změna rozložení či barev tlačítek v registračním formuláři,
- zvýraznění prvků uživatelského rozhraní,

- změna textu některých tlačítek,
- lepší zpětná vazba při zveřejnění rozvrhu,
- plánování rozvrhu na měsíc, nikoli na týden,
- úprava chybových hlášek, aby byly konkrétnější,
- označení povinných a nepovinných polí formuláře.

7.3.2 Návrhy do budoucna

- Umožnění úpravy rozvrhu po zveřejnění (*„Co když někdo třeba onemocní?“*),
- umožnění úpravy odeslaných dat,
- zobrazení kontaktů na uživatele zaměstnancům, přidání telefonního čísla,
- zobrazení rozvrhu ve formě tabulky, nejen seznamu,
- zobrazení celkového rozvrhu všem zaměstnancům,
- podpora platformy iOS.

7.4 Vyhodnocení

Uživatelé aplikací prošli v předpokládaném sledu obrazovek. Shodli se na tom, že by ji dokázali používat, myslí si, že by se s ní naučili a zvykli by si. Ocenili její jednoduchost, snadnost ovládání i to, že se zobrazují pouze informace, které jsou relevantní. Celkové hodnocení je pozitivní, navíc byla poskytnuta řadu návrhů na zlepšení, které by bylo vhodné vzít v potaz, pokud by vývoj dále pokračoval.

Závěr

V této práci byl představen způsob, jakým by bylo možné vylepšit interní procesy organizace, která rozvrhuje směny svým zaměstnancům na základě sady různých pravidel, ať už vyplývají z legislativy, personální situace ve firmě nebo z manažerských požadavků, a to prostřednictvím jednoduché mobilní aplikace, která bude dostupná jak vedoucím, tak zaměstnancům.

V úvodní kapitole teoretické části bylo popsáno plánování pracovních sil z hlediska manažerského rozhodování (jaké faktory jej mohou ovlivnit) a z hlediska legislativního, kdy je třeba rozvrhovat s ohledem na stanovenou týdenní pracovní dobu, s ohledem na druh pracovněprávního vztahu nebo s ohledem na přestávky mezi směnami.

Ve druhé kapitole dále byly popsány některé teoretické metody, které jsou vhodné pro optimalizaci rozvrhů směn na základě rozhodujích faktorů – může se jednat o tradiční deterministické metody pro optimalizaci cílové funkce, případně o metaheuristiky, které se přizpůsobí problému rozvrhování směn.

Ve třetí kapitole byla vytvořena představa o organizacích cílové skupiny, na základě toho byly sestaveny požadavky na novou aplikaci, která by zjednodušila interní procesy týkající se rozvrhování směn.

Ve čtvrté kapitole bylo popsáno rozhodování o technologiích pro implementaci uživatelského rozhraní a serverové části aplikace, na základě toho byly stanoveny požadavky na kvalitu software.

V páté kapitole byl přiblížen návrh algoritmu, který se v této aplikaci používá pro automatické rozvrhování směn – tento algoritmus byl inspirován již existujícími teoretickými metodami, a tak se rozvrhuje na základě řady podmínek, které byly rozděleny na zbytné a nezbytné. Nezbytné mají za úkol kontrolovat, zda řešení dává smysl z hlediska legislativy (zda má dostatečnou přestávku), případně obecně (zda například zaměstnanec nemá být na dvou místech zároveň), zbytné určují kvalitu na základě požadavků manažerů (vždy je někdo na pracovišti) a zaměstnanců (volné dva dny v kuse).

Šestá kapitola popisovala způsob, jakým je navržená aplikace implementována, a to jak její backendová část, pro kterou byl použit webový framwork Ruby on Rails, aby byl vývoj v co největší míře usnadněn, tak i Android aplikace, která se snaží v co největší míře použít moderní postupy vývoje.

Poslední kapitola byla věnována uživatelskému testování, které jak potvrdilo vůli potenciálních uživatelů takovou aplikaci používat, tak přineslo řadu nových návrhů, které uživatele při průchodu aplikací napadaly.

Cíl práce, jímž byla analýza, návrh a implementace mobilní aplikace pro plánování lidských zdrojů, byl celkově naplněn, aplikace byla otestována na několika potenciálních uživatelích, kteří vyjádřili ochotu podobnou aplikaci využívat, poskytli pozitivní zpětnou vazbu a řadu návrhů, které by bylo vhodné zapracovat v případě, že by práce byla dále rozšiřována.

Prokázalo se tak, že by bylo možné takovou podpůrnou aplikaci pro rozvrhování vyvinout, obzvlášť pokud by se jednalo o komerční projekt, který byl měl větší rozpočet,

více času a větší tým analytiků a vývojářů. V tom případě by bylo vhodné aplikaci rozšířit a dále vylepšit na základě důkladnějšího sběru požadavků od všech zúčastněných stran.

Druhou důležitou součástí této práce byl návrh vlastního algoritmu pro rozvrhování směn mezi zaměstnance, způsob jeho implementace reflektuje fakt, že se jedná pouze o součást bakalářské práce, je tedy navržen v triviální podobě, která není zdaleka připravená na reálné nasazení, přesto byl i tento cíl, tedy umožnit v aplikaci automatické rozvrhování směn na základě vybraných podmínek, naplněn. Pokud by měl být systém, který umožňuje automatické rozvrhování, nasazen v reálném prostředí, bylo by třeba zohlednit daleko větší množství podmínek a požadavků, a algoritmus jim přizpůsobit.

Na tomto místě je vhodné podotknout, že systém pro rozvrhování pracovních sil zcela určitě nestojí ve vakuu, ale naopak řeší problém, který je ryze praktický, a setkává se s ním řada potenciálních uživatelů ve svém každodenním životě. Softwarové řešení dává smysl, ale realita je vždy složitější, než aby šlo všechny varianty, které mohou nastat, ať už z hlediska legislativy, tak z hlediska lidského faktoru, pokrýt aplikací, kterou vývojáři musí připravit v konečném čase.

Součástí této práce je řada příloh, které obsahují kromě samotných zdrojových kódů aplikací (mobilní a webové) také podrobnější analýzu, návrh uživatelského rozhraní nebo výsledky testování.

Bibliografie

1. ADAMUTHE, Amol C; BICHKAR, Rajankumar S. Tabu search for solving personnel scheduling problem. In: *2012 International Conference on Communication, Information & Computing Technology (ICCICT)*. 2012, s. 1–6.
2. ANDROID. *Guide to app architecture* [online] [cit. 2020-12-28]. Dostupné z: `https://developer.android.com/jetpack/guide`.
3. ANDROID. *Jetpack* [online] [cit. 2020-12-31]. Dostupné z: `https://developer.android.com/jetpack`.
4. ANDROID. *Develop Android apps with Kotlin* [online] [cit. 2021-04-22]. Dostupné z: `https://developer.android.com/kotlin`.
5. ARMSTRONG, Michael; TAYLOR, Stephen. *Armstrong's handbook of human resource management practice*. 13. vyd. Kogan Page Publishers, 2014.
6. AWADALLAH, Mohammed A et al. A hybrid artificial bee colony for a nurse rostering problem. *Applied Soft Computing*. 2015, roč. 35, s. 726–739.
7. BECHTOLD, Stephen E. Work force scheduling for arbitrary cyclic demands. *Journal of Operations Management*. 1981, roč. 1, č. 4, s. 205–214.
8. BLÖCHLIGER, Ivo. Modeling staff scheduling problems. A tutorial. *European Journal of Operational Research*. 2004, roč. 158, č. 3, s. 533–542.
9. BURKE, Edmund K et al. The state of the art of nurse rostering. *Journal of scheduling*. 2004, roč. 7, č. 6, s. 441–499.
10. BUYUKOZKAN, Kadir; SARUCAN, Ahmet. Applicability of artificial bee colony algorithm for nurse scheduling problems. *International Journal of Computational Intelligence Systems*. 2014, roč. 7, č. sup1, s. 121–136.
11. CIPD. *Workforce planning factsheet.* [Online] [cit. 2020-10-18]. Dostupné z: `https://cipd.co.uk/knowledge/strategy/organisational-development/workforce-planning-factsheet`.
12. DE CAUSMAECKER, Patrick; BERGHE, Greet Vanden. A categorisation of nurse rostering problems. *Journal of Scheduling*. 2011, roč. 14, č. 1, s. 3–16.
13. DENNIS, Megan. *Native vs. Cross-Platform Apps – You'll Be the Winner* [online]. 2018-12-21 [cit. 2021-04-22]. Dostupné z: `https://zeolearn.com/magazine/native-vs-cross-platform-apps-youll-be-the-winner`.
14. DESIGNRUSH. *The Ultimate Guide To Hybrid App Development* [online]. 2020-11-27 [cit. 2021-04-23]. Dostupné z: `https://designrush.com/trends/hybrid-mobile-app-development`.

15. DOCFORGE. *Web application framework* [online] [cit. 2021-04-20]. Dostupné z: https://web.archive.org/web/20150723163302/http://docforge.com/wiki/Web_application_framework.

16. ERNST, Andreas T et al. Staff scheduling and rostering: A review of applications, methods and models. *European journal of operational research*. 2004, roč. 153, č. 1, s. 3–27.

17. FLO, Elisabeth et al. Shift-related sleep problems vary according to work schedule. *Occupational and environmental medicine*. 2013, roč. 70, č. 4, s. 238–245.

18. GEEKS FOR GEEKS. *Difference between SQL and NoSQL* [online]. 2020-12-23 [cit. 2021-04-22]. Dostupné z: https://geeksforgeeks.org/difference-between-sql-and-nosql/.

19. GLOVER, Fred. Tabu search: A tutorial. *Interfaces*. 1990, roč. 20, č. 4, s. 74–94.

20. GLOVER, F.; SÖRENSEN, K. Metaheuristics. *Scholarpedia*. 2015, roč. 10, č. 4, s. 6532. Dostupné z DOI: 10.4249/scholarpedia.6532. revision #149834.

21. GOOGLE. *Developer Program Policy (effective January 20, 2021)* [online] [cit. 2021-04-09]. Dostupné z: https://support.google.com/googleplay/android-developer/answer/10355942?hl=en.

22. JAIN, Rakshit. *Kotlin Clean Architecture*. 2019-02-17. Dostupné také z: https://proandroiddev.com/kotlin-clean-architecture-1ad42fcd97fa.

23. KLETZANDER, Lucas; MUSLIU, Nysret. Solving the general employee scheduling problem. *Computers & Operations Research*. 2020, roč. 113, s. 104794.

24. KOVACH, Kenneth A et al. Administrative and strategic advantages of HRIS. *Employment Relations Today*. 2002, roč. 29, č. 2, s. 43–48.

25. LIANG, Frank. *Optimization Techniques — Tabu Search* [online]. 2020-07-27 [cit. 2021-04-29]. Dostupné z: https://towardsdatascience.com/optimization-techniques-tabu-search-36f197ef8e25.

26. LIN, Dingding et al. Scheduling workforce for retail stores with employee preferences. In: *2015 IEEE International Conference on Service Operations And Logistics, And Informatics (SOLI)*. 2015, s. 37–42.

27. LISITSKY, Eugene. *Does Heroku support RPC (i.e. gRPC)?* [Stack Overflow]. 2018 [cit. 2021-04-27]. Dostupné z: https://stackoverflow.com/a/52114161.

28. MAENHOUT, Broos; VANHOUCKE, Mario. An evolutionary approach for the nurse rerostering problem. *Computers & Operations Research*. 2011, roč. 38, č. 10, s. 1400–1411.

29. MALLAWAARACHCHI, Vijini. *Introduction to Genetic Algorithms — Including Example Code* [online]. 2017-07-08 [cit. 2021-04-29]. Dostupné z: https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3.

30. MANCHANDA, Amit. *Where Do Cross-Platform App Frameworks Stand in 2021?* [Online]. 2020-12-08 [cit. 2021-04-22]. Dostupné z: https://netsolutions.com/insights/cross-platform-app-frameworks-in-2019/.

31. MATOUŠEK, Jiří. *Lineární programování: Úvod pro informatiky* [online]. KAM MFF UK, 2006 [cit. 2020-11-11]. Dostupné z: https://iti.mff.cuni.cz/series/2006/311.pdf.

32. MATYUNINA, Julia. *Native vs Hybrid vs Web App – Startup Hacks* [online]. 2020-11-24 [cit. 2021-04-23]. Dostupné z: https://codetiburon.com/choose-mobile-development-platform-startup-hacks/.

33. MUSLIU, Nysret et al. Efficient generation of rotating workforce schedules. *Discrete Applied Mathematics*. 2002, roč. 118, č. 1-2, s. 85–98.

34. NIELSEN, Jakob. *Why You Only Need to Test with 5 Users* [online]. 2000-03-18 [cit. 2021-05-07]. Dostupné z: https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/.

35. RUBYONRAILS.ORG. *Getting Started with Rails* [online] [cit. 2020-12-22]. Dostupné z: https://guides.rubyonrails.org/getting_started.html.

36. RAMLI, Razamin et al. A tabu search approach with embedded nurse preferences for solving nurse rostering problem. *International Journal for Simulation and Multidisciplinary Design Optimization*. 2020, roč. 11, s. 10.

37. RARDIN, Ronald L; UZSOY, Reha. Experimental evaluation of heuristic optimization algorithms: A tutorial. *Journal of Heuristics*. 2001, roč. 7, č. 3, s. 261–304.

38. SATHEESHKUMAR, B et al. Linear programming applied to nurses shifting problems. *International Journal of science and research*. 2014, roč. 3, č. 3, s. 171–173.

39. SHEKHAR, Amit. *Understand How does Retrofit work* [online]. 2020-03-04 [cit. 2020-12-28]. Dostupné z: https://blog.mindorks.com/mvvm-architecture-android-tutorial-for-beginners-step-by-step-guide.

40. STATCOUNTER. *Mobile Operating System Market Share Worldwide* [online] [cit. 2021-04-09]. Dostupné z: https://gs.statcounter.com/os-market-share/mobile/worldwide.

41. STEVENS, Emily. *What Is The Difference Between A Mobile App And A Web App?* [Online]. 2018-04-03 [cit. 2021-04-23]. Dostupné z: https://careerfoundry.com/en/blog/web-development/what-is-the-difference-between-a-mobile-app-and-a-web-app/.

42. STURGEON, Phil. *Understanding RPC Vs REST For HTTP APIs* [online]. 2016-09-20 [cit. 2021-04-25]. Dostupné z: https://smashingmagazine.com/2016/09/understanding-rest-and-rpc-for-http-apis/.

43. TAMIGO. *tamigo – Řešení* [online] [cit. 2020-12-24]. Dostupné z: https://tamigo.cz/reseni.

44. TANDA. *Rosters* [online] [cit. 2020-12-26]. Dostupné z: https://tanda.co/features/rosters/.

45. TODOROVIC, Nikola; PETROVIC, Sanja. Bee colony optimization algorithm for nurse rostering. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*. 2012, roč. 43, č. 2, s. 467–473.

46. VÁCLAVÍK, Roman et al. Roster evaluation based on classifiers for the nurse rostering problem. *Journal of Heuristics*. 2016, roč. 22, č. 5, s. 667–697.

47. VAN DEN BERGH, Jorne et al. Personnel scheduling: A literature review. *European journal of operational research*. 2013, roč. 226, č. 3, s. 367–385.

48. VOGEL, Lars. *Android Architecture with MVP or MVVM - Tutorial* [online]. 2017-04-18 [cit. 2020-12-29]. Dostupné z: `https://vogella.com/tutorials/AndroidArchitecture/article.html`.

49. WHEN I WORK. *Free employee scheduling app* [online] [cit. 2020-12-26]. Dostupné z: `https://wheniwork.com/features/employee-scheduling-software/`.

50. WHEN I WORK. *Pricing* [online] [cit. 2020-12-26]. Dostupné z: `https://wheniwork.com/pricing/`.

51. Zákon č. 262/2006 Sb., zákoník práce.

52. ZUCCHI, Giorgio et al. Personnel scheduling during Covid-19 pandemic. *Optimization Letters*. 2020, s. 1–12.

Přílohy

A Analýza systému

- Popis pojmů
- Diagram tříd
- Diagram nasazení
- Diagram komponent
- Hierarchie obrazovek, wireframes
- Návrh uživatelského rozhraní

B Uživatelské testování

- Instrukce k uživatelskému testování
- Testovací scénáře
- Zpráva z testování
- Předpokládaný průchod aplikací

C Testování algoritmu

- Tabulka s daty z testování

D Zdrojové kódy

- Android aplikace
- Ruby on Rails aplikace