FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Návrh a implementace webové aplikace pro obsazování míst v kanceláři
Student:	Jan Dunder
Vedoucí:	doc. Ing. Robert Pergl, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Informační systémy a management
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2024/2025

Pokyny pro vypracování

Cílem práce je návrh a implementace webové aplikace v PHP Symfony sloužící pro optimalizaci obsazování míst v kanceláři - přiřazování osob k místům na určité dny a časy.

1. Proveďte rešerši potřebných teoretických základů a příp. podobných existujících aplikací.
2. Proveďte analýzu firemních procesů spojených s problematikou a navrhněte budoucí stav s použitím aplikace.
3. Vytvořte návrh webové aplikace.
4. Aplikaci implementujte a řádně otestujte.
5. Diskutujte výsledek, zejména s ohledem na přínosy pro firmu.

Elektronicky schválil/a Ing. David Buchtela, Ph.D. dne 19. dubna 2023 v Praze.

Bakalářská práce

NÁVRH A IMPLEMENTACE WEBOVÉ APLIKACE PRO OBSAZOVÁNÍ MÍST V KANCELÁŘI

Jan Dunder

Fakulta informačních technologií
Katedra softwarového inženýrství
Vedoucí: doc. Ing. Robert Pergl, Ph.D.
10. ledna 2024

České vysoké učení technické v Praze
Fakulta informačních technologií
© 2024 Jan Dunder. Všechna práva vyhrazena.
Tato práce vznikla jako školní dílo na Českém vysokém učení technickém v Praze, Fakultě informačních technologií. Práce je chráněna právními předpisy a mezinárodními úmluvami o právu autorském a právech souvisejících s právem autorským. K jejímu užití, s výjimkou bezúplatných zákonných licencí a nad rámec oprávnění uvedených v Prohlášení, je nezbytný souhlas autora.

Odkaz na tuto práci: Dunder Jan. *Návrh a implementace webové aplikace pro obsazování míst v kanceláři.* Bakalářská práce. České vysoké učení technické v Praze, Fakulta informačních technologií, 2024.

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů kódu

Rád bych poděkoval především doc. Ing. Robertu Perglovi, Ph.D., vedoucímu této práce, za jeho jeho odborné vedení, cenné rady a pozitivní přístup během celého procesu vytváření této bakalářské práce. Velké poděkování patří také Ondřejovi Škrdlantovi, který mi pomohl navrhnout téma bakalářské práce a jehož úvodní postřehy a rady, jak se postavit k vývoji webové aplikace, pro mě byly velmi užitečné a přispěly ke zlepšení kvality mé práce.
Dále děkuji Ing. Janu Zemanovi za jeho laskavou nabídku odborně oponovat tuto práci i přes jeho velké pracovní vytížení.
Nakonec bych chtěl také poděkovat své rodině a přátelům za jejich neustálou podporu a povzbuzení během tvorby této bakalářské práce i předešlého studia včetně ochotné nabídky účastnit se uživatelského testování vyvíjené webové aplikace.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

V Praze dne 10. ledna 2024

Abstrakt

Tato bakalářská práce se věnuje návrhu a implementaci webové aplikace pro efektivní obsazování míst v kancelářích, reagující na potřeby firem v éře flexibilních pracovních režimů a rostoucích nákladů na kancelářské prostory. Práce analyzuje stávající řešení a identifikuje klíčové firemní procesy, na jejichž základě stanovuje požadavky na aplikaci. Dle sepsaných požadavků je dále součástí práce návrh, implementace a testování webové aplikace v PHP Symfony. Cílem je poskytnout řešení pro zvýšení efektivity využití kancelářských prostor, snížení nákladů a zvýšení produktivity.

Klíčová slova obsazenost pracovních míst, vizualizace kanceláří, webová aplikace, API Platform, PHP, Symfony, React

Abstract

This bachelor thesis focuses on the design and implementation of a web application for efficient office staffing, responding to the needs of companies in an era of flexible working schedules and rising office space costs. The thesis analyses existing solutions and identifies key business processes to determine the requirements for the application. According to the written requirements, the work also includes the design, implementation and testing of a web application in PHP Symfony. The aim is to provide a solution to increase the efficiency of office space utilization, reduce costs and increase productivity.

Keywords workplace occupancy, office visualization, web application, API Platform, PHP, Symfony, React

Seznam zkratek a pojmů

API	Application Programming Interface – rozhraní pro přístup k funkcím aplikace
API endpoint	Cílový (koncový) bod API pro specifické operace (např. CRUD operace)
API Platform	Webový framework založený na Symfony, navržený pro snadné vytváření projektů založených na rozhraní API
API token	Bezpečnostní klíč pro ověření v API (typicky používaný při requestech na API endpoint
AS-IS	V kontextu firemních procesů označení současného stavu
CSS	Cascading Style Sheets – kaskádové styly
CRUD operace	Create (vytvoření), Read (čtení), Update (úprava) a Delete (smazání) operace
DELETE request	Požadavek na smazání zdroje v API
Doctrine	Systém pro správu objektově-relačního mapování (ORM) sloužící pro automatizaci interakcí s databází
EasyAdmin	Administrační bundle pro Symfony
Engine	Základní software nebo mechanizmus
Figma	Populární webový nástroj pro design
Framework	Aplikační rámec – struktura pro podporu programování a vývoje softwaru
GET request	Požadavek na získání dat z API
GET COLLECTION request	Požadavek na získání kolekce dat z API
GCal	Google kalendář
HTML	Hypertext Markup Language – standardní značkovací jazyk pro webové stránky
Javascript	Programovací jazyk pro webové aplikace
MS Teams	Microsoft Teams – komunikační platforma od společnosti Microsoft pro týmovou spolupráci (používaná především v korporátním prostředí)
Outlook	Osobní informační manažer od společnosti Microsoft nabízející především správu e-mailů, kalendář, správu úkolů, kontakty a poznámky
Panther	Testovací nástroj pro Symfony aplikace
PATCH request	Požadavek na částečnou úpravu zdroje v API
PHP	Skriptovací programovací jazyk
POST request	Požadavek na vytvoření zdroje v API
PostgreSQL	Objektově relační databázový systém
PUT request	Požadavek na úpravu zdroje v API
React	JavaScriptová knihovna pro uživatelské rozhraní
Request	Požadavek (dotaz) na aplikaci (webový server)
REST	Representational State Transfer – způsob, jak jednoduše provádět CRUD operace pomocí jednoduchých HTTP volání
REST API	Standardizované rozhraní API, které splňuje zásady návrhu architektonického stylu REST
Response	Odpověď serveru na požadavek
Slack	komunikační platforma určená pro podniky a týmy
Software	Programy a aplikace pro počítače
Symfony	Framework pro vývoj webových aplikací v PHP
TO-BE	V kontextu firemních procesů označení budoucího stavu
Twig	Šablonovací engine pro PHP, distribuován jako součást Symfony

Kapitola 1
Úvod

V době flexibilních pracovních režimů a rostoucích nákladů na kancelářské prostory, zejména ve velkých městech, jako je Praha, čelí společnosti stále častěji výzvě efektivního využití kancelářských prostor. Typickou situací je stav, kdy firma zaměstnává větší množství osob, které pracují na dohodu o provedení práce, dohodu o pracovní činnosti, mají zkrácený pracovní úvazek nebo pracují na živnostenský list. V kombinaci se stále populárnější prací z domova, tzv. „home office“, není výjimkou, že někteří zaměstnanci tráví v kanceláři pouze jeden nebo dva dny v týdnu.

V takové situaci je pochopitelně ekonomicky nevýhodné, aby měl každý zaměstnanec své vlastní vybavené pracovní místo v kancelářích společnosti. Každé takové pracovní místo představuje finanční zátěž jak v podobě počátečních jednorázových výdajů (pracovní židle a stůl, při práci s počítačem také monitory, dokovací stanice, klávesnice, myš apod.), tak i v průběhu času (pronájem kanceláře, vytápění kanceláře, spotřeba elektřiny, úklid).

Je proto v zájmu každé firmy optimalizovat počet pracovních míst tak, aby měl každý zaměstnanec možnost pracovat během své pracovní doby z kanceláře, ale zároveň byl počet pracovních míst minimální. Bez účinného systému mohou snahy o minimalizaci pracovních míst vést k narušení pracovní harmonie (resp. ke konfliktům mezi zaměstnanci) a k neefektivnímu využívání kancelářských prostor. V důsledku toho pak může docházet k falešnému pocitu nedostatku pracovních míst, který je ovšem způsoben pouze zmíněnou neefektivitou a absencí relevantních dat a statistik o vytíženosti kanceláří.

Tato bakalářská práce se zaměřuje nejen na návrh a implementaci webové aplikace pro obsazování míst v kanceláři, ale také na identifikaci klíčových firemních procesů a z nich vyplývajících požadavků na aplikaci, které jsou s touto problematikou spjaty.

Součástí této práce je také analýza stávajících řešení na trhu, která jsou často finančně velmi nákladná, a přesto nesplňují některé z identifikovaných požadavků.

Hlavním cílem této práce je poskytnout řešení nastíněného problému formou webové aplikace, která umožní firmám efektivněji spravovat jejich pracovní prostory, což je v dnešním dynamickém pracovním prostředí klíčové pro snížení nákladů a zvýšení produktivity. Potenciální volná dostupnost zdrojového kódu (jeho zpřístupnění jako open-source) po obhajobě této bakalářské práce poskytne možnost budoucího rozvoje aplikace, který je ještě nepochybně nezbytný k tomu, aby aplikace mohla obstát mezi robustními a velmi nákladnými alternativami (již existujícími aplikacemi).

Kapitola 2
Cíle práce

Cílem teoretické části práce je provést rešerši existujících aplikací a tyto aplikace porovnat s ohledem na nabízené funkce a cenu těchto aplikací. Na základě zjištěných poznatků provést rešerši potřebných teoretických základů a navrhnout a popsat vhodné technologie pro vypracování praktické části práce s ohledem na očekávané funkce, které by aplikace měla nabízet (jejich analýza je součástí praktické části práce).

Cílem praktické části je na základě provedené rešerše zpracovat analýzu existujících řešení, tato řešení zhodnotit a diskutovat jejich přínos, klady a zápory. Poté zhodnotit, zda a za jakých podmínek, se firmě vyplatí investovat do vývoje vlastního řešení.

D8le identifikovat klíčové procesy související se správou míst v kancelářích a na základě osobních pracovních zkušeností z firmy popsat obvyklý stávající stav těchto procesů. Poté provést analýzu těchto procesů a navrhnout možný budoucí stav daných procesů za předpokladu, že se firma rozhodne používat vyvíjenou aplikaci. Jednotlivé procesy a jejich stavy popsat nejen slovně, ale také pomocí BPMN diagramů.

Dle analyzovaných firemních procesů a navržených budoucích stavů popsat požadavky na aplikaci, které jsou nezbytné k tomu, aby mohlo být těchto budoucích stavů dosaženo. Součástí popisu požadavků by také mělo být určité zhodnocení, které požadavky jsou zcela nezbytné a které jsou sekundární (např. pouze vylepšují uživatelskou přívětivost aplikace).

Poté vytvořit vlastní webovou aplikaci, která bude řešit problém optimalizace obsazování míst v kancelářích a nabízet vybrané stěžejní funkce definované v analýze požadavků na aplikaci. Tedy takovou aplikaci navrhnout, implementovat a řádně otestovat.

Závěrečným cílem praktické části práce je zhodnotit a diskutovat výsledky, zejména s ohledem na přínosy pro firmu, která by se rozhodla takové řešení implementovat.

Kapitola 3
Rešerše

3.1 Existující řešení

Tato podkapitola se zabývá průzkumem a srovnáním existujících aplikací určených k usnadnění správy kanceláří a pracovních míst a srovnává jejich funkce, použitelnost a možnosti integrace. Nalezených, již existujících, řešení je velké množství a není možné do srovnání zahrnout všechny. Webové stránky, které slouží k vyhledávání aplikací a k jejich hodnocení, uvádí při vyhledání kategorií jako například „Space Management Software“ (software pro správu prostor) či „Desk Booking Software“ (software pro rezervaci stolů) ve svých výsledcích více než 70 aplikací [1, 2].

Srovnání takového množství aplikací by mohlo rozsahem vystačit na samostatnou práci, proto se tato podkapitola omezuje pouze na nejpoužívanější aplikace tohoto typu. Zároveň byly ze srovnání vyřazeny aplikace, které veřejně neuvádí informace o ceně nebo o nabízených funkcích, protože takové aplikace by pochopitelně nebylo možné řádně porovnat.

Následuje seznam a popis aplikací, které byly vybrány ke srovnání. Pokud aplikace nabízí více plánů s rozdílnými cenami, vždy platí, že dražší plán pouze rozšiřuje funkce levnějších plánů. Proto jsou ve výčtu funkcí uváděny pouze nově získané funkce oproti předešlému plánu. Přehledné porovnání vybraných aplikací na základě vybraných funkcí a roční ceny pro 200 zaměstnanců viz tabulka 3.1.

3.1.1 Deskbird

Aplikace Deskbird je jednou z nejpoužívanějších aplikací pro správu kanceláří a přiřazování míst zaměstnancům. V internetovém vyhledávání a ve článcích, které srovnávají a hodnotí webové aplikace tohoto typu, se aplikace Deskbird umisťuje na předních příčkách [1, 3].

Deskbird nabízí 3 různé, cenově odstupňované, plány:

- Starter (1,80 EUR/uživatel/měsíc)
  - mobilní a webová aplikace
  - integrace s MS Teams a Slack
  - základní rezervace míst
  - synchronizace s Outlook a GCal
- Business (3,80 EUR/uživatel/měsíc)
  - interaktivní plány kanceláří
  - analýzy pro adminy

- parkovací místa
- sdílené prostory
- obousměrná synchronizace kanceláří a míst s kalendáři
- SSO

- Enterprise (cena individuální dle konkrétních požadavků)
  - synchronizace SCIM a HRIS
  - osobní konfigurace zabezpečení dat
  - dedikovaný CSM
  - další požadavky dle domluvy [4]

3.1.2 Yarooms

Yarooms jako jedna z mála aplikací nabízí fixní měsíční částku nehledě na počet uživatelů, kanceláří či míst k sezení [5]. Finančně se tak vyplatí společnostem s větším množstvím zaměstnanců, řádově alespoň nižší stovky zaměstnanců.

Stejně jako Deskbird nabízí i Yarooms 3 různé plány na základě požadovaných funkcí:

- Starter (200 USD/měsíc)
  - rezervace míst a kanceláří
  - webová a mobilní aplikace
  - zobrazení vhodné pro recepci či manažery (blíže nespecifikováno a bez ukázky)
- Business (500 USD/měsíc)
  - interaktivní plány kanceláří
  - analýza pro adminy
  - integrace s Google kalendářem
- Enterprise (cena individuální dle konkrétních požadavků)
  - integrace s MS Teams
  - SSO
  - přístup k API
  - vlastní požadované funkce dle domluvy
  - Azure a AWS nasazení [5]

3.1.3 NSpace

NSpace je další z mnoha aplikací pro správu míst v kancelářích. Stejně jako většina ostatních i NSpace nabízí 3 rozdílné plány s rozdílnými cenami a funkcemi a to konkrétně:

- Starter (30 dní zdarma, poté nutno přejít na Growth)
  - rezervace míst a kanceláří
  - webová a mobilní aplikace
  - možnost přidávat vybavení stolu (blíže nespecifikováno)
  - dostupnost míst v reálném čase

- Growth (3 USD/místo/měsíc + 15 USD/kancelář/měsíc)
  - integrace s Outlook a MS Teams
  - možnost přidávat vybavení celých kanceláří
  - integrace adresáře uživatelů
  - odstranění zombie rezervací
  - dostupnost kanceláří v reálném čase
  - analýza a poznatky o pracovišti
  - SSO
- Enterprise (funkce a cena dle domluvy) [6]

3.1.4 Tactic

Aplikace Tactic se na jednom z nejpopulárnějších srovnávacích webů, G2, pyšní mnoha různými oceněními. Dohromady se jedná o 14 ocenění, konkrétně například za nejlepší podporu za jaro 2023 a zimu 2023, nejsnazší na implementaci za léto 2023 nebo nejjednodušší admin za léto 2022 [7]. Na svém oficiálním webu se na G2 velmi odkazují, třeba také na srovnání své aplikace Tactic s jinými podobnými aplikacemi [8]. V rámci těchto srovnání je aplikace Tactic samozřejmě hodnocena vždy lépe v naprosté většině kategorií [7].

Tactic nabízí pouze jeden plán s cenou 2,25 USD za každé pracovní místo za měsíc [9].

- Funkce pro běžné uživatele:
  - rezervace stolů nebo pracovních míst
  - webová a mobilní aplikace
  - rezervace parkovacích míst
  - plánování a koordinace týdenních hybridních rozvrhů
  - interaktivní mapa kanceláře
  - dostupnost míst v reálném čase a vyhledání, kde sedí hledaná osoba
  - vícedenní a opakované rezervace stolů
  - opuštění místa dříve, aby byl stůl opět k dispozici
  - automatická oznámení a upomínky
  - integrace se Slack a Microsoft Teams
- Funkce pro administrátory:
  - přiřazení stolů a pravidla rezervací
  - rezervování, úpravy a rušení rezervací jménem ostatních uživatelů
  - správa pater, čtvrtí, stolů a parkování
  - určení kancelářských hodin a svátků
  - vytváření a vyžadování zdravotních prohlídek
  - zobrazení údajů o využití kanceláří [9]

3.1.5 Envoy

Další z nalezených populárních aplikací, Envoy, nabízí oddělených produkty, které jsou nazvané Envoy Visitors, Envoy Connect a Envoy Workplace. Kombinace těchto služeb může poskytnout kompletní správu nejen zaměstnanců a pracovních prostorů, ale také návštěv a schůzek [10]. Pro správu pracovních prostorů je určen produkt Envoy Workplace, který nabízí opět 3 cenově a funkcemi odlišené plány:

- Standard (3 USD/uživatel/měsíc)
  - mapy pracovišť
  - plánování pracovišť
  - sledování zásilek a dodávek
  - integrace s úkoly (ticketing)
  - oznámení a upozornění
  - integrace s mobilními zařízeními, Slack, Teams, GCal a Outlook
  - mobilní přihlašování na základě polohy
- Premium (5 USD/uživatel/měsíc)
  - kontrola zdraví a bezpečnosti
  - automatické přihlašování prostřednictvím kontroly přístupu a integrace wi-fi
  - rezervace konferenčních místností
  - rezervace stolu
  - analýza obsazenosti a využití prostor
  - nastavení a měření výkonnosti v porovnání s kancelářskými zásadami
  - SSO, synchronizace adresářů
  - plánování delegovaných asistentů
  - plánování scénářů
  - rozhraní API
- Premium Plus (7 USD/uživatel/měsíc)
  - odesílání vícekanálových nouzových oznámení zaměstnancům
  - automatická správa uživatelských identit a přístupových práv [11]

Produkt	API	Integrace	Interaktivní plány kanceláří	Analýza a statistiky	SSO	Cena pro 200 osob za rok*
Deskbird starter	ne	Slack, MS Teams, Outlook, Google kalendář	ne	ne	ne	106 704 CZK
Deskbird business	ne	Slack, MS Teams, Outlook, Google kalendář	ano	ano	ano	225 264 CZK
YAROOMS starter	ne	žádná	ne	ne	ne	56 160 CZK
YAROOMS business	ne	Google kalendář, Outlook	ano	ano	ne	140 400 CZK
YAROOMS enterprise	ano	Google kalendář, MS Teams, Outlook	ano	ano	ano	individuální
NSpace growth	ne	Google kalendář, MS Teams	ano	ano	ano	168 480 CZK
Tactic	ne	Slack, MS Teams	ano	ano	ne	94 770 CZK
Envoy standard	ne	Slack, Google kalendář, MS Teams, Outlook	ano	ne	ne	168 480 CZK
Envoy premium	ano	Slack, Google kalendář, MS Teams, Outlook	ano	ano	ano	280 800 CZK

*Pro služby závisející na počtu kanceláří a pracovních míst byly dále použity hodnoty: 15 kanceláří, 150 pracovních míst. Ceny byly vypočteny nehledě na aktuálně nabízené speciální nabídky a slevy (některé aplikace nabízí slevu 5–15 % při roční platbě namísto měsíční). Pro převod do CZK byly použity kurzy 1 EUR = 24,7 CZK, 1 USD = 23,4 CZK.

Tabulka 3.1 Srovnání existujících aplikací

3.2 Technologie

V rychle se rozvíjející oblasti vývoje webových aplikací je pro vytváření efektivních, robustních a uživatelsky přívětivých webových aplikací zásadní porozumět základním technologiím. Tato část se zabývá technologiemi, které tvoří páteř moderního vývoje webových aplikací, a ukazuje jejich vlastnosti, role a způsob, jakým společně přispívají k ekosystému webu.

Každá technologie, od základního jazyka HTML, který určuje strukturu webového obsahu, až po pokročilé frameworky jako Symfony a API Platform, které zefektivňují vývoj komplexních aplikací, má svůj specifický účel a uplatnění. Kromě toho se tato část zabývá stylistickými a funkčními vylepšeními, která poskytuje CSS, možnostmi jazyka PHP na straně serveru a dynamickou interaktivitou, kterou umožňuje JavaScript a jeho populární knihovna React.js. Kromě toho se podkapitola zabývá nedílnou úlohou PostgreSQL jako systému pro správu databází, zejména při správě složitých datových struktur.

Tyto technologie společně představují komplexní sadu nástrojů pro webové vývojáře, která umožňuje vytvářet vše od jednoduchých webových stránek až po složité webové aplikace. Cílem této podkapitoly je poskytnout základní informace o těchto technologiích a připravit půdu pro jejich praktické použití ve scénářích vývoje webových stránek.

3.2.1 HTML

HTML (HyperText Markup Language) je standardní značkovací jazyk používaný k vytváření a strukturování webových stránek a jejich obsahu. Je páteří každé webové stránky a v zásadě slouží k definování struktury a obsahu webové stránky, včetně textu, obrázků a dalších prvků. Jazyk HTML používá řadu prvků uzavřených značkami, které definují různé části obsahu. Tyto prvky mohou obsahovat atributy, které poskytují další informace o chování nebo vzhledu prvku. Soubory HTML mají obvykle příponu .htm nebo .html a webové prohlížeče je interpretují a zobrazují jejich obsah [12, 13].

HTML5, nejnovější verze jazyka HTML, zavádí mnoho nových syntaktických vlastností a zlepšuje funkčnost jazyka. Obsahuje značky jako <video>, <audio>, <canvas> a <nav>, které zlepšují schopnost zpracovávat multimediální a grafický obsah a také efektivněji strukturovat webové stránky [13].

3.2.2 Twig

Twig je naproti tomu moderní šablonovací engine pro jazyk PHP. Je navržen tak, aby poskytoval flexibilní, rychlý a bezpečný způsob vytváření šablon pro webové aplikace. Šablony Twig jsou v podstatě textové soubory, které mohou generovat libovolný textový formát, například HTML, XML, CSV atd. Tyto šablony obsahují proměnné nebo výrazy, které se při vyhodnocení šablony nahradí hodnotami, a značky, které řídí logiku šablony. V systému Twig existují dva druhy oddělovačů: {% ... %} pro spouštění příkazů, jako jsou smyčky, a {{ ... }} pro výstup výsledku výrazu [14].

Stručně řečeno, zatímco HTML je značkovací jazyk používaný ke strukturování a prezentaci obsahu na webu, Twig je šablonovací engine, který dynamicky generuje HTML nebo jiné textové formáty nahrazováním proměnných a vyhodnocováním výrazů v šablonách. Twig poskytuje dynamičtější a programovatelnější přístup ke generování webového obsahu, který se často používá ve spojení s jazykem HTML k vytváření dynamicky generovaných webových stránek.

3.2.3 CSS

CSS (Cascading Style Sheets) je jazyk používaný ke stylování a rozvržení webových stránek, který řídí design, rozvržení a změny zobrazení pro různá zařízení a velikosti obrazovky. Popisuje,

jak by se měly zobrazovat prvky HTML, a umožňuje vývojářům měnit barvy, písmo, rozvržení a další prvky [15].

CSS zvyšuje efektivitu a konzistenci webového designu tím, že řídí rozvržení více webových stránek najednou, a lze jej přidávat do dokumentů HTML různými způsoby, například jako inline, interní nebo externí soubory stylů [16]. CSS je jazyk založený na pravidlech, v němž vývojáři definují pravidla určením skupin stylů, které mají být použity na konkrétní prvky nebo skupiny prvků na webové stránce [15].

CSS je základní technologií pro vytváření vizuálně přitažlivého a dobře strukturovaného webového obsahu a často se používá ve spojení s jazykem HTML a JavaScript k vytváření dynamických a interaktivních webových stránek [15].

3.2.4 PHP

PHP, což je zkratka pro „PHP: Hypertext Preprocessor“, je široce používaný skriptovací jazyk na straně serveru určený především pro vývoj webových stránek [17]. Původně jej vytvořil dánsko-kanadský programátor Rasmus Lerdorf v roce 1993 a zveřejnil jej v roce 1995 [18].

Jazyk PHP je známý svou schopností vytvářet dynamické a interaktivní webové stránky, což z něj činí výkonný nástroj pro vývoj webových stránek. Jedná se o univerzální skriptovací jazyk, který lze nasadit na většině webových serverů, operačních systémů a platforem a který lze používat s různými relačními systémy správy databází. Jazyk PHP se často používá k vytváření dynamického obsahu webových stránek, dynamických obrázků a lze jej použít i pro skriptování v příkazovém řádku a pro aplikace grafického uživatelského rozhraní (GUI) na straně klienta [19] [20].

Jazyk PHP je primárně určen pro vývoj webových stránek na straně serveru, kdy kód PHP v požadovaném souboru spouští běhové prostředí jazyka PHP a vytváří dynamický obsah webových stránek nebo dynamické obrázky používané na webových stránkách nebo jinde. Je rychlý, flexibilní a pragmatický a pohání širokou škálu aplikací, od blogů až po komplexní webové aplikace. Jazyk PHP je také známý svou rozsáhlou komunitou a zdroji, což z něj činí oblíbenou volbu pro vývoj webových stránek [20].

3.2.5 Symfony

Symfony je open-source framework PHP, který poskytuje řadu nástrojů a funkcí pro vytváření komplexních webových aplikací. Jedná se o opakovaně použitelnou sadu samostatných, oddělených a soudržných komponent jazyka PHP, které řeší běžné problémy při vývoji webových stránek [21].

Jedná se o funkčně bohatý framework, který se používá k vytváření komplexních aplikací. Poskytuje řadu nástrojů a funkcí speciálně navržených pro vytváření škálovatelných webových aplikací PHP, včetně podpory různých prostředí a robustního systému ukládání do mezipaměti [22].

Co se týče architektury, framework Symfony je možné použít na různé typy architektury, ovšem primárně je určen na strukturovanou architekturu MVC (model-view-controller), která je skvělá pro škálovatelné a systematické projekty vývoje webových stránek. Nabízí předem vytvořenou adresářovou strukturu a mnoho nezávislých komponent. Dokumentace Symfony obsahuje články, výukové programy a knihy, které umožňují seznámit se s celým frameworkem Symfony PHP [21, 22].

Symfony nabízí flexibilitu a přizpůsobení, takže je skvělou volbou pro vytváření složitých webových aplikací PHP, které vyžadují vysokou úroveň přizpůsobení. Je navržen tak, aby optimalizoval vývoj webových aplikací, a s každou další verzí se rozrůstá o další funkce [22].

3.2.6 API Platform

API Platform je moderní webový framework navržený tak, aby usnadňoval vytváření projektů založených na rozhraní API bez omezení rozšiřitelnosti a flexibility. Poskytuje stabilní, moderní rozhraní REST API a GraphQL API, automatickou dokumentaci pomocí OpenAPI, SwaggerUI a GraphQL, a podporu různých standardů a formátů. Framework je navržen tak, aby byl použitelný od kontextu rychlého vývoje aplikací až po přístupy typu Domain-Driven Design (zaměření na vývoj doménového modelu) nebo Clean Architecture (důraz na oddělení zájmů rozdělením systému do vrstev), a v případě potřeby umožňuje vývojářům kombinovat oba přístupy [23].

Framework API Platform je známý svou schopností zjednodušit tvorbu a správu rozhraní API a poskytuje jednotné prostředí pro návrh, tvorbu, správu a použití rozhraní API v průběhu celého životního cyklu rozhraní API. Umožňuje vývojovým týmům efektivně vytvářet, spravovat, publikovat a konzumovat rozhraní API, což z něj činí cenný nástroj pro organizace, které chtějí realizovat strategii „API-first“ (nejprve se vytváří API, které slouží jako základ, na němž se vyvíjí zbytek aplikace) a využít plný potenciál rozhraní API [23, 24].

3.2.7 JavaScript

JavaScript je všestranný programovací jazyk, který je pro moderní web nezbytný. Odpovídá standardu ECMAScript a je známý svými prvotřídními funkcemi, které umožňují jeho interpretaci nebo kompilaci v pravý čas. Jeho všestrannost podporuje různé programovací styly, takže je ideální pro vytváření dynamických, interaktivních a multimediálních webových stránek. JavaScript se neomezuje pouze na webové prohlížeče, ale funguje i v jiných než prohlížečových prostředích, jako je Node.js, Apache CouchDB a Adobe Acrobat [25].

Při vývoji webových stránek hraje JavaScript klíčovou roli při implementaci složitých funkcí na webových stránkách. Oživuje webové stránky nad rámec statického zobrazení a umožňuje aktualizovat obsah, vytvářet interaktivní mapy, animovanou grafiku a další funkce. Tato schopnost manipulace s jazyky HTML a CSS spolu s výpočtem, manipulací a validací dat z něj činí mocný nástroj pro interaktivitu webu [26].

Vliv JavaScriptu je široce rozšířen po celém internetu. Od roku 2023 využívá 98,8 % webových stránek pro chování webových stránek na straně klienta jazyk JavaScript, který často zahrnuje knihovny třetích stran. Toto široké využití podtrhuje jeho význam jako jedné ze základních technologií World Wide Webu, vedle HTML a CSS [27].

JavaScript, který vymyslel Brendan Eich, je nejen výkonný, ale také všestranný a vhodný pro začátečníky. S přibývajícími zkušenostmi mohou vývojáři pomocí JavaScriptu vytvářet hry, animovanou 2D a 3D grafiku a komplexní aplikace založené na databázích, což podtrhuje jeho široké možnosti využití [28].

3.2.8 React.js

React.js, běžně známý jako React, je bezplatná knihovna JavaScriptu s otevřeným zdrojovým kódem, která se používá především k vytváření uživatelských rozhraní. Byla vyvinuta ve společnosti Facebook a vydána v roce 2013, React spravuje společnost Meta (dříve Facebook) spolu s různorodou komunitou vývojářů. Její hlavní vlastností je používání opakovaně použitelných komponent uživatelského rozhraní, což zjednodušuje vývoj jednoduchých jednostránkových aplikací i složitých webových rozhraní. React není framework, ale cílená knihovna, která se používá spolu s dalšími knihovnami, jako je ReactDOM, pro vývoj webových aplikací a je rozšířena na vývoj mobilních aplikací pomocí React Native. Díky komponentové architektuře se React stal základním kamenem moderního vývoje webových aplikací a používá se v populárních aplikacích, jako je Facebook a Instagram [29, 30].

3.2.9 PostgreSQL

PostgreSQL, často označovaný jako Postgres, je významný open-source objektově-relační systém pro správu databází. Je uznáván pro své pokročilé schopnosti, je to databázový systém podnikové třídy, který je vysoce rozšiřitelný. PostgreSQL je známý svými robustními funkcemi, spolehlivostí, vysokým výkonem a škálovatelností, díky čemuž je vhodný pro zpracování složitých a rozsáhlých datových zátěží [31].

Jedním z klíčových aspektů PostgreSQL je podpora dotazů SQL a JSON, což mu umožňuje spravovat širokou škálu datových typů a struktur. Tato univerzálnost umožňuje jeho použití v řadě aplikací a prostředí. Systém využívá a rozšiřuje jazyk SQL a obsahuje mnoho funkcí, které zajišťují bezpečné ukládání a škálování složitých datových úloh [31, 32].

Databázový systém je založen na projektu POSTGRES, který se zasloužil o průkopnictví mnoha konceptů, jež jsou dnes v databázové technologii běžné. Tento základ přispěl k dobré pověsti PostgreSQL ve světě správy databází, kde je často volen pro své komplexní a spolehlivé možnosti zpracování dat [31].

Kapitola 4
Analýza

4.1 Analýza řešení

4.1.1 Zhodnocení existujících řešení

Každá z nalezených aplikací má své výhody a nevýhody a nabízí různé funkce. Některé aplikace zároveň neslouží pouze ke správě míst v kancelářích, ale nabízí také mnoho dalších funkcí, které by firma mohla ocenit. Je tak možné, že aplikace pro správu míst v kancelářích vyřeší zároveň i problémy s návštěvami, schůzkami s klienty nebo problém s parkovacími místy.

Ovšem, pokud firma hledá pouze aplikaci pro správu míst a další funkce nepotřebuje, protože s dalšími podobnými problémy se nepotýká, pak budou tyto funkce navíc způsobovat nepřehlednost aplikace, zmatení a používání aplikace budou komplikovat. Záleží tedy vždy na požadavcích konkrétní firmy a nárocích, které na aplikaci bude klást.

4.1.1.1 Nabízené funkce

Téměř všechny aplikace nabízí očekávané hlavní funkce, kterými jsou: přiřazování zaměstnanců na pracovní místa pro konkrétní dny a časy včetně opakujících se vzorů (pattern), interaktivní plány kanceláří nebo vyhledání osoby a zobrazení jejího aktuálního pracovního místa. V tomto ohledu tedy téměř všechny aplikace vyhovují přirozeným požadavkům na aplikaci na správu pracovních míst v kancelářích.

Mnohé aplikace zároveň neslouží pouze ke správě míst v kancelářích, ale nabízí také plánování alokací, evidenci návštěv, týmových schůzek, schůzek s klienty, správu a rezervaci parkovacích míst, zaznamenávání zdravotních prohlídek či bezpečnostních školení a další potenciálně užitečné související funkce.

Pokud tedy jde o nabízené funkce, většina aplikací splňuje určité minimální nároky na aplikaci pro správu pracovních míst. Neideální volbou by z tohoto hlediska mohly být pouze plány Deskbird starter a YAROOMS starter, které nenabízí interaktivní plány kanceláří. Zároveň neumožňují ani zobrazení statistik či analytických dat souvisejících s vytížením kanceláří a pracovních míst.

4.1.1.2 Integrace

Většina aplikací nabízí integraci do vybraných kalendářů a služeb, kterými jsou Slack, MS Teams, Outlook a Google kalendář. Zde tedy bude opět velmi záležet na konkrétních požadavcích firmy a na službách, které používá.

Pokud jde o konkrétní podobu integrace, žádná aplikace bohužel veřejně nerozvádí do podrobností, jak integrace vypadá a co všechno nabízí. Je tedy otázkou, zda například integrace do Slack bude umožňovat mimo očekávaného zobrazení pracovních míst a přiřazených osob také třeba vyhledání osoby a jejího aktuálního pracovního místa či zda bude dokonce možné ze Slack vytvářet nová přiřazení zaměstnanců na pracovní místa apod. V případě těchto potřeb by bylo tedy třeba aplikace zakoupit na vyzkoušení či využít časově omezenou demo verzi, kterou mnoho z aplikací nabízí, a funkčnost a použitelnost integrace si přímo vyzkoušet.

4.1.1.3 API

Je neuspokojivým zjištěním, že naprostá většina aplikací nenabízí rozhraní API pro možnou integraci mezi ostatní firemní aplikace. Pokud by tedy firma používala zároveň i jiné aplikace na vedení údajů o zaměstnancích, zaznamenávání dovolených nebo plánování alokací, pak by bylo nutné při každé změně údaje ručně upravovat. Neexistence rozhraní API tedy znesnadňuje používání aplikace spolu s jinými firemními aplikacemi a vytváří tak tlak na povýšení (upgrade) na dražší cenový plán a na kompletní přechod na dané řešení aplikace.

4.1.1.4 Nejvhodnější aplikace

Jak vyplývá z předešlých odstavců, většina aplikací obstála z hlediska nabízených funkcí a přímé integrace do Slack, MS Teams, Outlook nebo Google kalendáře, nicméně webové rozhraní API nabízí pouze dvě z nalezených aplikací a jejich plánů. Konkrétně se jedná o YAROOMS enterprise a Envoy premium. Roční cena za Envoy premium pro 200 osob je 280 800 CZK viz tabulka 3.1. Cena za YAROOMS enterprise není uváděna, nicméně vzhledem k cenám levnějších plánů YAROOMS starter a YAROOMS business (viz tabulka 3.1) lze předpokládat, že se bude pohybovat nad 200 000 CZK za rok.

4.1.2 Vývoj vlastního řešení

Využití již existujícího řešení je snadnou a rychlou volbou. Pokud firma potřebuje řešit problém s pracovními místy okamžitě nebo pokud nechce riskovat, že se vývoj vlastního řešení nepodaří nebo se neplánovaně prodraží, pak je určitě využití již existujícího řešení vhodnou volbou. Nicméně mnoho manažerů a ředitelů firem se snaží přemýšlet v dlouhodobém horizontu deseti, nebo lépe dvaceti či třiceti, následujících let a v tomto horizontu bude částka za využití existujících řešení šplhat do milionů korun. Vyvinout vlastní řešení se tedy zdá být rozumným krokem v momentě, kdy řešení nespěchá a je možné vývoj vlastního řešení realizovat za nižší miliony korun.

Vzhledem k tomu, že tato aplikace vzniká v rámci bakalářské práce, vývoj je bezplatný a není realizovaný pro konkrétní firmu, úvaha o případné finanční výhodnosti tohoto řešení je samozřejmě bezpředmětná.

4.2 Procesní analýza

Tato podkapitola se věnuje důkladné analýze podnikových (business) procesů, pro zjištění a popsání rozdílů mezi současným (AS-IS) a budoucím (TO-BE) stavem po implementaci webové aplikace pro správu a rezervaci kancelářských míst.

Vzhledem k tomu, že aplikace není vyvíjena pro potřeby konkrétní firmy, současný stav nepopisuje aktuální stav procesů určité společnosti, ale popisuje současný stav procesů imaginární společnosti, která se potýká s problémy spojenými s nedostatkem míst v kancelářích a s jejich špatnou správou. Podoba tohoto popsaného současného stavu však pochopitelně není nahodilá

a odráží jak mé osobní zkušenosti z pracovního prostředí, tak i zkušenosti mých kolegů a spolužáků, především z oblasti firem věnujících se vývoji softwaru. Mnohé firmy se tedy nepochybně v popsaných AS-IS stavech naleznou.

Tato analýza poskytuje komplexní pohled na předpokládaná zlepšení a přínosy navrhovaného systému. Podkapitola popisuje jak globální pohled na současný a navržený budoucí stav, tak i současný a budoucí stav konkrétních podnikových procesů, které budou implementací aplikace nejvíce ovlivněny.

4.2.1 Globální pohled

Pro snadnější pochopení rozdílu mezi současným (AS-IS) a budoucím (TO-BE) stavem využití webové aplikace pro správu a rezervaci míst v kancelářích následuje globální pohled na tyto stavy, který ve stručnosti a obecnosti předesílá první výhody navrženého budoucího stavu.

4.2.1.1 AS-IS

- Neformální domluva: V současném stavu je přidělování pracovních míst založeno na neformální domluvě. Zaměstnanci si vybírají místa podle dostupnosti nebo zvyku, aniž by byla oficiální rezervace nebo systém sledování.
- Konflikty a neefektivita: Může docházet ke konfliktům o oblíbená místa, nejistotě ohledně dostupnosti a neefektivnímu využívání pracovních prostorů.
- Absence dat: Firma nemá přehled o využití pracovních míst, což komplikuje plánování a optimalizaci prostor.
- Závislost na osobní komunikaci: Rezervace a domluvy probíhají osobně nebo elektronickou komunikací, což může být časově náročné a náchylné k chybám.

4.2.1.2 TO-BE

- Centralizovaný rezervační systém: Aplikace poskytuje centralizovanou platformu pro rezervaci pracovních míst. Zaměstnanci si mohou rezervovat místa pro konkrétní dny a časy.
- Přehled o obsazenosti: Aplikace poskytuje jasný přehled o tom, která místa jsou kdy obsazena, což eliminuje konflikty a zvyšuje efektivitu využívání prostor.
- Plánování a analýza: Firma má možnost analyzovat data o využití pracovních míst pro lepší plánování a optimalizaci kancelářských prostor.
- Flexibilní přiřazování: Systém umožňuje jednorázové nebo opakované přiřazení míst, což zvyšuje flexibilitu a usnadňuje dlouhodobé plánování.
- Zefektivnění procesů: Automatizace procesu rezervace a správy místa snižuje administrativní zátěž a zvyšuje produktivitu zaměstnanců.

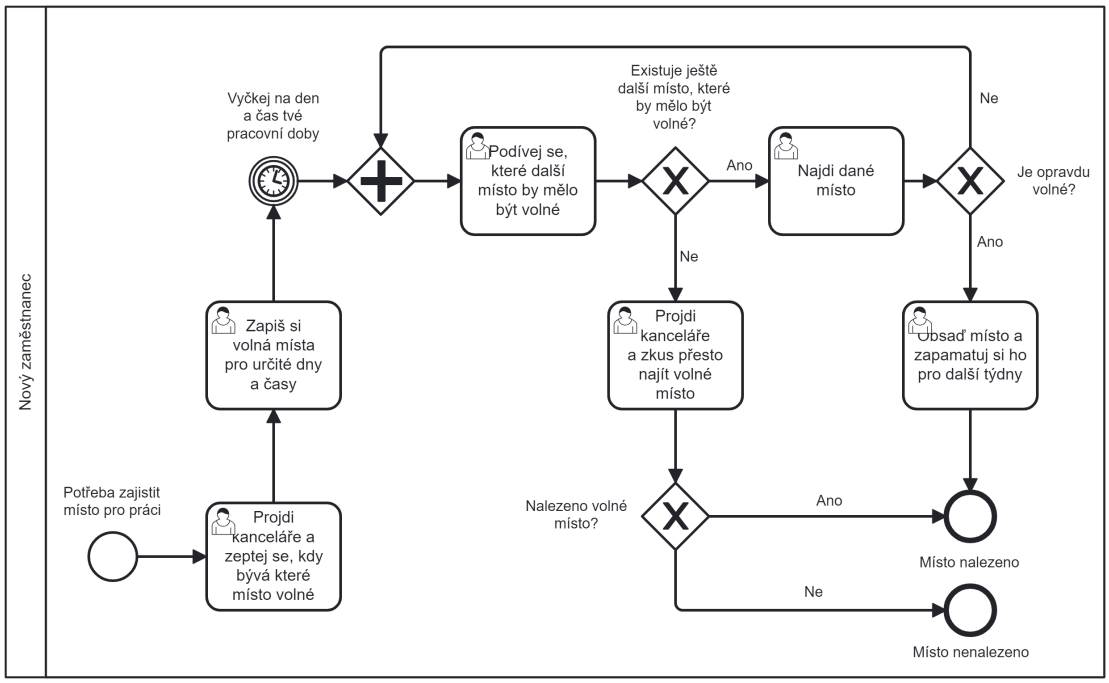
4.2.2 Přiřazení pravidelného místa novému zaměstnanci

4.2.2.1 AS-IS

V současném stavu je proces vyhledávání pracovního místa ve firmě pro nového zaměstnance především manuální a závislý na osobní komunikaci. Když nový zaměstnanec nastoupí do firmy, je okamžitě postaven před úkol najít volné pracovní místo. Tento proces obvykle zahrnuje fyzickou procházku zaměstnance po kanceláři, vizuální kontrolu různých místností a pracovních prostor s cílem najít stůl, který se zdá být volný.

Během tohoto hledání se musí nový zaměstnanec typicky ptát stávajících zaměstnanců, aby získal informace o dostupnosti pracovních míst. Tento přístup do značné míry závisí na znalostech a aktuální přítomnosti stávajících zaměstnanců, které se mohou lišit, což vede k možným nepřesnostem nebo spoléhání se na zastaralé informace. Tento přístup postrádá formalizovaný postup, takže proces je nestandardizovaný a náchylný k chybám, tedy potenciálním konfliktům, kdy dva zaměstnanci počítají s tím, že si zaberou stejné místo.

Celkově se současný proces vyznačuje neefektivitou a značnou časovou náročností při osobních domluvách, což vede ke ztrátě produktivní pracovní doby jak pro nového zaměstnance, tak pro stávající pracovníky, Navíc může tato absence formalizovaného a efektivně řešeného procesu vést k frustraci a ke špatné počáteční zkušenosti nových zaměstnanců na pracovišti.



Obrázek 4.1 Přiřazení pravidelného místa novému zaměstnanci AS-IS (BPMN diagram)

4.2.2.2 TO-BE

Po zavedení webové aplikace pro správu pracovních míst se proces přidělování pracovního místa novým zaměstnancům výrazně zjednoduší a zefektivní. Aplikace bude poskytovat funkce pro jednorázové i opakované přidělení místa a nabízet přehledný a organizovaný přístup ke správě pracovních prostor.

Když do společnosti nastoupí nový zaměstnanec, může v aplikaci prohlédnout dostupná místa pro zvolené datum a časový interval. Tato funkce umožňuje zaměstnanci rychle a snadno identifikovat volná místa bez nutnosti fyzického procházení napříč kancelářemi a spoléhání se na dostupnost informací od ostatních.

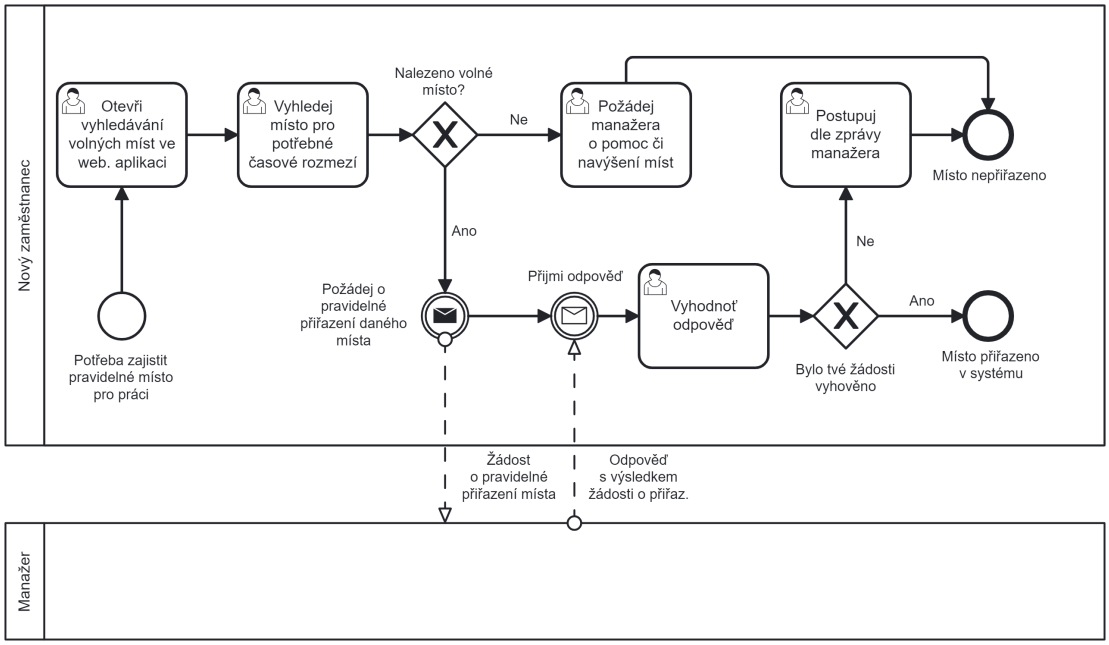
Proces je navržen tak, aby byl co možná nejvíce intuitivní. Zaměstnanec si vybere požadované pracovní místo prostřednictvím aplikace, kde je možné si zobrazit volná místa pro vybrané datum a čas. Pokud například zaměstnanec plánuje pracovat v kanceláři každé pondělí, vyfiltruje si, která místa budou volná pro následujících několik pondělků. Tato kontrola dostupnosti v reálném čase odstraňuje dohady a nejistotu přítomnou v manuálním procesu.

Jakmile zaměstnanec identifikuje vhodné místo, požádá pověřenou osobu o jeho přidělení. Tato žádost je směrována na manažera nebo správce, který má pravomoc přiřazovat pracovní

místa ve webové aplikaci. Aplikace umožňuje jednorázová i opakovaná přiřazení, čímž poskytuje flexibilitu a vychází vstříc různým pracovním režimům. Po potvrzení manažerem a vytvoření požadovaného přiřazení je pak toto místo pro zaměstnance na určenou dobu rezervováno a je pro něj tedy garantované pracovní místo.

Aplikace také vede záznamy o všech přiřazeních, čímž zajišťuje transparentnost a zabraňuje dvojím rezervacím nebo konfliktům. Tento digitalizovaný přístup nejen zvyšuje efektivitu procesu přidělování pracovních míst, ale také výrazně zlepšuje zkušenosti nových zaměstnanců. Ti mohou začít svou cestu ve firmě hladkým a bezproblémovým přidělením pracovního prostoru, což podporuje pozitivní počáteční dojem z organizačního prostředí.

Souhrnně lze říci, že implementace této webové aplikace mění proces přidělování pracovních prostor v efektivnější, spolehlivější a uživatelsky přívětivější systém, z něhož těží jak noví zaměstnanci, tak administrativní pracovníci spravující kancelářské zdroje.



Obrázek 4.2 Přiřazení pravidelného místa novému zaměstnanci TO-BE (BPMN diagram)

4.2.3 Konzultace problému s kolegou

4.2.3.1 AS-IS

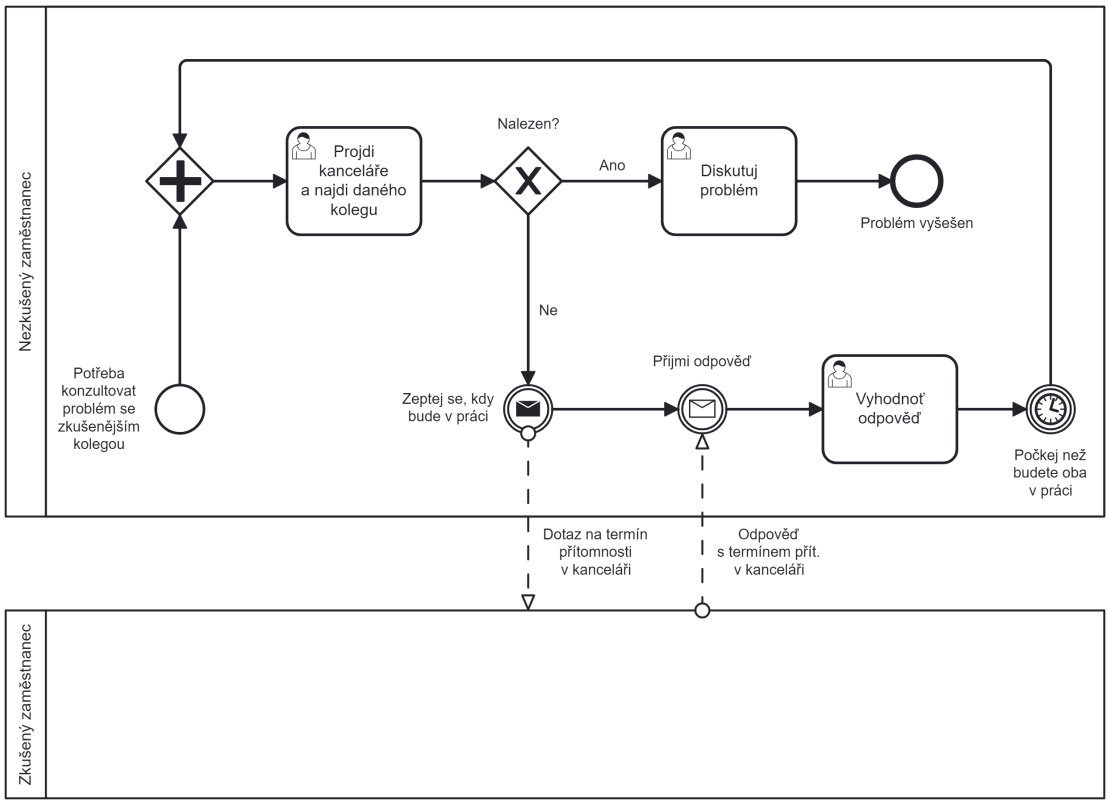
Za současného stavu je postup zaměstnance, který chce konzultovat s kolegou pracovní záležitost, poměrně těžkopádný a časově náročný. Když zaměstnanec potřebuje prodiskutovat problém nebo navázat spolupráci s kolegou, musí ho nejdříve fyzicky vyhledat v rámci kanceláře. Tento postup zahrnuje procházení různých kanceláří a pracovních prostor v naději, že se mu podaří najít konkrétní osobu, se kterou potřebuje mluvit.

Případně se zaměstnanec může rozhodnout poslat napřed zprávu nebo se zeptat ostatních kolegů, zda nevědí, kde se dotyčná osoba nachází. Tyto možnosti však nejsou vždy spolehlivé. Odeslání zprávy závisí na předpokladu, že ji kolega okamžitě uvidí a odpoví na ni, což nemusí být vždy možné, zejména pokud je na schůzce nebo mimo své pracovní místo. Dotazování se u ostatních zaměstnanců zase závisí na šanci, že někdo zná aktuální místo, kde se kolega nachází, což není nijak zaručeno.

Pokud se zaměstnanci nepodaří kolegu najít ani po fyzické prohlídce kanceláře, obvykle to znamená, že kolega není ten den přítomen v práci. Toto zjištění přichází po potenciálně zdlouhavém a bezvýsledném hledání, což vede ke ztrátě vynaloženého času a úsilí. Zaměstnanci pak nezbývá než nepřítomnému kolegovi poslat zprávu s dotazem na jeho další pracovní dostupnost nebo na to, kdy a kde se mohou sejít, aby problém prodiskutovali.

Tento proces se vyznačuje několika neefektivnostmi. Absence systematického přístupu k vyhledávání kolegů v rámci kanceláře vede ke zbytečnému času strávenému hledáním, který by mohl být využit k produktivnějším úkolům. K neefektivitě procesu navíc přispívá nejistota, že před zahájením hledání nevíme, zda je kolega v kanceláři přítomen, či nikoli. Spoléhání se na fyzické vyhledávání nebo neformální komunikační kanály vede ke zpoždění a potenciální frustraci, což brání hladkému průběhu práce a spolupráce v organizaci.

Souhrnně lze říci, že současný proces konzultace problému s kolegou je neefektivní, časově náročný a často vede ke ztrátě produktivity v důsledku spoléhání se na manuální metody vyhledávání osob v rámci kancelářských prostor.



Obrázek 4.3 Konzultace problému AS-IS (BPMN diagram)

4.2.3.2 TO-BE

V předpokládaném stavu TO-BE se zavedením webové aplikace pro správu pracovních míst se proces konzultace s kolegou ohledně pracovního problému výrazně zefektivní a zjednoduší. Aplikace poskytuje řešení pro zaměstnance, kteří chtějí se svými kolegy spolupracovat nebo s nimi diskutovat o problémech, které se jim naskytly.

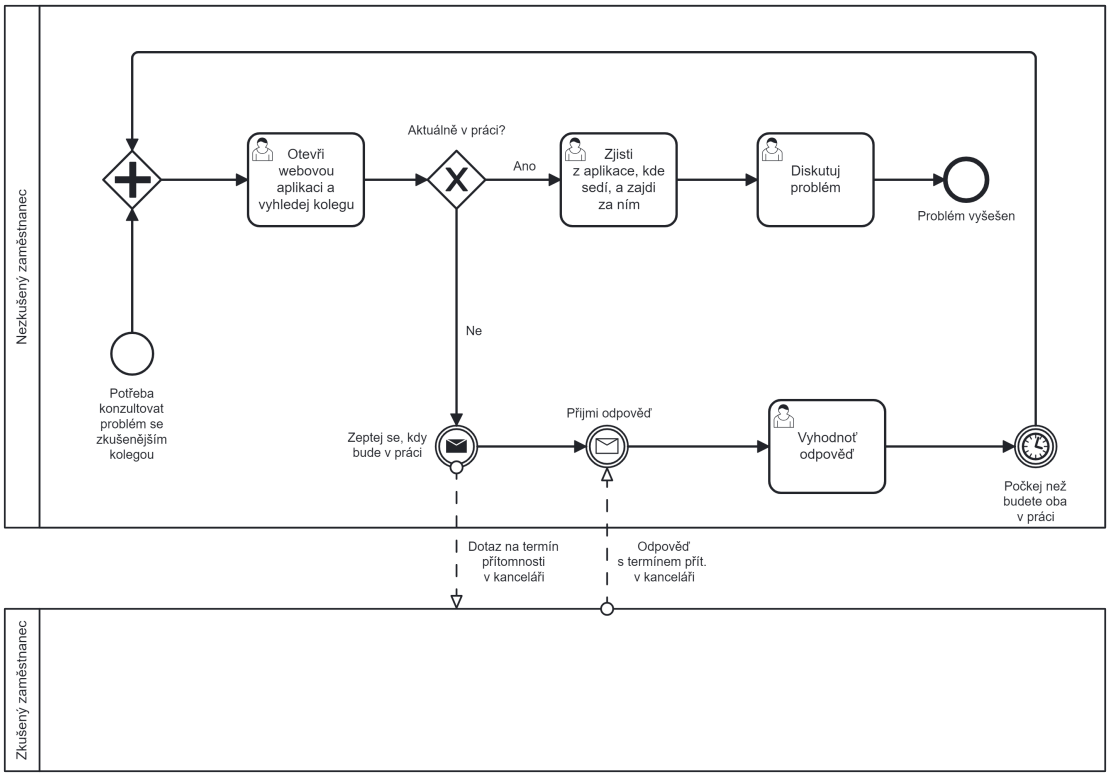
Když se zaměstnanec potřebuje poradit s kolegou, může jednoduše otevřít webovou aplikaci a zjistit, zda je kolega přítomen v práci, a co je důležité, jeho přesnou polohu v rámci kancelářských pracovišť. Aplikace zobrazí aktuální zasedací pořádek včetně informací o tom, které stoly

jsou právě obsazené a kým. Díky této funkci může zaměstnanec okamžitě zjistit, kde se jeho kolega nachází, aniž by musel fyzicky prohledávat kancelář nebo se spoléhat na to, že mu tyto informace poskytnou ostatní.

Kromě toho aplikace obsahuje podrobnosti o době, po kterou má být kolega u svého stolu. Tyto informace umožňují zaměstnanci oslovit kolegu ve vhodnou dobu, čímž je zajištěno, že konzultace proběhne včas a nezastihne kolegu jen pár minut před koncem jeho pracovní doby.

Pokud zaměstnanec zjistí, že kolega není v kanceláři přítomen, aplikace odstraňuje nutnost zbytečného hledání. Jak bylo shodně zaznamenáno i v případě AS-IS stavu tohoto procesu, zaměstnanec poté v ideálním případě zašle kolegovi zprávu s dotazem na jeho další dostupnost. Proces zde značně zlepšuje také to, že když má být následně po domluvě kolega v práci, může zaměstnanec znovu zkontrolovat aplikaci a potvrdit si jeho přítomnost a polohu. Tato funkce je užitečná zejména ve scénářích, kdy může dojít ke změnám nebo výměně místa na poslední chvíli, a zajišťuje, že zaměstnanec je vždy informován o aktuální situaci.

Implementace webové aplikace pro správu pracovních míst odstraňuje neefektivitu manuálního vyhledávání. Zajišťuje, že zaměstnanci mohou rychle a efektivně vyhledávat své kolegy, což vede k produktivnějšímu a spolupracujícímu pracovnímu prostředí. Tento systém nejen šetří čas, ale také snižuje znechucení spojené s tradičním způsobem vyhledávání kolegů, čímž zefektivňuje komunikaci a spolupráci v rámci organizace.



Obrázek 4.4 Konzultace problému TO-BE (BPMN diagram)

4.2.4 Možnost práce mimo běžnou pracovní dobu

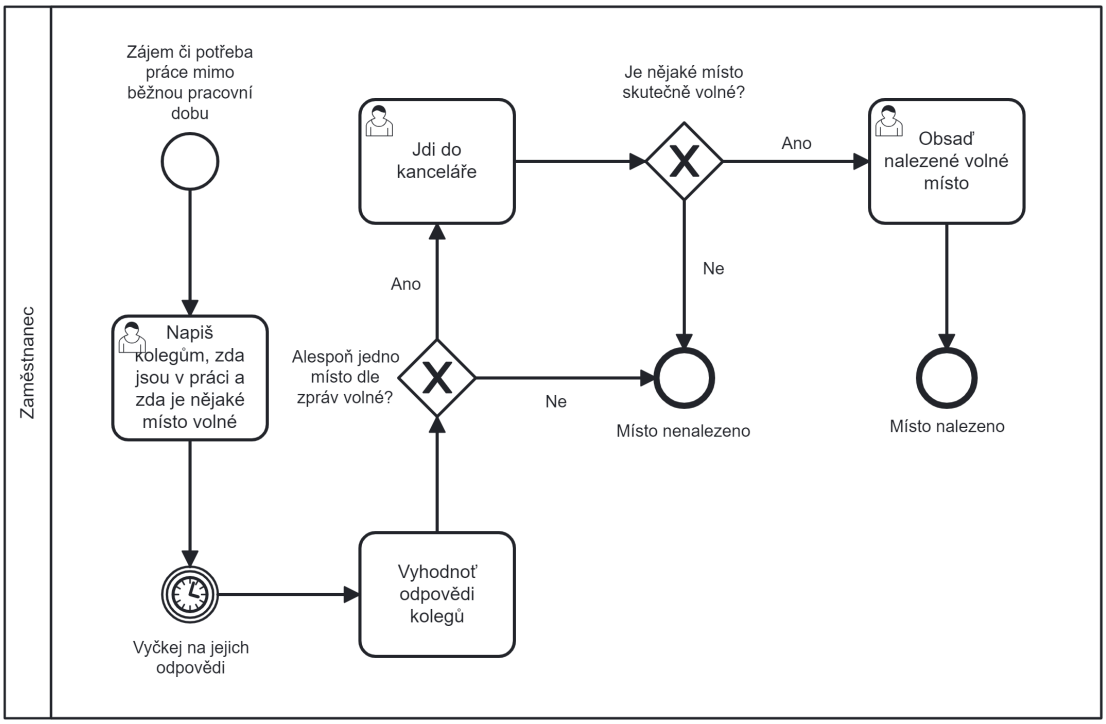
4.2.4.1 AS-IS

V současném stavu AS-IS, bez podpory webové aplikace pro správu pracovních míst, čelí zaměstnanci, kteří chtějí pracovat mimo běžnou pracovní dobu, poměrně nejistému a neefektivnímu procesu při zajišťování pracovního místa. Pokud se zaměstnanec rozhodne pracovat mimo pracovní dobu, primární možnost je zkusit se fyzicky dostavit do kanceláře a hledat volné místo. Tento způsob je do značné míry založen na náhodě, protože zaměstnanec nemá možnost předem zjistit, která místa jsou obsazená a která volná.

Nejistotu v tomto procesu způsobuje absence systému pro rezervaci nebo kontrolu dostupnosti pracovních míst pro nestandardní pracovní dobu. V důsledku toho se zaměstnanec, který najde zdánlivě volné místo, musí potýkat s možností, že může být obsazeno jiným kolegou, který se dostaví později a bude si jej nárokovat s tím, že na něm sedává pravidelně. To pochopitelně vede k potenciálním konfliktům a možnému narušení dobrých vztahů kolegů.

Další strategií, kterou mohou zaměstnanci využít, je zasílání textových zpráv svým kolegům s dotazem na aktuální volná místa v kanceláři. Tato metoda však není spolehlivá, protože informace mohou rychle zastarat. Než zaměstnanec, který například chce pracovat v den, kdy běžně nepracuje, dorazí do kanceláře, může být místo, o kterém mu bylo řečeno, že je volné, již obsazeno někým jiným. To vede k nespokojenosti a plýtvání cenným časem, který mohl být věnován práci.

Souhrnně lze říci, že absence aplikace pro správu pracovních míst v tomto procesu má za následek nedostatek informací o dostupnosti pracovních míst v reálném čase, což vede k neefektivitě a potenciálním konfliktům. Zaměstnanci pracující mimo běžnou pracovní dobu se musí spoléhat na dohady a potenciálně zastaralé informace, což činí proces hledání vhodného pracovního místa těžkopádným a nejistým.



Obrázek 4.5 Možnost práce mimo běžnou pracovní dobu AS-IS (BPMN diagram)

4.2.4.2 TO-BE

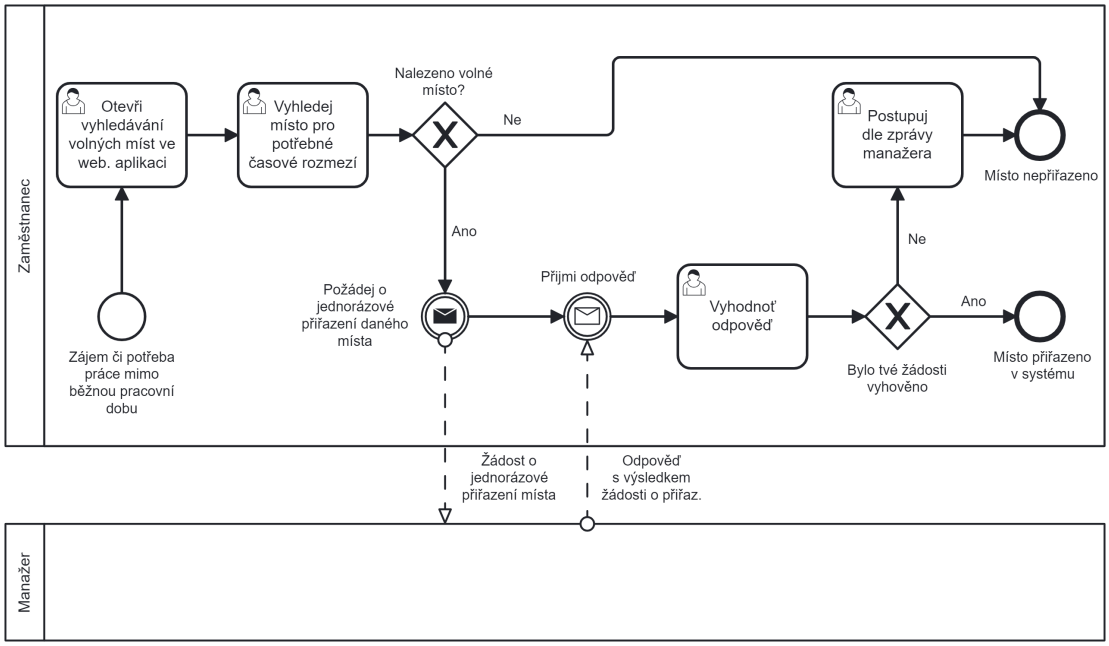
Ve stavu TO-BE, usnadněném zavedením webové aplikace pro správu pracovních míst, mají zaměstnanci, kteří chtějí pracovat mimo běžnou pracovní dobu, k dispozici mnohem efektivnější a jednodušší způsob zajištění pracovního místa. Aplikace zavádí uživatelsky přívětivé řešení, které výrazně snižuje nejistotu a potenciální konflikty spojené s přidělováním pracovního místa.

Když se zaměstnanec rozhodne pracovat mimo svou běžnou a pravidelnou pracovní dobu, může bez obtíží přejít do webové aplikace a vyhledat volné místo. Aplikace obsahuje stránku určenou k vyhledávání volných míst, kde může zaměstnanec zadat požadované datum a časový rozsah. Pokud si například zaměstnanec přeje pracovat zítra od 19:00 do 22:00 hodin, může tento časový interval zadat do aplikace. Aplikace poté zobrazí všechna místa, která budou v celém zadaném časovém rozmezí volná.

Tato funkce odstraňuje dohady a fyzické prohledávání kanceláří, které bylo dříve nutné. Snižuje také pravděpodobnost konfliktů při obsazování pracovních míst, protože aplikace poskytuje aktualizované a přesné informace o tom, která místa jsou volná a na jak dlouho.

Jakmile si zaměstnanec vybere požadované místo, osloví pověřenou osobu (typicky svého vedoucího s administrátorskými právy) prostřednictvím zprávy a sdělí mu svůj záměr pracovat u konkrétního stolu po stanovenou dobu. Pověřená osoba pak požadavek schválí (pokud neexistuje nějaký důvod jej zamítnout) a vytvoří přiřazení ve webové aplikaci. Zaměstnanec je poté informován, že je pro něj místo na stanovený interval rezervováno, a má tak jistotu, že pracovní místo bude při jeho příchodu k dispozici.

Tento vylepšený proces zajišťuje, že zaměstnanci mohou s jistotou efektivně plánovat svou práci i mimo běžnou pracovní dobu. Aplikace nejen zjednodušuje proces hledání volného místa, ale také eliminuje případné spory o využití pracovního prostoru. Umožňuje zaměstnancům soustředit se na práci s vědomím, že na ně v kanceláři čeká zaručeně volné pracovní místo bez jakýchkoli sporů a problémů.



Obrázek 4.6 Možnost práce mimo běžnou pracovní dobu TO-BE (BPMN diagram)

4.3 Analýza požadavků na aplikaci

Tato podkapitola popisuje specifické požadavky na webovou aplikaci určenou ke správě pracovních míst a přiřazování zaměstnanců na jednotlivá pracovní místa. Tyto požadavky reflektují nabízené a opomíjené funkce existujících řešení, které byly analyzovány v podkapitole 4.1, a také jsou voleny s ohledem na navržený budoucí stav procesů, které má webová aplikace optimalizovat viz podkapitola 4.2.

V rámci analýzy a definování požadavků se lze často setkat s rozdělením na funkční a nefunkční požadavky. Vzhledem k tomu, že v rámci této práce není vyvíjena webové aplikace přímo na míru konkrétní firmě, nelze specifikovat nefunkční požadavky na výkon ani škálovatelnost, protože ty závisí především na velikosti firmy (počtu míst a zaměstnanců). Pokud jde o udržitelnost, spolehlivost a rozšiřitelnost aplikace (další z častých nefunkčních požadavků), v rámci vývoje byla snaha dosáhnout co možná nejlepších výsledků prostřednictvím oddělení zájmů (viz kapitoly 5 a 6), použití nástrojů pro kontrolu kvality kódu (viz kapitola 6) a důkladného testování (viz kapitola 7).

V následujících podkapitolách jsou tedy rozebrány funkční požadavky na aplikaci (dále jen požadavky). Některé požadavky jsou pro aplikaci zcela zásadní, jiné jsou sekundární a pouze zlepšují použitelnost aplikace. Sekundární požadavky (či jejich sekundární části) jsou v textu vždy zmíněny frází, která evokuje, že se nejedná o zásadní požadavek, např. „v ideálním případě…“.

4.3.1 Přiřazování zaměstnanců na pracovní místa

Bezesporu hlavním a nejdůležitějším požadavkem je umožnit samotné přiřazování osob (zaměstnanců) k pracovním místům. K dosažení této možnosti je pochopitelně třeba též umožnit ve webové aplikaci vytvářet, upravovat a odstraňovat pracovní místa a osoby. Aplikace také nabízí vytváření, úpravy a odstranění kanceláří, ke kterým jsou pracovní místa přiřazena, aby byla webová aplikace použitelná pro firmy s mnoha kancelářemi.

4.3.1.1 Uživatelské rozhraní

Přiřazování zaměstnanců k pracovním místům probíhá v administrátorském rozhraní, do nějž mají přístup pouze přihlášení uživatelé s příslušnými administrátorskými právy. V tomto rozhraní je možné provádět veškeré zmíněné CRUD operace na pracovních místech, osobách a kancelářích. Dále má administrátor možnost vytvořit jednorázové či opakované přiřazení zaměstnance k pracovnímu místu pro zvolený den a zvolené časové rozpětí. Pro časové rozpětí postačuje volba hodin a minut, není žádoucí přidávat možnost volby sekund.

4.3.1.2 Validace

Před uložením nového jednorázového či opakovaného přiřazení webová aplikace validuje, zda je toto přiřazení uskutečnitelné a nezpůsobuje kolizi – tedy dochází ke kontrole, zda již neexistuje jednorázové či opakované přiřazení pro dané místo či danou osobu, které by zasahovalo do časového rozpětí nově vytvářeného přiřazení. Pokud dojde ke kolizi, aplikace nové přiřazení nevytvoří a vypíše chybovou hlášku, která bude obsahovat informace o kolidujících přiřazeních.

Výjimkou při validaci nového přiřazení je situace, kdy jedno přiřazení končí ve stejnou minutu, ve kterou jiné začíná, například jedno přiřazení je od 14.00 do 16.00 a druhé od 16:00 do 18:00. Tato dvě přiřazení se sice z matematického pohledu překrývají, ale intuitivně je běžné je vnímat tak, že první přiřazení bude končit v 15:59:59 a druhé bude začínat v 16:00:00. Je tedy žádoucí takováto přiřazení vyhodnotit jako nekolidující.

4.3.2 Vizuální zachycení obsazenosti kanceláří

Pro snadné a přehledné zobrazení kanceláří je nutné vizuálně zobrazit kanceláře s pracovními místy, která jsou v kanceláří umístěna pomocí souřadnicového systému. Místa jsou viditelně rozlišena tak, aby bylo na první pohled jasné, které místo je aktuálně obsazené a které je volné. Ideálně je rozlišení realizováno pomocí rozdílné barvy obsazené a volné židle.

Po přiložení myši na pracovní místo se zobrazí v nápovědě (tooltip) dodatečné informace obsahující jméno osoby, která na daném místě sedí a časové rozpětí, během kterého má toto místo přiřazeno. Pro podporu hladkého fungování i na dotykových zařízeních bude možné tuto dodatečnou informaci zobrazit také kliknutím na vybrané pracovní místo.

4.3.2.1 Filtr pro zobrazení minulosti či budoucnosti

Každý uživatel má možnost zobrazit si kromě aktuálního stavu obsazenosti kanceláře i stav minulý či budoucí. K tomuto účelu slouží jednoduchý filtr, kde uživatel zvolí požadovaný den a čas a vizualizace kanceláře se aktualizuje tak, aby tento požadovaný den a čas zobrazovala, tedy místa se přebarví dle nové obsazenosti a také se aktualizují zobrazované nápovědy s dodatečnými informacemi o obsazenosti zvoleného pracovního místa.

4.3.3 Interaktivní plány kanceláří

Mimo zmíněné vizuální zachycení obsazenosti webová aplikace také nabízí interakci uživatele s plány kanceláří. Uživatel má možnost přímo z náhledu kanceláře změnit velikost kanceláře, přidat nové pracovní místo a posunout ho na jinou pozici pomocí přetažení myší, tzv. „drag and drop“. Ideálně aplikace také nabízí možnost otočení pracovního místa, případně dokonce i přidání dalších objektů do kanceláře (stůl, křeslo, koš).

Veškeré tyto interakce jsou podmíněné administrátorskými právy a pochopitelně jsou dostupné nejen z náhledu kanceláře, ale také z administrátorského rozhraní.

4.3.4 Statistiky obsazenosti kanceláří

Aplikace poskytuje v administrátorském rozhraní (tedy opět podmíněno administrátorskými právy přihlášeného uživatele) statistiky obsazenosti jednotlivých kanceláří.

4.3.4.1 Výčet zobrazovaných statistik

Administrátor má možnost si přehledně na jednotlivých stránkách zobrazit:

- Aktuální obsazenost kanceláří – počet momentálně obsazených a volných míst
  - V celkovém součtu
  - V jednotlivých kancelářích
- Graf dnešní obsazenosti
  - Všech kanceláří – vývoj celkové obsazenosti po jednotlivých hodinách
  - Vybrané kanceláře – vývoj obsazenosti po jednotlivých hodinách
- Graf několikadenní či měsíční obsazenosti
  - Všech kanceláří – vývoj celkové obsazenosti po jednotlivých dnech
  - Vybrané kanceláře – vývoj obsazenosti po jednotlivých dnech

4.3.4.2 Význam údajů

Z těchto informací zanesených formou grafů je možné tedy přehledně vyčíst, které kanceláře a konkrétní hodiny jsou dnes nejméně vytížené a které kanceláře a dny jsou nejméně vytížené z dlouhodobého hlediska. To vedoucím (administrátorům) umožňuje optimalizovat vytíženost kanceláří a odhadnout, kolik zaměstnanců je možné ještě přijmout tak, aby neměli problém najít pracovní místo, případně v jakém momentě bude již potřeba rozšířit počet míst v kancelářích či zajistit nové kanceláře.

4.3.4.3 Vlastní definování požadovaných grafů

V ideálním případě bude aplikace umožňovat také vlastní definování a přidání nového grafu. Tedy administrátor si bude moct sám vytvořit nový graf pro vybrané kanceláře a vybrané rozpětí dnů a časů.

4.3.5 Správa prostřednictvím administrátorského rozhraní

Skrze administrátorské rozhraní je možné spravovat veškeré entity – provádět na nich jakékoliv CRUD operace, tedy libovolně jakékoliv entity vytvářet, zobrazovat, upravovat a mazat. Tyto možnosti úprav se tedy týkají především osob (zaměstnanců), kanceláří, pracovních míst, uživatelů (správa práv a API tokenů) a pochopitelně také jednorázových či opakovaných přiřazení.

4.3.6 API a integrace

Aplikace poskytuje rozhraní REST API pro možnost snadné integrace webové aplikace do libovolné firmy. Většina firem uchovává minimálně databázi svých zaměstnanců a REST API tedy značně usnadňuje práci při počáteční potřebě vytvořit v aplikaci veškeré zaměstnance. Bez jakéhokoliv rozhraní API by bylo nutné veškeré zaměstnance do aplikace přidávat ručně a všechny údaje přepisovat. S REST API rozhraním je možné tento proces snadno automatizovat pomocí skriptu, který skrze POST requesty vytvoří požadované zaměstnance na základě interní databáze dané firmy.

Další výhodou REST API je usnadnění jakékoliv komunikace s dalšími aplikacemi používanými ve firmě. Pomocí jednoduchých skriptů je tak možné data o přiřazení zaměstnanců k pracovním místům propsat do kalendářů nebo třeba určité přiřazení zaměstnance automaticky zrušit v momentě, kdy si zaměstnanec ohlásí na daný den dovolenou v jiné aplikaci používané firmou.

4.3.6.1 Volný přístup

Část REST API je přístupná volně všem uživatelům bez nutnosti poskytnutí API tokenu. Konkrétně se jedná o veškeré GET requesty, tedy čtení jakýchkoliv informací o entitách. Při volném přístupu musí být pochopitelně kvůli bezpečnosti skryté citlivé údaje jako jsou hesla uživatelů či API tokeny.

4.3.6.2 Administrátorský přístup

Administrátoři mají možnost vytvářet (vygenerovat) v administrátorském rozhraní API tokeny pro jednotlivé uživatele. Platí, že API token každého uživatele kopíruje práva daného uživatele, tedy API token běžného uživatele bude poskytovat stejné oprávnění, které je k dispozici i v rámci volného přístupu. API token administrátora oproti tomu tedy poskytuje přístup k celému API, které umožňuje provádět libovolné CRUD operace pro libovolné entity pomocí GET (včetně GET COLLECTION), POST, PUT, PATCH a DELETE requestů.

Kapitola 5
Návrh

5.1 Architektura

Tato část práce představuje podrobný přehled architektonického rámce, který je základem webové aplikace, a zdůrazňuje, jak jsou různé technologie integrovány pro vytvoření uceleného a výkonného systému.

Architektonický návrh aplikace uplatňuje třívrstvou MVC (Model-View-Controller) architekturu, čímž odděluje zájmy a separuje oblasti zodpovědnosti, tedy naplňuje princip SoC (Separation of Concerns).

Každá technologie hraje specifickou roli. PostgreSQL slouží jako úložiště dat. Doctrine překlenuje mezeru mezi logikou aplikace a databázovými operacemi. Symfony zpracovává operace na straně serveru a s pomocí šablonovacího nástroje Twig generuje HTML kód, čímž vykresluje většinu uživatelského rozhraní. Twig také slouží k integraci React kódu, který vylepšuje prostředí front-endu a doplňuje jej o složitější a interaktivní prvky. API Platform rozšiřuje aplikaci o rozhraní REST API. Tato struktura zajišťuje nejen modulární a udržovatelnou kódovou bázi, ale také podporuje snadnou rozšiřitelnost a modifikovatelnost.

Tato architektura je navržena tak, aby byla robustní a zároveň flexibilní a dokázala se přizpůsobit budoucímu technologickému pokroku a vyvíjejícím se požadavkům projektu. Kombinace těchto technologií poskytuje komplexní rámec pro budování moderní, efektivní a uživatelsky přívětivé webové aplikace. Podrobněji je architektura popsána v následujících podkapitolách.

5.1.1 PostgreSQL – databázový systém

Pro ukládání a správu dat je použit PostgreSQL, propracovaný objektově-relační databázový systém. Databáze PostgreSQL, známá svou robustností a škálovatelností, je nedílnou součástí pro zpracování veškerých datových struktur, které aplikace vyžaduje.

5.1.2 Doctrine – objektově relační mapování

Aplikace využívá Doctrine, objektově-relační mapovač (ORM), pro správu entit v rámci Symfony. Doctrine slouží jako výkonná vrstva pro abstrakci databáze a umožňuje bezproblémovou interakci s databází PostgreSQL. Tato integrace zjednodušuje proces manipulace s daty a jejich načítání a zajišťuje hladký tok informací mezi aplikací a databází.

5.1.3 Symfony a Twig – základní rámec

Základem aplikace je Symfony, univerzální a robustní framework PHP, který je doplněn šablonovacím enginem Twig. Tato kombinace tvoří páteř aplikace, která spravuje logiku na straně serveru a vykresluje šablony HTML. Struktura Symfony poskytuje dobře organizovaný a škálovatelný přístup k vývoji webových aplikací a umožňuje efektivní správu různých komponent aplikace.

5.1.4 React – interaktivní části frontendu

Pro rozšíření a vylepšení designu uživatelského rozhraní a zvýšení interaktivity aplikace využívá React, populární knihovnu jazyka JavaScript. React se v některých částech aplikace používá k vykreslování složitějších a interaktivnějších komponent, konkrétně například k vizualizaci kanceláře a poskytnutí funkce snadného posunutí pracovních míst pomocí přetažení myší.

5.1.5 API Platform – rozhraní API

API Platform se využívá k vytváření a správě většiny koncových bodů (endpointů) API, které jsou nedílnou součástí aplikace. Pomocí anotací API Platform aplikace efektivně vystavuje své funkce prostřednictvím rozhraní REST API, což usnadňuje rychlou automatizovanou výměnu dat a umožňuje především snadnou integraci aplikace. Výhodou API Platform je také automatické generování API dokumentace (OpenAPI specifikace).

5.2 Databázový model

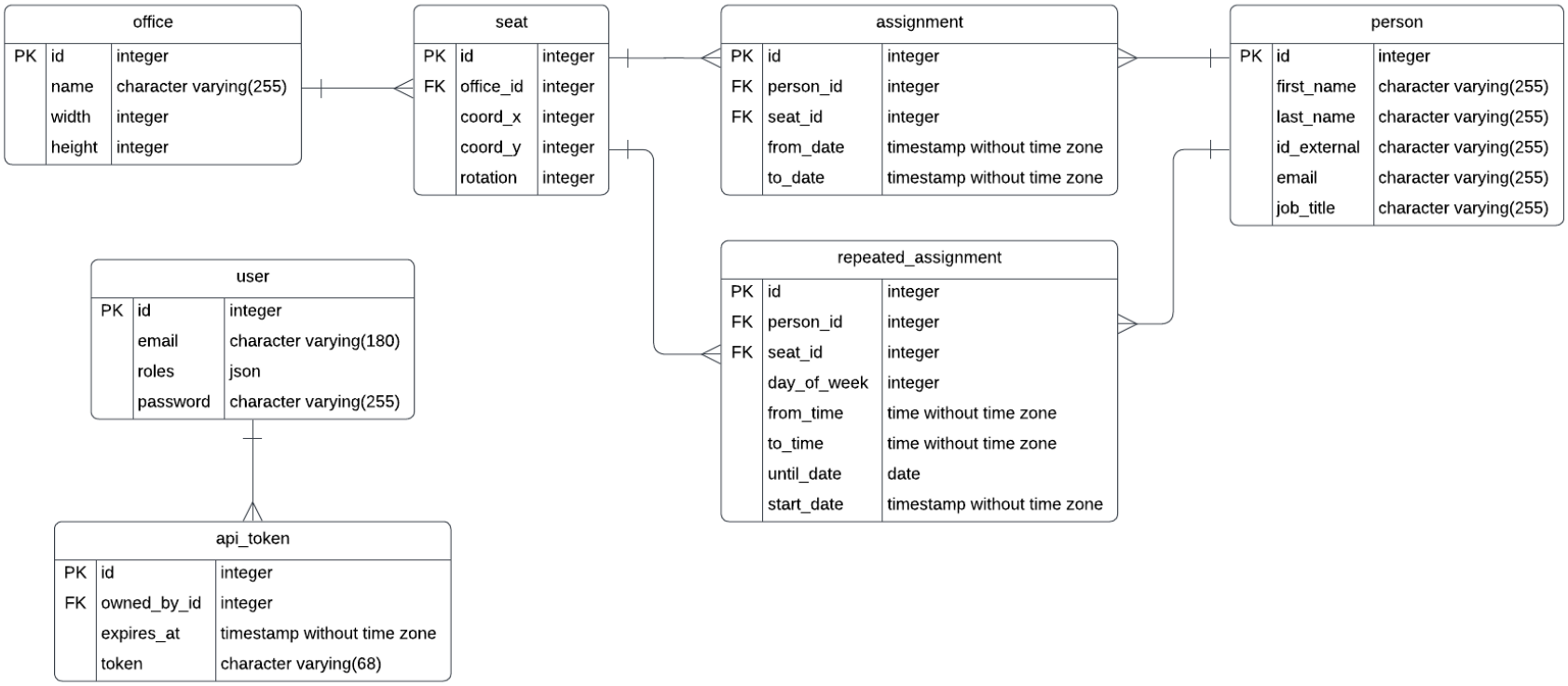
Pro dosažení všech stanovených funkcí a splnění požadavků na aplikaci je třeba správně rozmyslet a navrhnout databázový model. Prvotní databázový model vycházel z následujícího textu:

*Firma používající webovou aplikaci spravuje kanceláře (**Office**). Každá kancelář má typicky svůj název a své rozměry a jsou v ní pracovní místa (**Seat**). Pracovní místa nemají své názvy. Jsou identifikována pouze unikátním ID a pro jejich umístění v prostoru kanceláře jsou definovány souřadnice X a Y.*

*Zaměstnanci firmy jsou reprezentováni entitou **Person**, která uchovává informace o zaměstnanci (např. jméno, příjmení, e-mail). Tito zaměstnanci jsou přiřazováni na pracovní místa. Vzhledem k tomu, že zaměstnanec může mít současně přiřazeno více pracovních míst a stejně tak i místo může být přiřazeno vícero zaměstnancům, vzniká mezi zmíněnými entitami vazba „many-to-many“, tedy vícenásobná vazba na obou stranách vztahu. Je tedy třeba provést dekompozici a vytvořit propojovací entitu **Assignment**, která bude navíc uchovávat informace o počátku a konci přiřazení.*

Výsledný databázový model, viz obrázek 5.1, ještě oproti úvodnímu textovému popisu doplňuje databázi o entitu **RepeatedAssignment**, která slouží pro opakující se přiřazení a obsahuje tedy informace: den v týdnu (pro který dochází k opakování), čas začátku a konce, počáteční a koncové datum tohoto opakujícího se přiřazení. Dále je doplněn o uživatele (**User**), kteří se přihlašují e-mailem a heslem a mají přidělené role poskytující běžná či administrátorská práva. K uživatelům je možné přiřazovat API tokeny včetně data expirace. Práva, která **API token** poskytuje, odpovídají právům vlastníka daného tokenu a není tedy třeba je uchovávat.

Tento model zohledňuje a zachycuje veškeré objekty (entity), které jsou potřebné pro naplnění požadavků na aplikaci a vytváří tedy základ pro objektově-orientovaný přístup k vývoji aplikace.



Obrázek 5.1 Databázový model

5.3 Ošetření času

Každá aplikace, která pracuje s časem, musí také dbát na správné ošetření časových pásem včetně letního a zimního času. Řešení tohoto problému je značně komplikovanější, než se může na první pohled zdát. Letmý náhled do komplexnosti tohoto problému je nastíněn například ve videu „The Problem with Time & Timezones – Computerphile“ od populárního webového vývojáře Toma Scotta [33].

5.3.1 Časová pásma

Pokud jde o časová pásma a jejich ošetření při ukládání a zobrazování data a času, nabízí se vícero možností, jak k tomuto problému přistoupit. Možnosti, které byly zváženy, včetně jejich výhod a nevýhod, jsou popsány v následujících podkapitolách.

5.3.1.1 Zobrazení a ukládání dle časového pásma uživatele

Jednou z možností je určovat časové pásmo na základě aktuální polohy uživatele a zobrazovat i ukládat přiřazení zaměstnanců k místům v tomto časovém pásmu. Tento přístup umožňuje společnosti vlastnit a přidávat kanceláře v různých časových pásmech po celém světě, ovšem otevírá také řadu problémů.

Například, pokud by manažer z pražské kanceláře odjel na dovolenou do jiného časového pásma a během své dovolené částečně pracoval a vytvářel či upravoval přiřazení zaměstnanců k pracovním místům, pak by se veškerá takto vytvořená či upravená přiřazení zapisovala chybně posunutá o tolik hodin, jaký by byl aktuální rozdíl českého časového pásma a časového pásma manažera na dovolené.

Stejně tak i zaměstnanec na dovolené by viděl své budoucí přiřazení posunuté a kdyby se koncem dovolené chtěl podívat, v kolik hodin má následující den přiřazené místo, aplikace by mu zobrazila, že např. od 8. hodiny, ale reálně v českém časovém pásmu by jeho místo bylo dostupné až od 10. hodiny.

Tento přístup je použit například u mobilních aplikací Outlook a Google kalendář. Nutno podotknout, že u kalendáře Google je možné toto chování vypnout a v nastavení napevno změnit zobrazované časové pásmo na jiné [34, 35].

5.3.1.2 Zobrazení a ukládání ve fixně daném časovém pásmu

Druhou z možností je časové pásmo pevně zafixovat. To sice znemožní, aby společnost měla kanceláře v různých časových pásmech, ale vyřeší to problémy někdy nechtěného přepočítávání časů do různých časových pásem.

Tedy, jakékoliv nové jednorázové či pravidelné přiřazení bude vždy vytvořeno i zobrazeno v časové zóně aplikace bez ohledu na aktuální polohu uživatele. V momentě, kdy se nacházejí kanceláře firmy v jednom časovém pásmu, jedná se o plně vyhovující řešení. Naopak, pokud by měla firma kanceláře v různých časových pásmech, tento přístup bude zcela nevyhovující a aplikace bude pro firmu nepoužitelná.

5.3.1.3 Možnost zobrazení v různých časových pásmech

Třetí možností je implementace libovolného přepínání do různých časových pásem. Každá kancelář by zároveň mohla mít jako atribut své výchozí časové pásmo (dle místa, ve kterém se kancelář fyzicky nachází). Při zobrazení kanceláře X by pak uživatel mohl zvolit, zda si přeje zobrazovat data a časy dle své aktuální polohy, dle polohy kanceláře nebo dle jiné vlastní volby. Stejně tak i manažer (administrátorská role) by při vytváření a upravování přiřazení měl možnost zvolit

časové pásmo kanceláře nebo své aktuální. Tento přístup by byl uživatelsky nepřívětivější, ovšem jeho aplikování a také testování by bylo velmi náročné.

Tento přístup momentálně nabízí desktopová aplikace Outlook, kde je možné si k defaultnímu časovému pásmu přidat ještě další 2 časová pásma. Při zobrazení kalendáře jsou pak všechna tato pásma současně zobrazena [34].

5.3.2 Výběr vhodné metody pro implementaci

Po zvážení náročnosti a kladů a záporů vyvíjená webová aplikace implementuje druhou možnost a časové pásmo bude zafixované pro Českou republiku. Toto rozhodnutí zdánlivě omezuje použití aplikace pouze na území České republiky, nicméně nutno poznamenat, že pokud by o aplikaci měla zájem firma sídlící v jiné zemi, změnit zafixované časové pásmo je poměrně jednoduchý a časově nenáročný úkol s nímž si poradí i méně zkušený programátor.

Implementace třetí možnosti, tedy přepínání časových pásem, bude ponechána pro případný budoucí vývoj a další verze aplikace.

5.3.3 Zimní a letní čas

Pro správné fungování aplikace je třeba řádně ošetřit změny zimního a letního času. Do databáze je vhodné ukládat datový typ DateTime (den a čas) v UTC. Pro každé jednorázové přiřazení tak aplikace musí správně vyhodnotit, že bylo vytvořeno na den a čas, který spadá do zimního (UTC+2) či letního (UTC+1) času. Dle tohoto vyhodnocení pak aplikace do databáze uloží čas posunutý o jednu či dvě hodiny zpět.

V tomto ohledu jde naštěstí jazyk PHP naproti a při použití třídy DateTime v kombinaci s balíčkem EasyAdmin (3 nebo 4) je třeba pouze nastavit ukládání do databáze v UTC, ale zobrazování v pásmu „Europe/Prague“. O zbytek už se použité třídy a balíčky starají automaticky.

Větší výzva přichází pro ukládání a zobrazování opakujících se událostí. Pro opakované události se v adminu ukládá pouze čas, ovšem databáze jej ukládá jako DateTime a to s datem 01.01.1970. Protože datum 01.01. je v zimním časovém pásmu (UTC+1), výchozí chování je takové, že aplikace uloží čas posunutý o jednu hodinu. Tedy například 17.00 bude do databáze uloženo jako 16.00 UTC. EasyAdmin bohužel pracuje v tomto ohledu nešikovně, protože čas přepočítává na základě toho, zda je aktuální den zimní či letní čas. Tedy, taková opakovaná událost se zobrazí s počátečním časem 17.00 během zimního času či 18.00 během letního času. To rozhodně není žádoucí chování. Proto je po úvaze čas ukládán bez jakéhokoliv časového posunu, tedy 17.00 se uloží jako 17.00 UTC. Při zobrazení pro uživatele pak čas nemusí být nikam posouván, a je zobrazen tak, jak je uložen v databázi. Tento přístup bohužel přináší komplikaci při hledání kolizí jednorázových a opakujících se přiřazení, její řešení je nastíněno v následující podkapitole 5.4.

5.4 Validace přiřazení

V oblasti webových aplikací určených pro správu pracovních míst a přiřazování zaměstnanců je kritickou součástí proces validace. Tento proces zajišťuje, že přiřazení zaměstnanců k pracovním místům je proveditelné, bezkolizní a odpovídá požadavkům aplikace. Tato podkapitola se zabývá validačními mechanismy používanými v aplikaci a věnuje pozornost zpracování jednorázových a opakovaných přiřazení.

Validační mechanismus v této webové aplikaci hraje klíčovou roli. V následujících podkapitolách je demonstrována pečlivá rovnováhu mezi přísnou detekcí kolizí a uživatelsky orientovaným přístupem k návrhu, například intuitivním zpracováním přiřazení, které končí a začínají ve stejnou minutu.

5.4.1 Uživatelské rozhraní pro přiřazování

CRUD operace pro správu jednorázových i opakovaných přiřazení jsou podmíněny administrátorskými právy a jsou k dispozici z administrátorského rozhraní prostřednictvím formulářů. Každé přiřazení je spojeno s právě jedním pracovním místem a s právě jednou osobou. Administrátor pracovní místo a osobu vybírá ze všech instancí, které v databázi aplikace existují. Není tedy problém pokusit se vytvořit nové přiřazení, které bude tvořit kolizi s již existujícím, proto je třeba, aby aplikace nové přiřazení validovala a v případě nalezené kolize nové přiřazení neuložila a vypsala chybovou hlášku.

5.4.2 Logika ověřování

Základním kamenem této aplikace je její robustní validační logika, která je klíčová před dokončením jakéhokoli nového přiřazení, ať už jednorázového, nebo opakovaného. Validace zajišťuje, že nové přiřazení není v rozporu se stávajícími přiřazeními z hlediska času a prostoru.

5.4.2.1 Jednorázová přiřazení

Aplikace používá specifickou třídu `Assignment`, která zpracovává jednorázová přiřazení. Tato třída obsahuje atributy pro identifikaci, zúčastněnou osobu, přidělené místo a dobu trvání přiřazení. Uvedený pseudokód (viz výpis kódu 1) popisuje logiku pro ověřování nových jednorázových přiřazení oproti stávajícím přiřazením s důrazem na detekci kolizí na základě časového překryvu a místa nebo zúčastněné osoby.

5.4.2.2 Opakovaná přiřazení

Pro opakovaná přiřazení je použita třída `RepeatedAssignment`. Ta obsahuje další atributy, které zohledňují opakující se povahu těchto přiřazení, jako je den v týdnu a datum začátku a konce opakujícího se vzoru. Logika ověřování pro nová opakovaná přiřazení (viz výpis kódu 2) vůči stávajícím jednorázovým přiřazením má podobnou strukturu, pouze s přidanou složitostí zohledňující opakující se vzor včetně ošetření letního a zimního času.

5.4.3 Algoritmus detekce kolizí

Uvedené pseudokódy 1 a 2 ilustrují pro ukázku dva z použitých algoritmů pro detekci kolizí. Algoritmy zahrnují iteraci každého existujícího jednorázového přiřazení (`A`) v databázi a jeho porovnání s nově vytvářeným (či upravovaným) jednorázovým přiřazením (`B`) viz pseudokód 1 či s nově vytvářeným (či upravovaným) opakovaným přiřazením (`REP`) viz pseudokód 2. Pokud je nalezeno již existující přiřazení, které se s nově vytvářeným (či upravovaným) časově překrývá a buď místo, nebo přiřazená osoba je stejná, „vyhodí se výjimka“, tedy dojde k chybě validace, která zabrání vytvoření konfliktních přiřazení.

V pseudokódu 1 je zároveň také podmínka pro kontrolu shodného identifikátoru nového přiřazení. Důvodem této podmínky je fakt, že se tento algoritmus používá jak při vytváření nového přiřazení, tak i při úpravě již existujícího. Pokud dochází však k úpravě existujícího, chci jej kontrolovat pouze vůči jiným, jinak by upravované přiřazení kolidovalo samo se sebou.

V pseudokódu 2 by mohla vyvolávat nejasnosti funkce, která převádí datum a čas do české časové zóny. Tato funkce je nevyhnutelná z důvodu, který je podrobně popsán v sekci 5.3.3, tedy opakovaná přiřazení musí ukládat čas v UTC ač ve skutečnosti se jedná o čas, který není převeden do UTC a je mu pouze uměle přepsaná časová zóna z české na UTC.

```
vytvoř přiřazení B z dat od uživatele
pro každé přiřazení A z databáze:
  pokud (A.počátek_přiřazení je dříve než B.konec_přiřazení
  a zároveň B.počátek_přiřazení je dříve než A.konec_přiřazení
  a zároveň A.id není rovno B.id)
    pokud (A.pracovní_místo je shodné s B.pracovní_místo)
      vyhoď chybu validace: kolize místa s přiřazením A
      konec
    pokud (A.přiřazená_osoba je shodná s B.přiřazená_osoba)
      vyhoď chybu validace: kolize osoby s přiřazením A
      konec
konec cyklu
ulož nové přiřazení B do databáze
```

Výpis kódu 1 Pseudokód pro validaci jednoráz. přiřazení vůči uloženým jednoráz. přiřazením

```
// čas() vrátí pouze čas ze vstupního 'datačasu'
// v_české_časové_zóně() vrátí datum a čas převedený do české časové zóny

vytvoř pravidelné přiřazení REP z dat od uživatele
pro každé přiřazení A z databáze:
    pokud (
          REP.den_v_týdnu je shodný s den_v_týdnu(A.počátek_přiřazení)
        a zároveň
          čas(REP.počátek_přiřazení) je dříve než
          čas(v_české_časové_zóně(A.konec_přiřazení))
        a zároveň
          čas(v_české_časové_zóně(A.počátek_přiřazení)) je dříve než
          čas(REP.konec_přiřazení)
        a zároveň
          REP.počátek_opakování je dříve než/shodně s A.počátek_přiřazení
        a zároveň
          (REP je nekonečně se opakující
          nebo
          A.konec_přiřazení je dříve než/shodně s REP.konec_opakování)
    )
        pokud A.pracovní_místo je shodné s REP.pracovní_místo
            vyhoď chybu validace: kolize místa s přiřazením A
            konec
        pokud A.přiřazená_osoba je shodná s REP.přiřazená_osoba
            vyhoď chybu validace: kolize osoby s přiřazením A
            konec
konec cyklu
ulož nové přiřazení REP do databáze
```

Výpis kódu 2 Pseudokód pro validaci opakovaného přiřazení vůči uloženým jednoráz. přiřazením

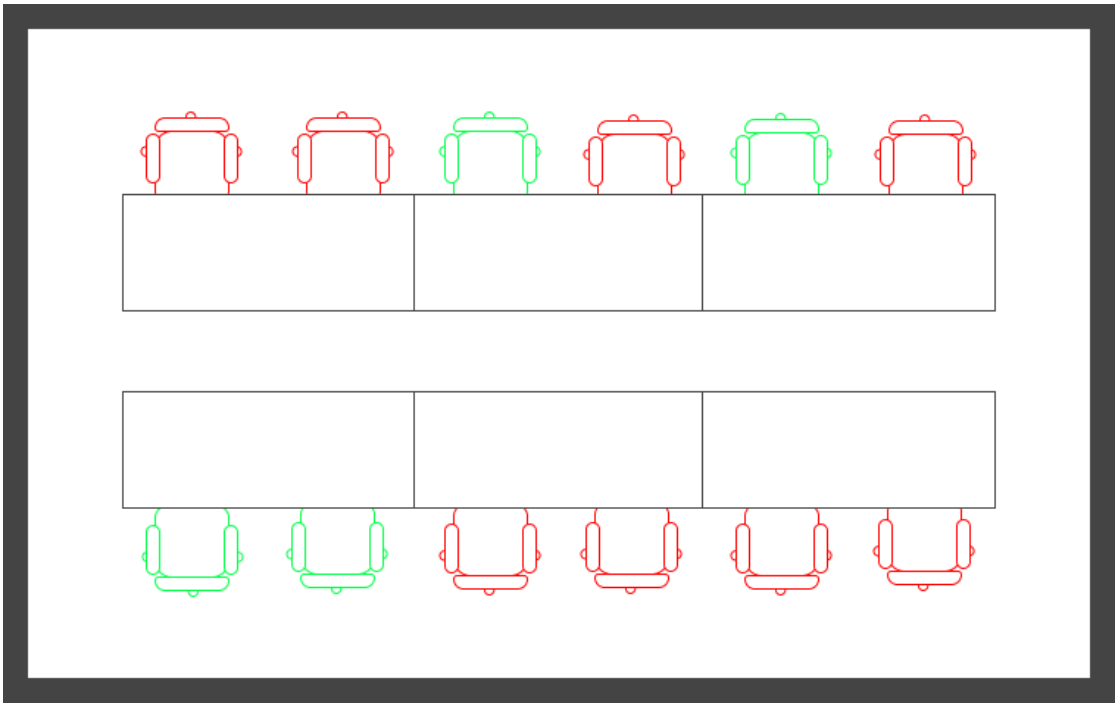
5.5 Zobrazení kanceláře

Důležitou a nepostradatelnou součástí vyvíjené aplikace je vytvořit vizuálně atraktivní a interaktivní zobrazení kanceláře. Vzhledem k tomu, že pro většinu zobrazovaných stránek je zamýšleno použít Twig šablony a serverová část aplikace tedy neslouží pouze k poskytnutí API endpointů (koncových bodů), ale také k definování šablon pro generovaný kód HTML, je žádoucí, aby byl kód Javascriptu integrován do Symfony aplikace prostřednictvím nástroje Webpack Encore.

Webpack Encore je nástroj frameworku Symfony pro správu frontendových zdrojů (CSS a JavaScript) a podporuje integraci frameworků Vue a React. Oba ze zmíněných frameworků jsou schopny poskytnout uspokojivé zobrazení a interaktivitu kanceláří. Pro účel tohoto projektu byl v rámci návrhu zvolen framework React z důvodu své sady komponent React DnD, která poskytuje robustnější a modulárnější přístup a tedy snazší možnost rozvoje v případě budoucího vývoje webové aplikace nad rámec této bakalářské práce.

5.5.1 Návrh designu

Pro snazší pozdější implementaci zobrazení kanceláře, která je včetně ukázky rozebrána v podkapitole 6.6.1, bylo zobrazení kanceláře také navrženo prostřednictvím nástroje Figma (nástroj pro vytváření designu aplikací). Navržené zobrazení kanceláře je zachyceno na obrázku 5.2.



Obrázek 5.2 Návrh zobrazení kanceláře v nástroji Figma

Kapitola 6
Implementace

V této kapitole je představena praktická realizace předchozích návrhů s využitím vybraných technologií a frameworků. Klíčové aspekty implementace zahrnují především vývoj serverové logiky v PHP Symfony, vytvoření Twig šablon pro generování kódu HTML a vývoj interaktivních frontendových komponent s využitím JavaScriptové knihovny React.

Cílem je demonstrovat, jak byly teoretické koncepty převedeny do funkčního softwarového řešení, a jakým způsobem byly řešeny specifické výzvy vývoje. Tato kapitola poskytuje náhled do hlavních implementačních kroků, včetně některých klíčových částí zdrojového kódu a ukázek uživatelského rozhraní, jež ilustrují základní funkce aplikace.

6.1 Databáze a entity

Pro objektově-relační mapování byla použita sada knihoven Doctrine, která je součástí balíčku Symfony ORM. Pro tvorbu entit byl použit balíček Symfony Maker (MakerBundle), který umožňuje snadno vytvářet entity a definovat jejich atributy a vztahy. Takto vytvořené entity je následně možné libovolně editovat jak nástrojem Maker, tak i přímo v kódu. Po vytvoření či editaci entit je možné provést migraci, která vykoná nad databází odpovídající SQL kód.

6.2 Controller (Kontrolér) a Twig šablony

Pro routování (tedy mapování URL adres) slouží metody tříd `Controller`. Tyto metody slouží k případnému uplatnění jednoduché logiky a k vykreslení webové stránky prostřednictvím Twig šablony včetně předání případných parametrů viz výpis kódu 3.

```
class PersonController extends AbstractController {
    #[Route('/person', name: 'app_person')]
    public function index(Request $request): Response {
        $searchTerm = $request->query->get('search', '');
        $persons = $this->personRepository->search($searchTerm);
        return $this->render('person/show.html.twig', [
            'persons' => $persons,
            'search_term' => $searchTerm]);
    }}
```

Výpis kódu 3 Část kódu metody `index()` třídy `PersonController` (přeformátováno)

Jak je z výpisu kódu 3 patrné, po přistoupení uživatele na url adresu `/person` je volána metoda `index()`, která vykresluje webovou stránku pomocí Twig šablony `person/show.html.twig`. Tato šablona generuje výsledný HTML kód zobrazované stránky prostřednictvím syntaxe Twig a její ukázka je zobrazena ve výpisu kódu 4. V této konkrétní šabloně lze spatřit mimo jiné rozšíření (extends) o soubor `base.html.twig`, který obsahuje například navigační panel, dále HTML kód pro vytvoření vyhledávacího pole a tlačítka, HTML kód pro zobrazení tabulky, a pomocí Twig syntaxe vytvořený „for cyklus“ pro procházení pole `persons`, který za jednotlivé řádky tabulky dosazuje jednotlivé atributy objektů `person`.

```
{% extends 'base.html.twig' %}
{% block title %}People{% endblock %}
{% block body %}
    {# ... #}
    <form method="GET" action="{{ path('app_person') }}" class="mb-3 d-flex
    ↪  inline-search">
        <input type="text" name="search" class="form-control"
        ↪  placeholder="Search" value="{{ search_term }}" />
        <button type="submit" class="btn btn-primary">Search</button>
    </form>
    <div class="table-responsive" style="max-height: 70vh; overflow-y: auto;">
    <table class="modern-table">
        <thead>
            <tr>
                <th>ID</th>
                <th>First Name</th>
                <th>Last Name</th>
                <th>Email</th>
                <th>Job Title</th>
            </tr>
        </thead>
        <tbody>
        {% for person in persons %}
            <tr style="cursor:pointer" onclick="window.location='{{
            ↪  path('app_person_detail', {'id': person.id}) }}'">
                <td>{{ person.id }}</td>
                <td>{{ person.firstName }}</td>
                <td>{{ person.lastName }}</td>
                <td>{{ person.email }}</td>
                <td>{{ person.jobTitle }}</td>
            </tr>
        {% else %}
            <tr>
                <td colspan="5">No records found</td>
            </tr>
        {% endfor %}
        </tbody>
    </table>
    {# ... #}
{% endblock %}
```

Výpis kódu 4 Část kódu Twig šablony `person/show.html.twig`

Twig šablona aplikováním logiky v ní zanesené (například zmíněný „for cyklus“, v jiných šablonách pak typicky „if-else podmínky“) vytvoří výsledný kód HTML, který je následně webovým prohlížečem zobrazen. Obrázek 6.1 zachycuje část webové stránky na URL adrese `/people`, která je vytvořena již zmíněnou částí Twig šablony (viz výpis kódu 4). Zobrazená data jsou samozřejmě falešná (automaticky generovaná pomocí balíčku DoctrineFixturesBundle v kombinaci s balíčkem ZenstruckFoundryBundle).

Obrázek 6.1 Část výsledné stránky na URL adrese `/people`

6.3 Repository (Repozitář)

Další podstatnou částí aplikace jsou repozitáře (`Repository`), které umožňují pomocí Doctrine Query Builder definovat vlastní databázové dotazy. Toho je v implementaci využito především u entit `Assignment` a `RepeatedAssignment` (tedy entity sloužící pro přiřazení), u kterých je nutné rychle získat všechna přiřazení pro zadaný den a čas (případně rozpětí). Toho je využito jak pro logiku filtru, který pro uživatelem zadaný den a čas zobrazuje minulá či budoucí přiřazení, tak pro validační logiku, která je podrobněji rozebrána v následující podkapitole 6.4.

6.4 Validace

Pro automatickou validaci aplikace využívá komponentu `Validator` poskytovanou frameworkem Symfony. Tato komponenta nabízí tvorbu omezení (constraints) pro atributy entit. V implementaci jsou použita například omezení `NotBlank` (ověřuje, zda hodnota není prázdná), `Email` (ověřuje, zda je hodnota platnou e-mailovou adresou) nebo `GreaterThanOrEqual` (ověřuje, zda je hodnota větší nebo rovna jiné hodnotě definované v parametrech).

Pro validaci při vytváření nových přiřazení (`Assignment` a `RepeatedAssignment`) je vytvořeno vlastní ověřovací omezení a vlastní validační třída `IsAvailableAssignmentValidator`. Tato třída přistupuje k repozitářům tříd `Assignment` a `RepeatedAssignment` a prostřednictvím metod daných repozitářů vykonává databázové dotazy, které slouží pro nalezení kolidujících přiřazení.

```
class IsAvailableAssignmentValidator extends ConstraintValidator
{
    // ...
    public function validate(mixed $value, Constraint $constraint): bool
    {
    $valid = true;
    $assignmentRepository = $this->em->getRepository(Assignment::class);

    // ...
    $personConflictsWithAssignments =
    ↪  $assignmentRepository->findOverlappingAssignments($value, "person");
    // ...

    // Overlapping one-time assignments using same person
    if (count($personConflictsWithAssignments) > 0) {
        $this->context->buildViolation($constraint->message)
        // ...
        ->addViolation();
        $valid = false;
    }
    // ...
}
```

Výpis kódu 5 Část kódu vlastní validační třídy

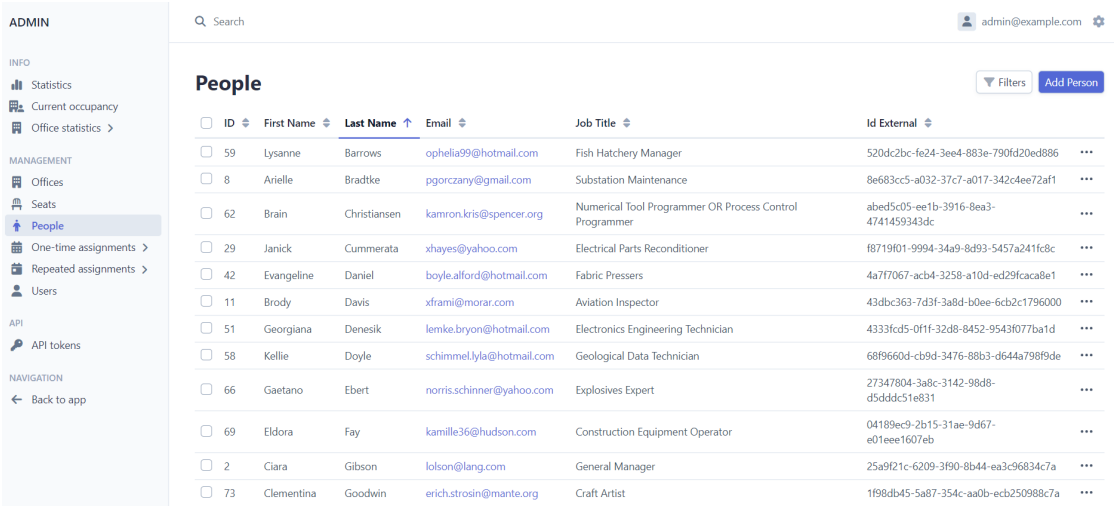
Jak je z výpisu kódu 5 patrné, pro zjištění a spočítání konfliktů je použita vlastní metoda `findOverlappingAssignments()` vytvořená v repozitáři přiřazení (`AssignmentRepository`). Tato metoda vytváří databázový dotaz pro nalezení kolidujících přiřazení a je tedy nejzásadnějším zdrojem validační logiky. Je založena na pseudokódech 1 a 2 a její část je k nahlédnutí ve výpisu kódu 6.

```php
public function findOverlappingAssignments(
Assignment|RepeatedAssignment $assignment, string $param) : mixed
{
    // ...
    // Find overlapping assignments with new one-time Assignment
    if ($assignment instanceof Assignment) {
        // Time overlapping assignment
        $qb->andWhere('e.fromDate < :toDate AND e.toDate > :fromDate AND e.id
        ↪  <> :id')
        ->setParameter('fromDate', $assignment->getFromDate())
        ->setParameter('toDate', $assignment->getToDate())
        ->setParameter('id', $assignment->getId() ?: -1);

        // Filter just those with the same person
        if ($param === 'person') {
            $qb->andWhere('e.person = :person')
            ->setParameter('person', $assignment->getPerson());
        }
        // Filter just those with the same seat
        else {
            $qb->andWhere('e.seat = :seat')
            ->setParameter('seat', $assignment->getSeat());
        }
        return
        ↪  $qb->getQuery()->setHydrationMode(AbstractQuery::HYDRATE_ARRAY)->execute();
    }
    // Find overlapping assignments with new RepeatedAssignment
    else {
        //...
        $sql = "SELECT *, e.from_date AS \"fromDate\", e.to_date AS \"toDate\"
        FROM assignment e
        WHERE e.$param_id = :param_id AND (EXTRACT(dow FROM e.from_date) =
        ↪  :dayOfWeek OR EXTRACT(dow FROM e.to_date) = :dayOfWeek) AND
        ↪  (((e.from_date::timestamptz AT TIME ZONE 'Europe/Prague')::TIME <
        ↪  :toTime AND (e.to_date::timestamptz AT TIME ZONE
        ↪  'Europe/Prague')::TIME > :fromTime)) AND (e.from_date >=
        ↪  :startDate)";

        $params = [
            //...
        ];
        //...
        $stmt = $connection->prepare($sql);
        return $stmt->executeQuery($params)->fetchAllAssociative();
    }
}
```

Výpis kódu 6 Kód pro validaci jednorázového přiřazení vůči uloženým přiřazením

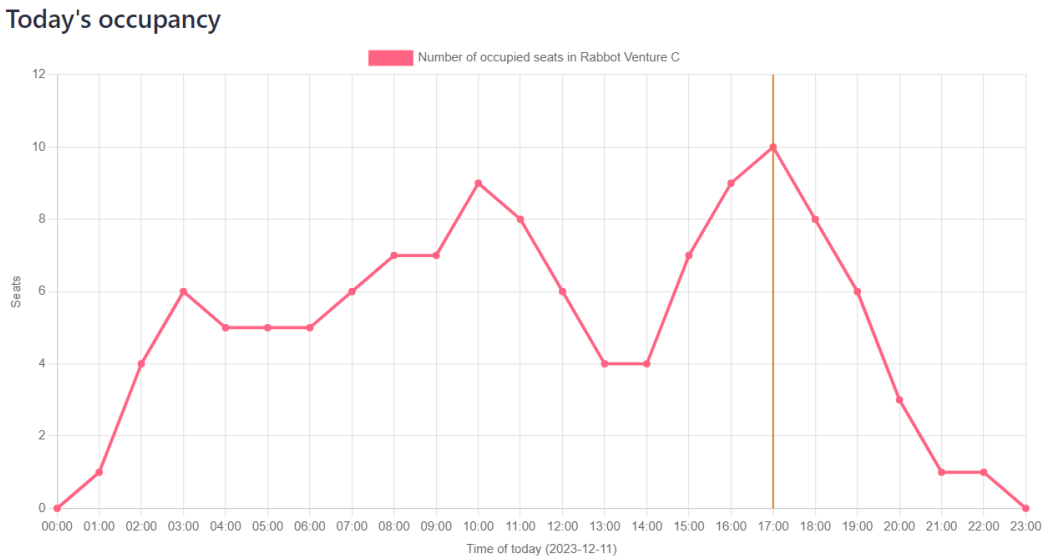
6.5 Administrátorské rozhraní

Důležitou částí implementace je také administrátorské rozhraní (dále jen admin), které umožňuje uživatelům s potřebnými administrátorskými právy provádět libovolné CRUD operace pro veškeré objekty (entity). Implementaci adminu značně usnadňuje použití komponenty EasyAdmin. EasyAdmin je rychlý a moderní generátor adminu pro Symfony aplikace. Pomocí předdefinovaných tříd a metod umožňuje snadno vytvářet v adminu nové stránky pro úpravu entit včetně nastavení zobrazovaných a upravovatelných atributů pro každou entitu.



Obrázek 6.2 Administrátorské rozhraní implementované pomocí EasyAdmin

Mimo to také poskytuje třídu pro tvorbu informačních desek (dashboard), které jsou v implementaci adminu využity pro zobrazení různých statistik viz podkapitola 4.3.4. Pro vizualizaci statistik je využit balíček Symfony UX Chart.js, který umožňuje integraci velmi populární knihovny pro tvorbu grafů v jazyce JavaScript, Chart.js, do Symfony aplikací.



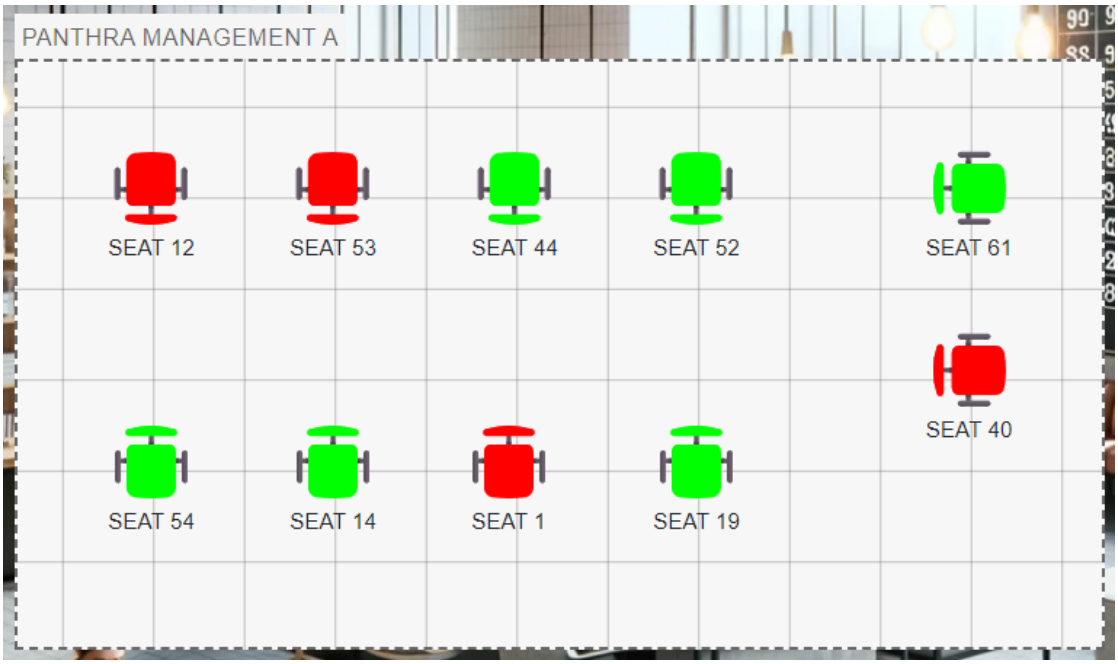
Obrázek 6.3 Zobrazení statistik pomocí Chart.js – dnešní obsazenost vybrané kanceláře

6.6 React

6.6.1 Vizualizace a interaktivita plánů kanceláří

Pro vizualizaci plánů kanceláří a zobrazení a interaktivitu pracovních míst byl využit balíček React DnD. Jedná se o sadu nástrojů React, které umožňují vytvářet rozličná rozhraní typu „drag and drop“. Pracovní místa jsou zobrazovaná pomocí SVG ikony kancelářské židle, která je zbarvena červenou či zelenou barvou podle toho, zda je místo v aktuálním čase (či ve filtrovaném čase) obsazené nebo volné viz obrázek 6.4.

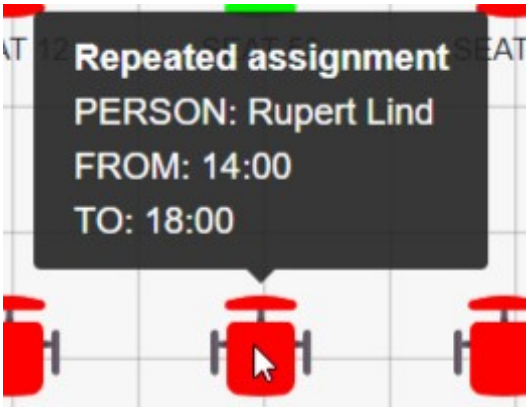
Uživatel s administrátorskými právy má díky implementaci balíčku React DnD možnost snadno přesouvat tažením myši pracovní místa či je prostřednictvím dvojkliku myši (double-click) otáčet o 90°. Pro snadnější manipulaci je plán kanceláře doplněn o mřížku a pracovní místa (kancelářské židle) se dle dané mřížky automaticky po přesunu zarovnávají.



Obrázek 6.4 Implementované zobrazení kanceláře

Aplikace pochopitelně poskytuje dle specifikovaných požadavků dodatečné informace o obsazených pracovních místech, konkrétně jméno osoby, která na zvoleném místě sedí a časové rozpětí tohoto přiřazení. Tyto informace jsou zobrazeny jak v nápovědě (tooltip) po přiložení myši, tak i po kliknutí v samostatném boxu pro možné použití aplikace na dotykových zařízeních viz obrázky 6.5 a 6.6.

V požadavcích na vizualizaci kanceláře (viz podkapitola 4.3.3) je zmíněno, že v ideálním případě dojde také k přidání možnosti vkládání dalších objektů do plánu kanceláře. Stejně tak s možností přidávání a zobrazení stolů počítá i návrh zobrazení kanceláře (viz obrázek 5.2). Tato funkcionalita nakonec není součástí implementace a je ponechána pro případný budoucí rozvoj aplikace. Díky modularitě kódu React (oddělení jednotlivých komponent) je však aplikace dobře připravena na toto možné rozšíření.

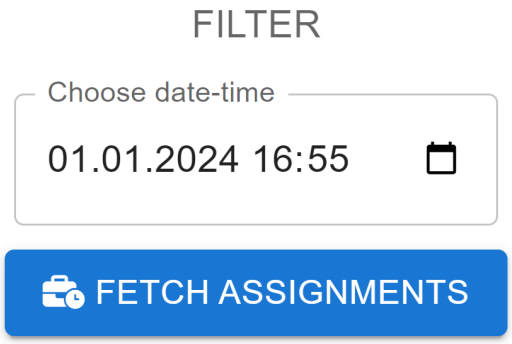


Obrázek 6.5 Informace o přiřazení (tooltip)

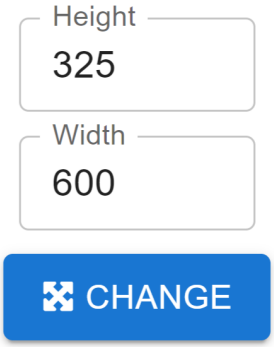
Obrázek 6.6 Informace o přiřazení (box)

6.6.2 Další prvky – filtr a okno adminu

V rámci interaktivních prvků implementovaných prostřednictvím React kódu je dále využita knihovna komponent Material UI (MUI), která nabízí mnoho předpřipravených komponent Reactu, například `Button` (tlačítko), `Card` (karta) nebo `Box` (box). Prostřednictvím těchto komponent je vytvořen filtr (viz obrázek 6.7) a také malé okno adminu, které umožňuje oprávněným uživatelům přímo na stránce zobrazující kancelář změnit velikost kanceláře (viz obrázek 6.8) nebo přidat do kanceláře nová pracovní místa.



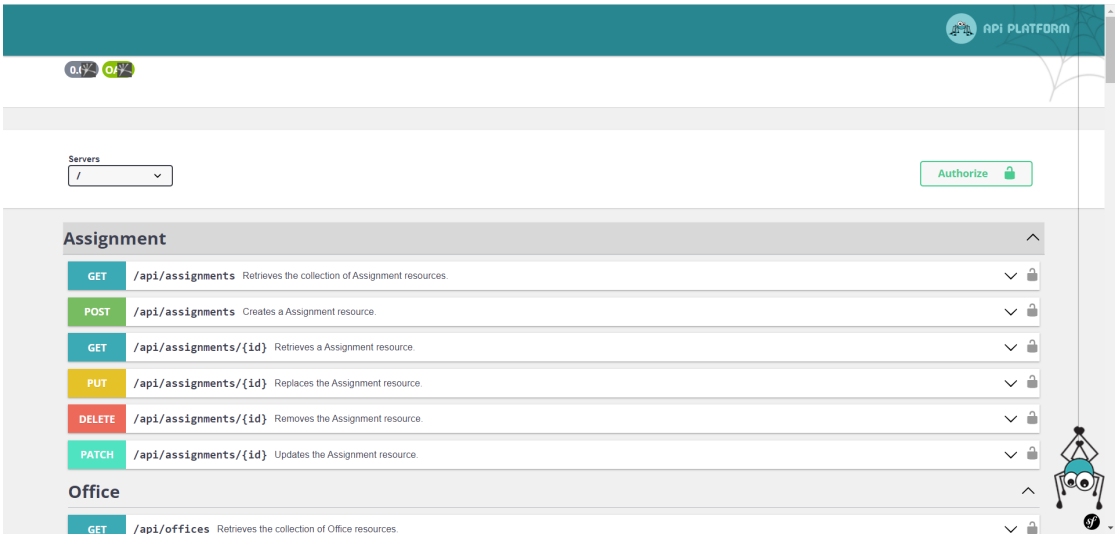
Obrázek 6.7 Filtr pro zobrazení minulých či budoucích přiřazení



Obrázek 6.8 Část adminu na stránce zobrazující kancelář

6.7 API Platform – REST API

Pro vytvoření a správu rozhraní REST API bylo využito frameworku API Platform. Framework byl integrován do webové aplikace Symfony prostřednictvím sady balíčků „api-platform/api-pack“. Po přidání veškerých potřebných závislostí umožňuje API Platform vytvářet koncové body (dále jen endpointy) pro jakékoliv metody (GET, GET COLLECTION, POST, PUT, PATCH, DELETE) REST API požadavků (dále jen requestů) velmi jednoduše prostřednictvím anotací k existujícím entitám a automaticky k těmto vytvořeným API endpointům generovat dokumentaci pomocí OpenAPI specifikace (někdy také označovaná jako Swagger specifikace) viz obrázek 6.9.



Obrázek 6.9 OpenAPI specifikace rozhraní API (generované API Platform anotacemi)

Pomocí anotací je možné k vytvořeným endpointům pro libovolné requesty stanovit také celou řadu parametrů, například omezení práv použití daného requestu viz výpis kódu 7.

```
#[ApiResource(
    operations: [
        new Get(),
        new GetCollection(),
        new Post(security: 'is_granted("ROLE_ADMIN")'),
        new Put(security: 'is_granted("ROLE_ADMIN")'),
        new Patch(security: 'is_granted("ROLE_ADMIN")'),
        new Delete(security: 'is_granted("ROLE_ADMIN")'),
    ],
)]
```

Výpis kódu 7 Ukázka API Platform anotací s omezením práv requestů

Dále API Platform nabízí vytváření libovolných filtrů, které budou prostřednictvím API requestů dostupné, například možnost prostřednictvím API requestu vyhledávat přiřazení (`Assignment`) pro zvoleného zaměstnance (`Person`) identifikovaného pouze částí jeho jména viz výpis kódu 8

```
#[ApiFilter(SearchFilter::class, properties: [
    'seat.office.name' => 'partial',
    'person.name' => 'partial',
    'seat.id' => 'exact',
])]
```

Výpis kódu 8 Ukázka API Platform anotací pro vytvoření filtru

Nechybí ani možnost vytváření skupin (groups) pro normalizaci (čtení, respektive převod objektu na pole) a denormalizaci (zápis, respektive převod pole na objekt). Díky těmto normalizačním a denormalizačním skupinám je možné skrýt některé atributy pro čtení a některé atributy

naopak skrýt pro zápis. Skrytí atributů pro čtení je ve webové aplikaci využito například pro hesla uživatelů a skrytí atributů pro zápis je využito typicky pro veškerá `id` (protože je pro ně využit databázový „auto-increment“, tedy automatické zvyšování `id` pro každý nový záznam v databázi a není tedy žádoucí je manuálně upravovat).

Dále je těchto skupin využito pro propagaci vybraných atributů do requestů jiných entit, které jsou ve vztahu s danou entitou. To je opět snáze pochopitelné na konkrétním případu použití této funkce v aplikaci včetně ukázek:

1. Aplikace nabízí endpoint pro GET request pro získání informací o přiřazení (`Assignment`) ve formátu JSON viz výpis kódu 9.

```
curl -X 'GET' 'http://localhost/api/assignments/301' \
-H 'accept: application/json'
```

Výpis kódu 9 GET request pro získání informací o přiřazení (pomocí nástroje curl)

2. Přiřazení je spojeno skrze databázový vztah se zaměstnancem a pracovním místem, kterého se týká. Standardně bude součástí odpovědi (response) na request informace o zaměstnanci a prac. místě pouze ve formátu IRL (tedy například „/api/people/18“) viz výpis kódu 10.

```
{
  "person": "/api/people/134",
  "seat": "/api/seats/81",
  "fromDate": "2023-12-29T15:00:00+00:00",
  "toDate": "2023-12-29T22:30:00+00:00"
}
```

Výpis kódu 10 JSON response na GET request přiřazení (`Assignment`) bez propagace atributů

3. Při vhodném použití normalizačních skupin je však možné propagovat vybrané atributy zaměstnance do odpovědi (response) na GET request týkající se přiřazení viz výpis kódu 11.

```
{
  "person": {
    "id": 134,
    "firstName": "Adelia",
    "lastName": "Kuhn"
  },
  "seat": {
    "office": {
      "name": "Giruff Solutions B"
    }
  },
  "fromDate": "2023-12-29T15:00:00+00:00",
  "toDate": "2023-12-29T22:30:00+00:00"
}
```

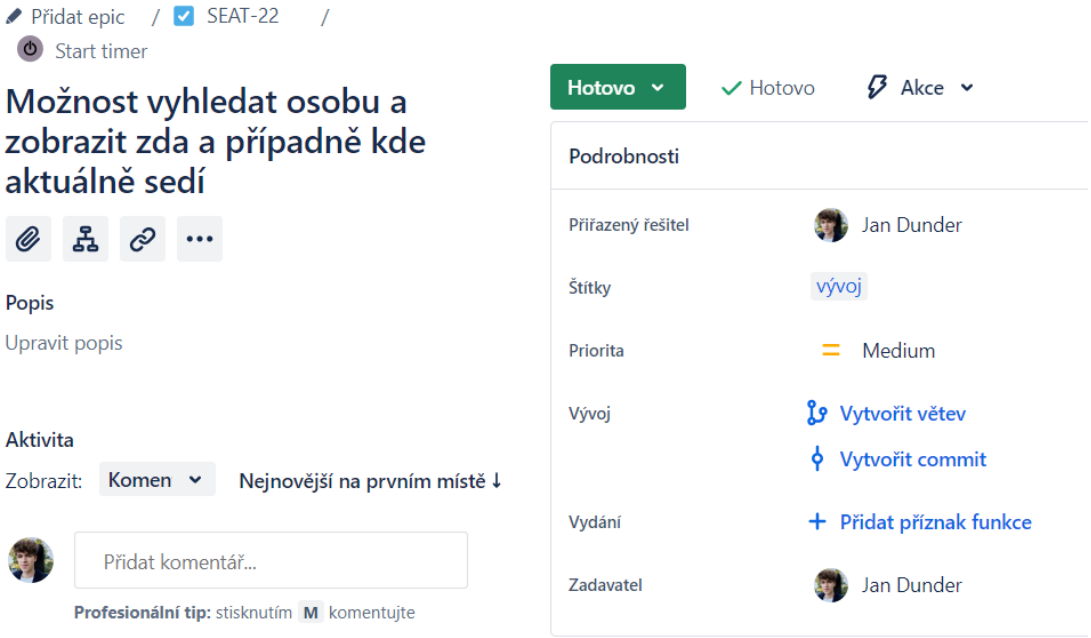
Výpis kódu 11 JSON response na GET request přiřazení (`Assignment`) s propagací atributů

6.8 Řízení úkolů, verzování a kvalita kódu

V rámci implementace webové aplikace byl kladen důraz nejen na samotné psaní kódu, ale také na udržování vysoké úrovně jeho kvality, efektivní verzování a systematický přístup k vývoji. Klíčovou roli v tomto procesu hrály nástroje a postupy, které zajišťovaly efektivní řízení pracovních úkolů a kontinuální kontrolu kvality kódu.

6.8.1 Jira

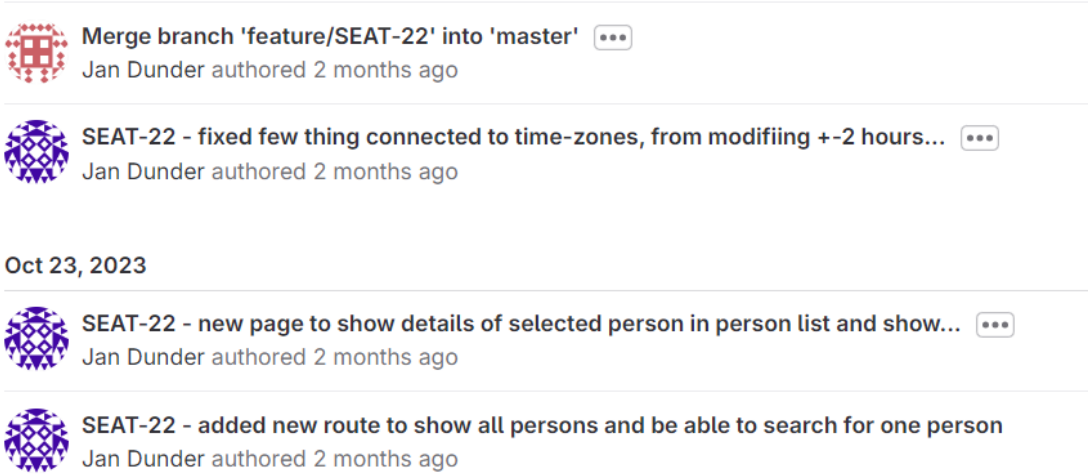
Pro správu a organizaci pracovních úkolů v průběhu vývoje byl využit projekt vytvořený v nástroji Jira. Tento nástroj umožnil převádět analyzované požadavky na jednotlivé úkoly (tickety), což přispělo k jasnému a strukturovanému zápisu úkolů a zefektivnění celého vývojového procesu. Mimo úkolů souvisejících s vývojem webové aplikace byly také zaznamenány některé klíčové úkoly související s celkovou tvorbou této práce (např. úvodní konzultace s vedoucím).



Obrázek 6.10 Ukázka vytvořeného úkolu (ticketu) v nástroji Jira

6.8.2 Git

Pro verzování kódu byl použit systém Git, konkrétně repozitář na fakultním úložišti GitLab dostupný na URL adrese `https://gitlab.fit.cvut.cz/dundejan/seatmaster`. Jednotlivé větve v repozitáři byly pojmenovány podle odpovídajících úkolů (ticketů) v Jira, například pro realizaci úkolu s identifikátorem „SEAT-22“ byla vytvořena větev s názvem „feature/SEAT-22“. Tento přístup zajišťoval jasnou korelaci mezi pracovními úkoly a změnami v kódu, což usnadňovalo revizi a sledování vývoje jednotlivých funkcionalit aplikace. Dodržování konzistentního pojmenování commitů ve stylu „SEAT-22 - *popis změn v angličtině*“ přispělo k ještě lepší orientaci v historii změn po jejich propsání do hlavní větve (merge to master).



Obrázek 6.11 Ukázka commitů a merge do větve master

6.8.3 Kvalita kódu

Důraz byl také kladen na kontrolu kvality kódu. Po dokončení požadavků stanovených v jednotlivých úkolech (ticketech) bylo provedeno manuální otestování implementované funkcionality na lokálně spuštěném webovém serveru a byla provedena kontrola kvality pomocí nástrojů pro automatickou analýzu kódu.

Pro kontrolu kvality PHP kódu byl použit nástroj PHPStan, a to na úrovni (level) 8, což představuje druhou nejvyšší možnou úroveň kontroly. Pro kontrolu kódu JavaScript, konkrétně tedy pro soubory využívající React, byl použit nástroj ESLint. Tato kombinace nástrojů zajišťovala dodržování nejlepších programovacích praktik a pomáhala identifikovat a opravit potenciální problémy v kódu, což je zásadní pro zachování kvality a udržitelnosti vývoje.

```
dundejan:~/seatmaster(master)$ make analyse
Starting Docker container for the connections to database and PHP/Nginx services...
[+] Running 4/0
 ✓ Container seatmaster-database_test-1  Running                                0.0s
 ✓ Container seatmaster-database-1       Running                                0.0s
 ✓ Container seatmaster-php-1            Running                                0.0s
 ✓ Container seatmaster-web-1            Running                                0.0s
Analysing code using PHPStan...
Note: Using configuration file /var/www/symfony/phpstan.neon.
 63/63 [▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓] 100%

 [OK] No errors

Installing Node.js packages...
yarn install v1.22.19
[1/4] Resolving packages...
success Already up-to-date.
Done in 0.21s.
Linting JavaScript files...
No linting errors found.
```

Obrázek 6.12 Ukázka výsledku kontroly kvality kódu pomocí PHPStan a ESLint

6.9 Nasazení a automatizace

6.9.1 Docker

V rámci vývoje aplikace bylo také vytvořeno prostředí (kontejnery) Dockeru, což je klíčový nástroj pro zajištění konzistence a efektivity vývojového procesu. Docker umožňuje vytváření izolovaných kontejnerů pro jednotlivé části aplikace, což značně usnadňuje správu závislostí a konfigurací [36].

Ve vytvořeném souboru `docker-compose.yml` bylo definováno několik služeb, které jsou nezbytné pro běh aplikace. Služba **php** zajišťuje běh PHP pomocí oficiálního PHP obrazu (image) s dodatečnými závislostmi a PHP rozšířeními potřebnými pro aplikaci Symfony, jejich instalace je součástí souboru `Dockerfile`. Aplikace a zdrojový kód jsou sdíleny mezi hostitelským systémem a kontejnerem prostřednictvím Docker svazků (volumes). Služba **web** používá obraz Nginx pro webový server a její konfigurace umožňuje připojení k aplikaci. Pro databázi byly definovány služby **database** a **database_test**, obě využívající obraz Postgres. Tyto služby jsou propojeny v rámci vlastní sítě (docker network) **mynetwork**, což umožňuje efektivní komunikaci mezi nimi.

Tato konfigurace Dockeru umožňuje vývojářům snadné a rychlé nasazení vývojového prostředí s konzistentním nastavením, což je klíčové pro hladký vývoj a testování aplikace.

6.9.2 Makefile

Pro ještě snazší nasazení, testování a vývoj byl použit Makefile, klíčový nástroj pro automatizaci a efektivní správu různých aspektů vývojového procesu. Makefile umožňuje definovat a spouštět různé cíle (make targets) viz obrázek 6.13, což zjednodušuje a urychluje běžné úkoly vývojářů.

```
dundejan:~/seatmaster(master)$ make help
Available targets:
  docker-build  - Build Docker images without using cache.
  docker-up     - Start Docker containers necessary for the development environment.
  compose       - Install PHP dependencies using Composer inside Docker.
  up            - Start the entire development environment, including Docker setup and front-end assets.
  down          - Shut down the development environment and stop Docker containers.
  test          - Run automated tests excluding Panther tests.
  test-panther  - Run Panther tests, requires local PHP setup and appropriate environment configurations.
  php-stan      - Perform PHP static analysis using PHPStan with an increased memory limit.
  yarn-install  - Install Node.js dependencies with Yarn.
  eslint        - Lint JavaScript files.
  analyse       - Run both PHP and JavaScript analysis using PHPStan and ESLint.
  clean         - Clean up generated files and clear Symfony cache.
  rebuild       - Rebuild the entire development environment.
  help          - Display this help message.
```

Obrázek 6.13 Seznam vytvořených cílů Makefile

6.9.3 Readme

Konkrétní postup pro vývoj a správné použití Makefile a Dockeru je popsán v souborech `README.md` (anglická verze) a `README-CZ.md` (česká verze) v kořenovém adresáři repozitáře webové aplikace.

Kapitola 7
Testování

V této kapitole je podrobně popsán strukturovaný přístup k testování webové aplikace, který zahrnuje kromě průběžného manuálního testování také jednotkové, integrační i aplikační testy, a závěrečné uživatelské testování. Toto komplexní testování zajišťuje funkčnost, uživatelský komfort a integritu vyvíjené webové aplikace.

Počátkem vývoje webové aplikace docházelo k radikálnějším změnám kódu a uživatelského zobrazení a proto bylo využíváno především manuální testování skrze lokální webový server. V pozdějších částech vývoje naopak dominovalo automatické testování, protože většina webové aplikace již byla neměnná a bylo pouze nezbytné ověřit, že nově vyvinuté části a drobné úpravy kódu nijak nenarušují funkci již existujících částí webové aplikace.

7.1 Manuální testování

Během celého vývoje byla aplikace manuálně testovaná prostřednictvím lokálně spuštěného webového serveru na němž byla prováděna očekávaná uživatelská interakce. Před připojením nově vyvíjené větve do hlavní větve v repozitáři (merge do masteru) pak byly vždy manuálně otestovány veškeré nové funkce a části kódu, které byly v rámci dané větve vyvinuté. V případě nalezené chyby související s novou větví došlo k okamžité opravě. Pokud byla nalezena chyba nebo nedokonalost s větví přímo nesouvisející, pak byl požadavek na odstranění této chyby přidán do systému Jira k nejvíce relevantnímu úkolu (ticketu).

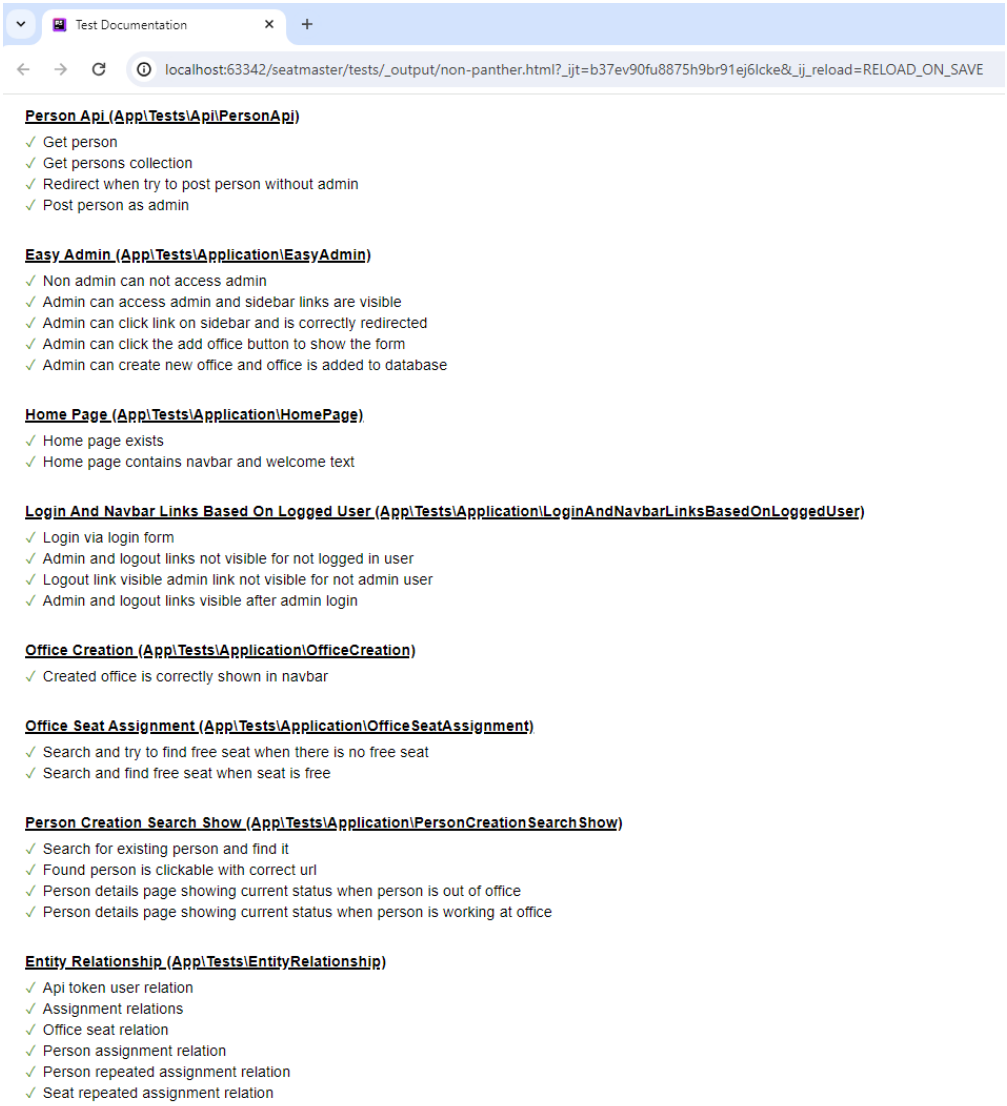
K manuálnímu testování a co nejsnazšímu opravování chyb byl využit především Symfony Profiler, oblíbený nástroj pro vývoj Symfony aplikací, který poskytuje panel nástrojů pro ladění webové aplikace (debug toolbar), Tento nástroj poskytuje detailní informace o provedených požadavcích a odpovědích (request a response), validaci, směrování (routing), bezpečnosti a dalších částech webové aplikace.

7.2 Automatické testování

Pro automatické testování využívá aplikace PHPUnit integrovaný s testovacími třídami `WebTestCase` a `KernelTestCase` od frameworku Symfony, což vytváří robustní prostředí pro různé testovací scénáře. Toto prostředí je klíčové pro ověřování funkčnosti aplikace v různých modulech a interakcích s uživateli. Pro automatické testování je vytvořena testovací databáze, aby bylo zaručeno, že automatické testování neohrozí a neovlivní vývojovou či produkční databázi webové aplikace, a také naopak – tedy, že výsledek automatického testování nebude změnami v těchto databázích ovlivněn.

Hlavní výzvou bylo především psaní testů, které simulují reálné uživatelské scénáře a vyžadují tak například existenci uživatele s administrátorskými právy, jeho přihlášení se, vstup do administrátorského rozhraní a teprve poté provedení vybraných akcí v administrátorském rozhraní. Pokud jde o nalezené a opravované chyby, nejkritičtější bylo testování kolizí jednorázových a opakovaných přiřazení zaměstnanců k pracovním místům. Testování kolizí si vyžádalo opakované zdokonalování kódu po mnoha nalezených nepřesnostech způsobených především chybnou prací s časovými pásmy.

Pro snadné vykonání automatického testování je možné využít „make targets“, tedy cíle souboru `Makefile`, které spustí automatické testování a výstup tohoto testování zobrazí jak v terminálu, tak i v HTML formátu prostřednictvím souboru vygenerovaného TestDox generátorem viz obrázek 7.1.



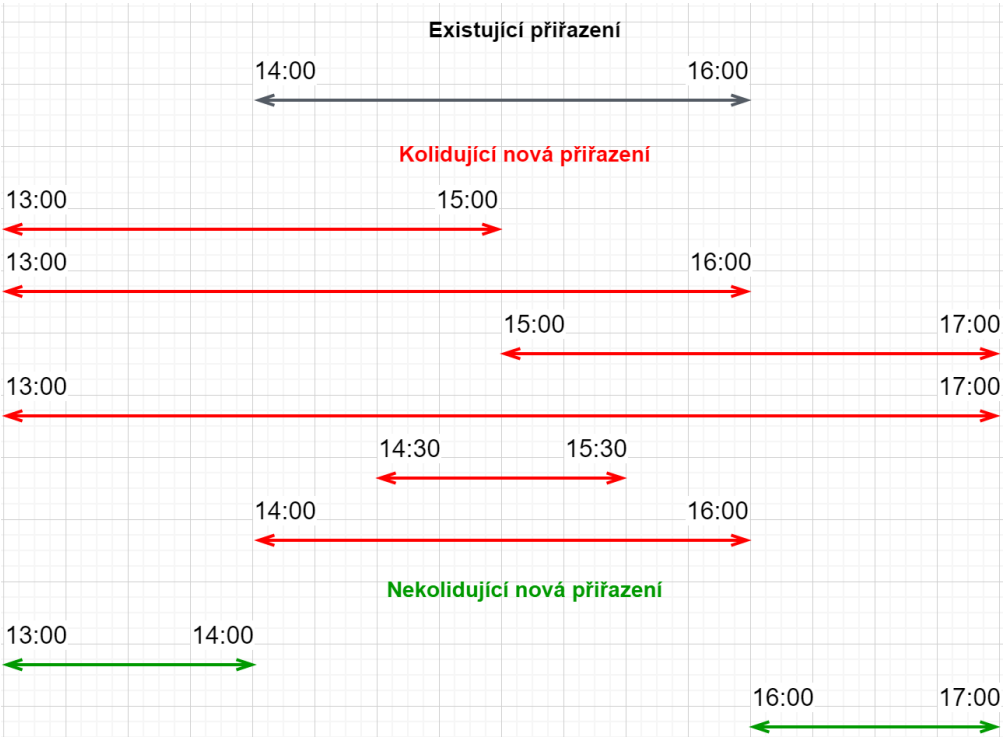
Obrázek 7.1 Část HTML výstupu testů zobrazená ve webovém prohlížeči

7.2.1 Jednotkové testy

Jednotkové testy se zaměřují na izolované komponenty a zajišťují, aby každá z nich fungovala správně sama o sobě.

7.2.1.1 Testy detekce kolizí

Tyto testy hodnotí schopnost aplikace zpracovávat a detekovat kolize v přiřazeních. Jak již bylo popsáno v podkapitolách věnující se validaci, je třeba detekovat kolize nejen samostatně mezi jinými jednorázovými a opakujícími se přiřazeními, ale také napříč mezi nimi (tedy validace jednorázových vůči opakovaným a naopak). Ověřují algoritmus detekce kolizí pomocí specifických datových sad a zajišťují přesné zpracování překrývajících se a nepřekrývajících se přiřazení, které jsou v souladu s pravidly pro detekci kolizí. Obrázek 7.2 ilustruje přehled různých přiřazení, která je třeba testovat pro ověření správné detekce kolizí vůči jednomu již existujícímu přiřazení.



Obrázek 7.2 Přehled kolidujících a nekolidujících přiřazení

7.2.2 Integrační testy

Integrační testy slouží pro otestování správné interakce a vazby mezi různými součástmi aplikace. Ve vyvíjené webové aplikaci do této kategorie spadají pouze testy správně definovaných vztahů mezi entitami a jejich atributy.

7.2.2.1 Testy vztahů entit

Tyto testy ověřují vztahy a správné mapování mezi různými entitami, například vztah mezi osobami a přiřazeními nebo uživateli a API tokeny. Zajišťují, aby schéma databáze a vztahy entit správně fungovaly při integraci s ostatními částmi aplikace.

7.2.3 Aplikační testy

Aplikační testy zkoumají celkovou funkčnost aplikace a odrážejí interakce s koncovými uživateli. Některé z testů kopírují přímo uživatelské scénáře, například vyhledání osoby a z detailní stránky zjištění informací o dané osobě včetně aktuálního místa, kde tato osoba sedí viz testy 7.2.3.6.

7.2.3.1 Testy domovské stránky

Tato část ověřuje existenci domovské stránky a její obsah (uvítací text a navigační panel).

7.2.3.2 Testy přihlašování uživatelů a navigačních odkazů

Testy přihlašování a navigace kontrolují proces přihlášení uživatelů prostřednictvím přihlašovacího formuláře a viditelnost odkazů na navigačním panelu na základě rolí uživatelů, tedy pouze uživatel s rolí administrátora vidí v navigačním panelu odkaz vedoucí na admin. rozhraní.

7.2.3.3 Testy administrátorského rozhraní EasyAdmin

V těchto testech dochází ke kontrole funkcí administrátorského rozhraní. Zajišťují, že do administrátorského rozhraní má přístup pouze oprávněný uživatel a ten může v administrátorském rozhraní provádět veškeré CRUD operace, což je testováno na vytváření kanceláří prostřednictvím formulářů v administrátorském rozhraní.

7.2.3.4 Testy zobrazení kanceláří v navigačním panelu

Tento testovací segment zajišťuje správnou funkčnost navigačního panelu po vytvoření kanceláře. Tedy dochází ke kontrole, že každá existující kancelář v databázi je správně zobrazena v rozbalovací nabídce navigačního panelu.

7.2.3.5 Testy přidělování míst v kanceláři

Tyto testy ověřují proces přidělování míst, zejména ve scénářích s různou dostupností míst.

7.2.3.6 Testy vyhledávání a zobrazování osob a detailů

Klíčovou funkcí těchto testů je potvrzení funkčností souvisejících s osobami (zaměstnanci), tedy tyto testy potvrzují možnost vyhledat existujícího zaměstnance prostřednictvím vyhledávací lišty a následně zobrazit detailní informace o tomto zaměstnanci včetně jeho aktuálního umístění (tedy místo, na kterém v aktuálním čase zaměstnanec sedí).

7.2.3.7 Test vykreslení Reactu

Tento test ověřuje, že webová aplikace vykresluje součástky Reactu, které by měly být přítomny ve webové aplikaci na stránce, která zobrazuje detail kanceláře. Jedná se tedy spíše o test správné konfigurace Reactu. Testo test využívá knihovnu Panther, která využívá k testování webový prohlížeč a scrapování.

7.2.4 Testy API

Tyto testy ověřují koncové body (endpointy) API rozhraní vytvořeného pomocí API Platform. Konkrétně jsou testovány endpointy rozhraní API pro osoby (zaměstnance). Součástí testů je i ověření práv přístupu k různým endpointům na základě práv uživatele, který k API endpointům přistupuje, např. pouze administrátor má možnost přistoupit k POST requestům.

7.3 Uživatelské testování

Uživatelské testování je klíčovou fází vývoje každé aplikace. Jeho hlavním účelem je ověřit použitelnost, funkčnost a celkový uživatelský zážitek, který aplikace nabízí. Tato podkapitola popisuje proces uživatelského testování, které bylo realizováno na konci vývoje webové aplikace.

7.3.1 Struktura testování – scénáře

Proces testování je strukturován do dvou odlišných kategorií, které odrážejí různé úrovně uživatelského přístupu v rámci aplikace, konkrétně se jedná o základní uživatelský přístup (běžní zaměstnanci) a administrátorský přístup (manažeři a vedoucí). Pro obě tyto kategorie byly vytvořeny odlišné scénáře, které obsahují jak jednoduché a konkrétní úkoly, tak i komplexnější úkoly obsahující příběh (story), které přímo vycházejí z identifikovaných procesů (viz podkapitola 4.2) a ověřují, zda je aplikace ve stavu, který umožňuje hladký průběh navržených budoucích (TO-BE) stavů těchto procesů.

7.3.1.1 Scénáře základního uživatelského přístupu

A. Zobrazení kanceláře a vnímání údajů, zobrazení budoucí obsazenosti

- Zobraz si kancelář „Snayke Business C“. Kolik je aktuálně obsazených a kolik prázdných míst?
- Kdo momentálně sedí na místě číslo 74 a do kdy na tomto místě bude sedět?
- Zobraz si kancelář „Zebrox Workspace C“. Kolik bude obsazených a kolik prázdných míst dnes v 22.00?

B. Vyhledání zaměstnance a zobrazení jeho detailů

- Najdi zaměstnance, o kterém víš pouze příjmení „Barrows“ a pracuje na pozici „Library Technician“. Sděl jeho nebo její křestní jméno.

C. Přiřazení pravidelného místa novému zaměstnanci (proces)

- Nastupuješ jako brigádník do nové firmy a dostal si za úkol zajistit si místo v kanceláři. Budeš pracovat pouze jeden den v týdnu – v pondělí od 14.00 do 22.00. Rád bys byl v kanceláři „Giruff Venture A“. Už si podepsal smlouvu, ale nastupuješ až v dubnu tohoto roku. Tvým úkolem je ve firemní webové aplikaci najít vhodné pracovní místo (/místa) a sdělit jeho číslo (/čísla).

D. Konzultace problému s kolegou (proces)

- Jako nový zaměstnanec firmy ses brzy dostal do nesnází a potřebuješ pomoc zkušenějšího kolegy. Od manažera si dostal informaci, že s problémy se můžeš obracet na jednoho ze sourozenců s příjmením „Nolan“, ale nevíš, zda je alespoň jeden z nich nyní v práci a kde případně sedí. Zjisti to.

E. Možnost práce mimo běžnou pracovní dobu (proces)

- Toto pondělí si byl na koncertě a z práce ses omluvil. Manažer ti dal možnost si práci nahradit jiný den. Rád bys tedy přišel pracovat tento pátek od 10.00 do 18.00. Nejprve však musíš zjistit, zda je v tuto dobu nějaké místo volné a které. Ideálně bys rád seděl v „Zebrox Workspace A“, protože tam bude tvůj známý.

7.3.1.2 Scénáře administrátorského přístupu

A. Přihlášení a statistiky v administrátorském prostředí

- Přihlas se do aplikace s uvedenými údaji – email: admin@example.com, heslo: admin.
- Otevři admin a z úvodní stránky urči, kolik osob celkem ve firmě pracuje a kolik je nyní v kanceláři.
- Zobraz si statistiky kanceláře „Snayke Business C“ a z grafu urči, kolik osob v této kanceláři bude dnes v 20.00.

B. Admin CRU(D) operace

- Přidej novou kancelář s názvem „Podkroví“ s rozměry 400 x 400, přidej do této kanceláře jedno nové pracovní místo. Přidej sám sebe jako nového zaměstnance (osobu), vyplň jen povinné údaje. Následně uprav tohoto zaměstnance a změň pracovní pozici na „manažer“.

C. Interaktivita kanceláře

- Odejdi z adminu, zobraz si kancelář s názvem „Podkroví“. Posuň pracovní místo na jinou pozici a zmenši kancelář.

D. Admin (CRU)D operace

- Vstup opět do adminu a smaž veškeré objekty, které si vytvořil (kancelář, pracovní místo a zaměstnanec)

E. Přiřazení pravidelného místa novému zaměstnanci (proces)

- Jsi manažerem firmy a nový brigádník „Jody Weber“ tě požádal o pravidelné přiřazení místa 108 každé pondělí od 14.00 do 22.00. Přiřaď mu dané místo.

F. Možnost práce mimo běžnou pracovní dobu

- Jsi manažerem firmy a nový brigádník „Jody Weber“ tě požádal o možnost práce mimo svou běžnou pracovní dobu. Tento pátek by rád pracoval od 10.00 do 18.00 na místě 107. Přiřaď mu dané místo.

7.3.2 Metodika

7.3.2.1 Účastníci

Uživatelského testování se zúčastnilo celkem 5 osob a všechny tyto osoby se účasnili jak testování základního uživatelského přístupu, tak i testování administrátorského přístupu. Uživatelé byli vybráni tak, aby poskytovali co možná nejvíce různorodou skupinu, pokud jde o znalost práce s PC (příp. obdobnými aplikacemi) a znalost angličtiny. Testování jsem se účastnil i já, jakožto vývojář aplikace, pro poskytnutí referenčních časů dokončení úkolů a tedy snazší identifikaci problémů s použitelností aplikace.

Jména účastníků jsou anonymizována a k účastníkům je referováno prostřednictvím jejich iniciál. Údaje A1, B2 a C1 v závorce označují úroveň angličtiny testujících osob vyjádřenou pomocí členění dle CEFR (Společný evropský referenční rámec pro jazyky) [37]. Účastník ED, který disponuje pouze velmi základní znalostí angličtiny odpovídající úrovni A1, bude před použitím aplikace a opětovně také před konkrétními úkoly seznámen s potřebnými pojmy tak, aby pro něj bylo reálné úkoly splnit, ale přesto byla zachována jeho ztížená schopnost orientaci v aplikaci kvůli jazykové bariéře. Všem účastníkům bude v případě dotazu na význam konkrétního anglického slova či slovního spojení pravdivě odpovězeno.

Testování se účasnili následující osoby:

- **JD** – vývojář, absolutní znalost aplikace
- **OL** – student IT, profesionální znalost práce s PC včetně zkušenosti s administrací, pokročilá znalost angličtiny (B2)
- **VP** – student, dobrá znalost práce s PC, profesionální znalost angličtiny (C1)
- **OD** – dospělý pod 30 let, dobrá znalost práce s PC, pokročilá znalost angličtiny (B2)
- **ED** – dospělý nad 50 let, základní znalost práce s PC, základní znalost angličtiny (A1)

7.3.2.2 Postup

Každý účastník provede (nebo se pokusí provést) veškeré úkoly, které jsou součástí vytvořených scénářů (viz podkapitola 7.3.1). Nejprve bude jednou přečteno celé zadání úkolu a toto zadání bude poskytnuto také ve vytištěné podobě. Následně bude spuštěn čas a účastník bude vyzván k provedení úkolu. Po zdárném provedení úkolu bude čas zastaven a zaznamenán.

Během provádění úkolů bude každý účastník testování pozorován. Pokud účastník provede interakci, která není nezbytná k dokončení úkolu, tato nadbytečná interakce bude zaznamenána. Pokud se účastník v aplikaci ztratí a nebude schopen úkol dokončit, bude mu napovězeno a následně s ním bude úkol krátce diskutován, aby došlo k porozumění, jaká část aplikace problém s dokončením úkolu pravděpodobně způsobila. Zároveň bude po každém splnění úkolu účastník vyzván k reakci, zda pro něj byl tento úkol snadný a pochopitelný nebo zda ho v průběhu úkolu například něco zmátlo a činilo mu problém pochopit.

7.3.3 Výsledky a diskuse

Téměř všechny scénáře a úkoly proběhly bez jakýchkoliv komplikací co se týče funkčnosti aplikace i schopnosti rychlé orientace v uživatelském i administrátorském prostředí aplikace. Za nedostatečný lze považovat pouze jeden úkol ze základního uživatelského scénáře, který je podrobněji rozebrán v následující podkapitole 7.3.3.1. Uživatelské testování zároveň přineslo několik zajímavých podnětů ze strany orientace (viz podkapitola 7.3.3.2), které si při případném dalším vývoji aplikace nepochybně zaslouží podrobnější rozbor, na jehož základě bude možné aplikaci zdokonalit tak, aby lépe vyhovovala potřebám svých uživatelů, a to jak na základní uživatelské, tak i na administrativní úrovni. Ukázka záznamu o testování viz obrázek 7.3.

7.3.3.1 Funkčnost

Z hlediska možnosti správného splnění byl nalezen pouze jeden problematický úkol a to úkol **C. Přiřazení pravidelného místa novému zaměstnanci (proces)** ze scénáře základního uživatelského přístupu. Zadáním tohoto úkolu bylo nalézt pomocí aplikace pracovní místo, které bude volné každé pondělí v zadaném časovém rozpětí počínaje stanoveným měsícem. Aplikace ovšem nabízí pouze možnost vyhledání volných pracovních míst pro stanovené rozpětí dnů a časů, tedy je možné zkontrolovat vždy jen jedno vybrané pondělí v zadaném čase. Pro naprostou jistotu nalezení správného místa a tedy splnění úkolu by tak musel uživatel postupně zkontrolovat všechny pondělky v zadaném časovém rozpětí počínaje stanoveným měsícem. Lze předpokládat, že kontrola např. náhodných 4 pondělí bude dostatečná a ve většině případů správná, nicméně není možné na toto absolutně spoléhat.

Pro opravení tohoto úkolu by tedy bylo třeba do aplikace implementovat ještě druhý způsob vyhledávání, který by umožnil filtrování volných míst pro týdně se opakující časové rozpětí včetně specifikace počátečního a koncového data těchto opakování. Logika stojící za tímto způsobem vyhledávání je již implementována a slouží k validaci kolizí, nicméně není poskytnuta navenek prostřednictvím uživatelského rozhraní.

7.3.3.2 Orientace

Pokud jde o schopnost orientace, ze zaznamenaných časů nelze vybrat žádné konkrétní úkoly, se kterými by účastníci měli problémy z hlediska obtížné orientace a tedy delšího času. Všichni účastníci testování však poskytli k uživatelskému rozhraní aplikace cenné podněty ať už svým postupem a dotazy při plnění úkolu nebo svými připomínkami po vykonání úkolu. Ač se připomínky napříč účastníky zpravidla neopakovaly vícekrát, jedná se o důležitou zpětnou vazbu, kterou je vhodné vzít v potaz a při případném dalším vývoji aplikace se na ni zaměřit. Následuje seznam připomínek a podnětů, které byly během testování zaznamenány včetně zmínění osoby, která je autorem tohoto podnětu:

- OD zaměnil úkol opustit administrátorské rozhraní za odhlášení se z aplikace.
- VP byl nejprve zmaten z pojmu *Assignment* pro přiřazení zaměstnanců k pracovním místům a domníval se, že se jedná o úkoly pro zaměstnance, což je další možná interpretace tohoto slova.
- OL i ED by ocenili při použití filtru pro zobrazení minulého či budoucího obsazení kanceláře, aby tlačítko pro aplikování filtru bylo i součástí otevíracího panelu pro výběr data a času viz obrázek 7.4.
- OL by uvítal možnost tmavého režimu (tzv. „dark mode“) v rámci celé aplikace. Aktuálně je tmavý režim nabízen pouze v administrátorském rozhraní viz obrázek 7.5.
- OL by uvítal možnost prokliku z administrátorského rozhraní na zobrazení kanceláří.
- OD nejprve hledal volná místa pomocí filtrování budoucí obsazenosti kanceláří a ne prostřednictvím stránky „Free seats“, která je zobrazena na hlavní navigační liště. Sám se však opravil a danou stránku správně použil.
- ED chtěl scrollovat obsah stránky v momentě, kdy se kurzorem myši nacházel v grafu a scrollování tak způsobovalo přibližování a oddalování grafu namísto posunu stránky.

Některé výše zmíněné orientační problémy byly způsobeny pouze snahou účastníků splnit úkol rychle a účastníci logicky preferovali stránky, které již viděli a použili před zcela novými stránkami. Například, OD byl nejprve prostřednictvím úkolů seznámen s plány kanceláře, které také zobrazují volná místa, proto měl následně tendenci hledat volná místa prostřednictvím plánů kanceláří a ne prostřednictvím doposud pro něj neznámé stránky „Free seats“. Nicméně rychle si uvědomil, že hledání volných místa skrze plány kanceláří by bylo časově náročné a sám se nakonec opravil a rozhodl se přejít na stránku „Free seats“.

Oproti tomu jiné orientační problémy a podněty si nepochybně zaslouží důkladnější rozbor, kupříkladu zmatení z pojmu *Assignment* vede k nutné úvaze, zda nezvolit pro přiřazení jiný pojem (např. *Booking* či *Placement*).

7.4 Shrnutí výsledků testování

Úspěšné provedení testů svědčí o vysoké míře funkčnosti a spolehlivosti celé aplikace. Jednotkové testy potvrzují robustnost jednotlivých komponent, zatímco integrační a aplikační testy prokazují správné propojení jednotlivých částí aplikace a korektní fungování webové aplikace jako celku.

Závěrečné uživatelské testování pak přímo na potenciálních uživatelích aplikace napříč různými znalostmi práce s počítačem a anglického jazyka prokázalo splnění všech funkčních požadavků a tedy i splnitelnost navržených budoucích (TO-BE) stavů identifikovaných procesů, vyjma nedokonalé možnosti filtrování volných míst související s procesem „Přiřazení pravidelného místa novému zaměstnanci“ viz podkapitola 7.3.3.1.

Manuální i automatické testování bylo při vývoji naprosto nezbytné a bez jeho použití by nebylo možné dosáhnout výsledné spolehlivosti a uživatelské přívětivosti webové aplikace.

Uživatelské testování – běžný uživatel

Základní úkony

Zobrazení kanceláře a vnímání údajů, zobrazení budoucí obsazenosti

Zobraz si kancelář „Snayke Business C“. Kolik je aktuálně obsazených a kolik prázdných míst?

RESPONDENT	ODPOVĚĎ	ČAS	POZNÁMKY
JD (vývojář)	✓	10 s	
VP	✓	23 s	
OL	✓	16 s	
OD	✓	16 s	
ED	✓	14 s	

Kdo momentálně sedí na místě číslo 74 a do kdy na tomto místě bude sedět?

RESPONDENT	ODPOVĚĎ	ČAS	POZNÁMKY
JD (vývojář)	✓	6 s	
VP	✓	4 s	
OL	✓	9 s	
OD	✓	11 s	
ED	✓	10 s	

Zobraz si kancelář „Zebrox Workspace C“. Kolik bude obsazených a kolik prázdných míst dnes v 22.00?

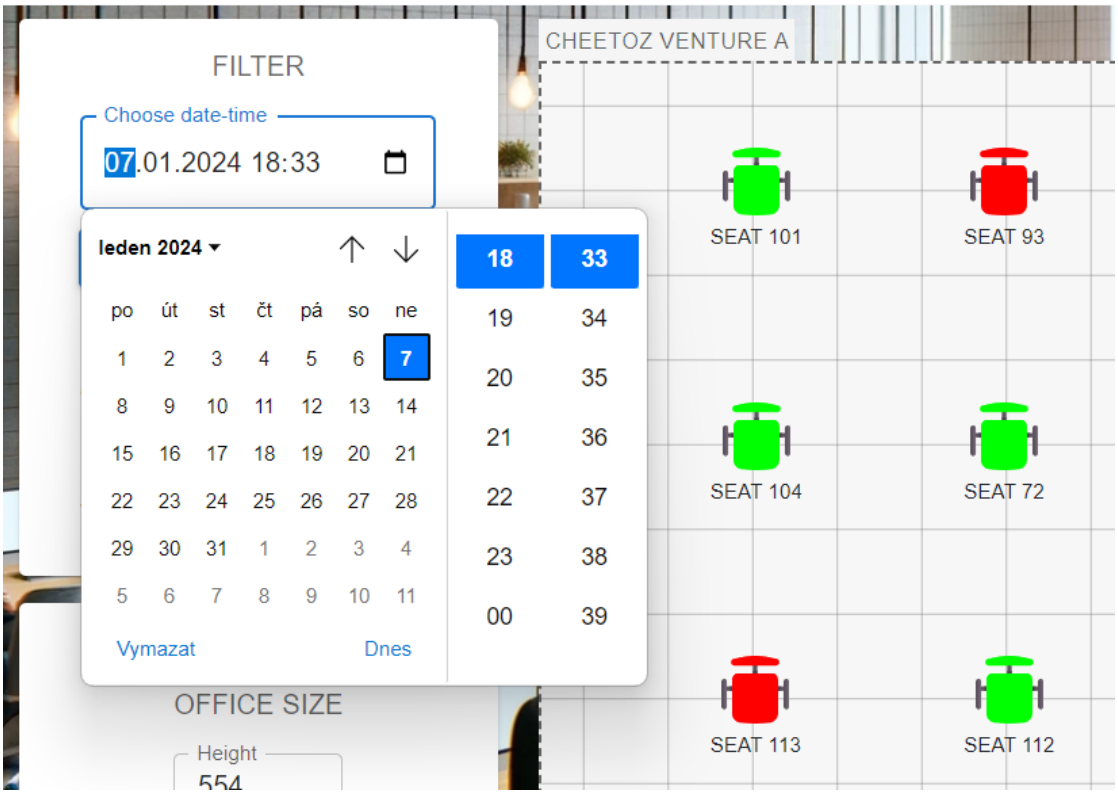
RESPONDENT	ODPOVĚĎ	ČAS	POZNÁMKY
JD (vývojář)	✓	17 s	
VP	✓	35 s	
OL	✓	30 s	TLAČÍTKO „FETCH ASSIGNMENTS“ -NÁVRH PŘIDÁVÁNÍ DO WIDGETU
OD	✓	22 s	
ED	✓	61 s	TLAČÍTKO „FETCH ASSIGNMENTS“ -NEJASNOST, ZDA JE TŘEBA JEJ POUŽÍT

Vyhledání zaměstnance a zobrazení jeho detailů

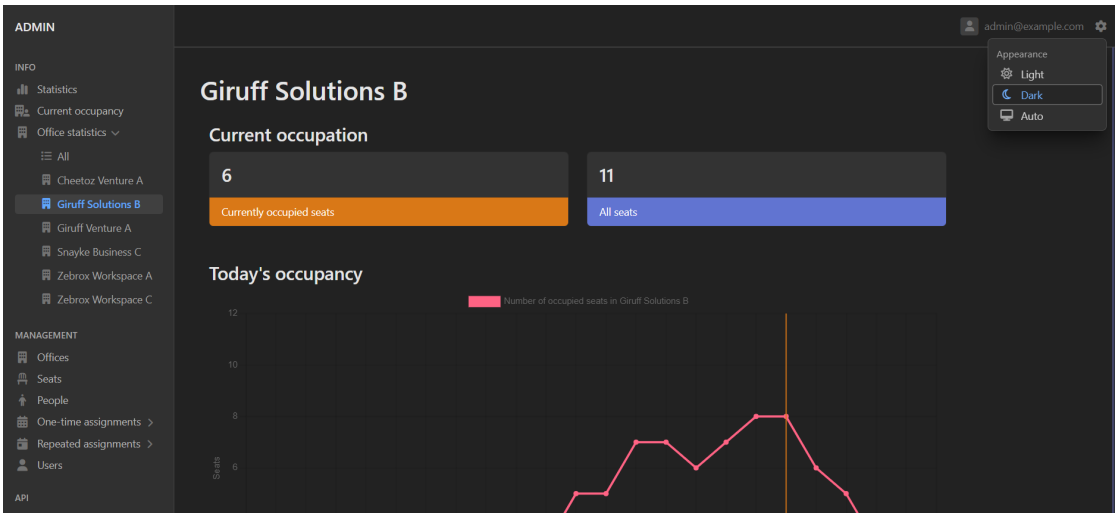
Najdi zaměstnance, o kterém víš pouze příjmení „Barrows“ a pracuje na pozici „Library Technician“. Sděl jeho nebo její křestní jméno.

RESPONDENT	ODPOVĚĎ	ČAS	POZNÁMKY
JD (vývojář)	✓	10 s	
VP	✓	12 s	
OL	✓	11 s	
OD	✓	15 s	
ED	✓	23 s	hledáno pomocí scrollování ne pomocí vyhl. pole

Obrázek 7.3 Scan části vyplněného záznamu o testování



Obrázek 7.4 Otevírací panel (widget) pro výběr data a času



Obrázek 7.5 Tmavý režim v administrátorském rozhraní

Kapitola 8
Zhodnocení a závěr

V rámci této bakalářské práce byl úspěšně navržen a implementován systém pro efektivní správu pracovních míst v kancelářích. Hlavní cíle práce, které zahrnovaly podrobnou analýzu stávajících řešení, návrh a vývoj aplikace a její komplexní testování, byly úspěšně splněny. Aplikace demonstruje schopnost efektivně řešit problematiku správy pracovních prostorů ve firmách.

Počáteční rešerše teoretických základů (technologií) a existujících aplikací včetně jejich analýzy poskytla cenné vhledy, které byly využity při následující identifikaci a analýze klíčových firemních procesů souvisejících se správou pracovních míst v kancelářích a při návrhu jejich možné optimalizace.

Na základě navrženého budoucího stavu procesů a s nimi souvisejících popsaných požadavků na aplikaci byl vytvořen návrh webové aplikace, popisující, jakým způsobem budou jednotlivé technologie použity a k jakému účelu budou sloužit. Součástí návrhu aplikace bylo i vytvoření databázového modelu a rozmyšlení, jakým způsobem budou řešeny očekávané problémy a výzvy, které bude implementace přinášet, konkrétně ošetření času, validace a zobrazení interaktivních plánů kanceláří.

Aplikace byla následně implementována a otestována za dodržení řádného řízení projektu, verzování, dodržení programovacích konvencí a udržení vysoké kvality a čitelnosti kódu. Z testování vyplývá, že aplikace byla implementována tak, že je přehledná, spolehlivá a plní klíčové požadavky na aplikaci.

Přínosem této práce je především samotná výsledná implementace webové aplikace pro obsazování míst v kancelářích v kombinaci s rešerší a analýzou existujících podobných aplikací. Jak z analýzy těchto aplikací vyplývá, veškerá existující vyhovující řešení jsou cenově velmi nákladná a neexistuje žádné, které by bylo bezplatné, volně dostupné a nabízelo funkce, které poskytuje aplikace vytvořená v rámci této práce, tedy možnost přiřazování zaměstnanců k pracovním místům, vizualizace obsazenosti míst, interaktivní plány kanceláří, statistiky obsazenosti a vytíženosti kanceláří a také rozhraní API pro snadnou integraci mezi případné další firemní aplikace. V případě budoucího uvolnění zdrojového kódu pro volné užití (např. formou open-source) by tak aplikace mohla být inovativním bezplatným řešením v této oblasti pro mnoho firem. Ke zvážení je samozřejmě i případné zpoplatnění aplikace, avšak nutno uznat, že i přes implementaci požadovaných funkcí a splnění definovaných požadavků na aplikaci by vyvinutá webová aplikace v konkurenci robustních řešení vyvíjených početnými a zkušenými týmy vývojářů neobstála.

Na tomto místě je třeba zmínit, že v rámci praktické realizace byly některé sekundární funkce, uvedené jako ideální, ale ne nezbytné, vynechány. Mezi takové funkce patří například možnost přidání dalších objektů do plánů kanceláří, jako jsou stoly, monitory a další vybavení související s pracovními místy. Také uživatelské testování odhalilo jeden funkční nedostatek v možnosti vyhledávání volných míst pro opakující se časové rozpětí a též poskytlo cennou zpětnou vazbu a pomohlo tak nastínit oblasti pro možná budoucí vylepšení.

V případném budoucím vývoji je možné tyto oblasti podrobněji identifikovat a následně vylepšit. Zásadním vylepšením by bylo nepochybně rozšíření funkčnosti aplikace o možnost vyhledávání volných míst pro opakující se termíny a rozšíření interaktivních plánů kanceláří o další objekty. Dále by mohlo dojít k opětovné revizi procesů a odhalení dalšího prostoru pro vylepšení. Jedním z těchto vylepšení by mohla být například implementace systému pro podávání žádostí o pracovní místa přímo v aplikaci. To by výrazně usnadnilo komunikaci mezi zaměstnanci a manažery při schvalování žádostí o přiřazení. Další možný vývoj zahrnuje využití algoritmů pro optimalizaci přiřazení pracovních míst, což by mohlo zvýšit efektivitu využití kancelářských prostorů, zejména ve velkých organizacích, kde je důležité optimalizovat rozmístění zaměstnanců, třeba i s ohledem na přiřazené projekty. V neposlední řadě je pak třeba připomenout, že webová aplikace je zafixovaná pro konkrétní časové pásmo a budoucí rozvoj by tak mohl přinést také možnost volby zobrazení a ukládání pro různá časová pásma.

Závěrem lze konstatovat, že přestože aplikace nedosahuje kvalit současných nejlepších existujících řešení, tak splňuje klíčové požadavky, kladené na aplikace pro obsazování míst v kancelářích, a její současný stav poskytuje pevný základ pro řízení kancelářských prostorů a nabízí mnoho možností pro další rozvoj a inovace.

Bibliografie

1. *Best Desk Booking Software* [online]. G2.com, 2023 [cit. 2023-11-18]. Dostupné z: https://www.g2.com/categories/desk-booking.
2. *Space Management Software* [online]. Capterra Inc., 2023 [cit. 2023-11-18]. Dostupné z: https://www.capterra.com/space-management-software/.
3. BEHNKE, Kim. *40 Best Hot Desk Booking Software of 2023 for Hybrid Teams* [online]. People Managing People, 2023 [cit. 2023-11-18]. Dostupné z: https://peoplemanagingpeople.com/tools/best-hot-desk-booking-software/.
4. *Our plans* [online]. deskbird, 2023 [cit. 2023-11-18]. Dostupné z: https://www.deskbird.com/pricing.
5. *Transparent pricing. No hidden costs* [online]. Yarooms International, 2023 [cit. 2023-11-18]. Dostupné z: https://www.yarooms.com/pricing.
6. *The right price for companies of all sizes* [online]. Nspace, 2023 [cit. 2023-11-18]. Dostupné z: https://www.getnspace.com/pricing.
7. *Tactic Features* [online]. G2.com, 2023 [cit. 2023-11-18]. Dostupné z: https://www.g2.com/products/tactic/features.
8. *The #1 workplace software for hybrid teams* [online]. Tactic, 2023 [cit. 2023-11-18]. Dostupné z: https://www.gettactic.com/why-tactic/.
9. *Customizable plans for a flexible workplace* [online]. Tactic, 2023 [cit. 2023-11-18]. Dostupné z: https://www.gettactic.com/pricing-for-workplace-management-solutions/.
10. *Connecting people, spaces, and data to power the places where people work best together* [online]. Envoy, 2023 [cit. 2023-12-28]. Dostupné z: https://envoy.com/.
11. *The right price for any workplace* [online]. Envoy, 2023 [cit. 2023-11-18]. Dostupné z: https://envoy.com/pricing?tab=workplace-tab.
12. *HTML basics* [online]. Mozilla, 2023 [cit. 2023-11-14]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics.
13. *What is HTML: Common uses and defining features* [online]. Codecademy, 2021 [cit. 2023-11-14]. Dostupné z: https://www.codecademy.com/resources/blog/what-is-html/.
14. *Twig for Template Designers* [online]. SensioLabs, 2023 [cit. 2023-11-14]. Dostupné z: https://twig.symfony.com/doc/3.x/templates.html.
15. *What is CSS?* [online]. Mozilla, 2023 [cit. 2023-11-14]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps/What_is_CSS.

16. GUDELIAUSKAS, Domantas. *What Is CSS and How Does It Work?* [online]. Hostinger, 2023 [cit. 2023-11-15]. Dostupné z: https://www.hostinger.com/tutorials/what-is-css.

17. *What is PHP?* [online]. The PHP Group, 2023 [cit. 2023-11-15]. Dostupné z: https://www.php.net/manual/en/intro-whatis.php.

18. LERDORF, Rasmus. *Believe the hype: PHP founder backs Facebook's HipHop technology* [online]. Paul Krill. InfoWorld, 2013 [cit. 2023-11-15]. Dostupné z: https://www.infoworld.com/article/2609877/believe-the-hype--php-founder-backs-facebook-s-hiphop-technology.html.

19. *General Installation Considerations* [online]. The PHP Group, 2023 [cit. 2023-11-15]. Dostupné z: https://www.php.net/manual/en/install.general.php.

20. SAROSA, Astari Pinasthika. *What Is PHP? Learning All About the Scripting Language* [online]. Hostinger, 2023 [cit. 2023-11-15]. Dostupné z: https://www.hostinger.com/tutorials/what-is-php/.

21. *Introduction* [online]. SensioLabs, 2023 [cit. 2023-11-15]. Dostupné z: https://symfony.com/doc/current/create_framework/introduction.html.

22. SHUKLA, Surabhi. *What Makes Symfony Framework a Great Choice for PHP Web Development?* [online]. Net Solutions, 2022 [cit. 2023-11-15]. Dostupné z: https://www.netsolutions.com/insights/symfony-framework-features/.

23. *Getting Started With API Platform: Create Your API and Your Jamstack Site* [online]. Kévin Dunglas, 2023 [cit. 2023-11-16]. Dostupné z: https://api-platform.com/docs/distribution/.

24. WEAVER, Ryan. *API Platform 3 Part 1: Mythically Good RESTful APIs* [online]. SymfonyCasts, 2023 [cit. 2023-11-16]. Dostupné z: https://symfonycasts.com/screencast/api-platform.

25. *JavaScript* [online]. Mozilla, 2023 [cit. 2023-11-16]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/javascript.

26. *What is JavaScript?* [online]. Mozilla, 2023 [cit. 2023-11-16]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript.

27. *Usage statistics of JavaScript as client-side programming language on websites* [online]. Q-Success, 2023 [cit. 2023-11-16]. Dostupné z: https://w3techs.com/technologies/details/cp-javascript.

28. *JavaScript basics* [online]. Mozilla, 2023 [cit. 2023-11-16]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/JavaScript_basics.

29. *Getting started with React* [online]. Mozilla, 2023 [cit. 2023-11-16]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/React_getting_started.

30. COPES, Flavio. *React for Beginners – A React.js Handbook for Front End Developers* [online]. freeCodeCamp, 2020 [cit. 2023-11-16]. Dostupné z: https://www.freecodecamp.org/news/react-beginner-handbook/.

31. *What is PostgreSQL?* [online]. PostgreSQL Tutorial, 2011 [cit. 2023-11-16]. Dostupné z: https://www.postgresqltutorial.com/postgresql-getting-started/what-is-postgresql/.

32. *About* [online]. The PostgreSQL Global Development Group, 2023 [cit. 2023-11-16]. Dostupné z: https://www.postgresql.org/about/.

33. SCOTT, Tom. *The Problem with Time & Timezones - Computerphile* [online]. Youtube, 2023 [cit. 2023-11-20]. Dostupné z: https://www.youtube.com/watch?v=-5wpm-gesOY.

34. HARFIELD, Jake. *How to Change Time Zone and Language in Outlook* [online]. Help Desk Geek.com, 2023 [cit. 2023-11-20]. Dostupné z: https://helpdeskgeek.com/office-tips/how-to-change-time-zone-and-language-in-outlook/.

35. *Use Google Calendar in different time zones* [online]. Google, 2023 [cit. 2023-11-20]. Dostupné z: https://support.google.com/calendar/answer/37064?hl=en&co=GENIE.Platform%3DAndroid.

36. NANA, TechWorld with. *Docker Tutorial for Beginners [FULL COURSE in 3 Hours]* [online]. Youtube, 2023 [cit. 2024-01-03]. Dostupné z: https://www.youtube.com/watch?v=3c-iBn73dDE.

37. *Společný evropský referenční rámec pro jazyky SERR (CEFR)* [online]. Cambridge University Press & Assessment, 2024 [cit. 2024-01-07]. Dostupné z: https://www.cambridgeenglish.org/cz/exams-and-tests/cefr/.