VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

INFORMAČNÍ SYSTÉM PRO ŘÍZENÍ ZAMĚSTNANCŮ FIRMY

INFORMATION SYSTEM FOR MANAGEMENT OF COMPANY EMPLOYEES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **PAVEL SKLENÁŘ**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. VLADIMÍR BARTÍK, Ph.D.**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2020/2021

Zadání bakalářské práce

Student: **Sklenář Pavel**
Program: Informační technologie
Název: **Informační systém pro řízení zaměstnanců firmy**
Information System for Management of Company Employees
Kategorie: Informační systémy

Zadání:

1. Seznamte se s principy tvorby webových aplikací, dostupnými prostředími a frameworky. Prostudujte také Google Drive API, problematiku OLAP a další potřebné technologie.
2. Analyzujte požadavky na informační systém pro řízení zaměstnanců zahrnující plánování směn, možnost práce s Google Drive, generování různých PDF souborů a možnost analýzy dat o docházce zaměstnanců. Systém bude využíván libovolným počtem firem, které se zaregistrují.
3. Po konzultaci s vedoucím navrhněte informační systém dle požadavků, využijte k tomu vhodné diagramy jazyka UML.
4. Navržený systém implementujte.
5. Otestujte vytvořený systém na vhodném vzorku dat.
6. Zhodnoťte dosažené výsledky a další možnosti pokračování tohoto projektu.

Literatura:

- Žára, O.: JavaScript - Programátorské techniky a webové technologie, Computer Press, 2015. ISBN: 978-80-251-4573-9.
- Welling, L., Thomson, L.: PHP a MySQL: Kompletní průvodce vývojáře. CPress, 2017.
- Laberge, R. Datové sklady: agilní metody a business intelligence. Brno: Computer Press, 2012. Expert (Grada). ISBN 978-80-251-3729-1.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1-3.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Bartík Vladimír, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 12. května 2021
Datum schválení: 27. října 2020

Abstrakt

Účelem této práce je vytvořit informační systém, který firmám pomůže se správou jejich zaměstnanců, včetně analýzy zaměstnaneckých směn pomocí OLAP, díky které lze například zjistit celkové zpoždění zaměstnanců na jejich směnách. Informační systém je realizován ve formě webové aplikace implementované v jazyce PHP s využitím frameworku Laravel společně s MySQL databází. Mezi hlavní funkce systému patří správa zaměstnanců, včetně správy jejich docházek, práce s Google Drive diskem, generování různých souborů ve formátu PDF, zobrazení různých statistik, hodnocení zaměstnanců, správa směn, zaměstnaneckých jazyků, dovolených, nemocenských a nahlášení zaměstnanců.

Abstract

Purpose of this thesis is to help companies in terms of managing their own employees including analysis of employees shifts using OLAP, thanks to OLAP analysis companies can find out for example total delays on employees shifts. Information system is created as a web application implemented in PHP using the Laravel framework with MySQL database. Information system functionalities are: employees management, including management of employees attendances, Google Drive management, generating various files in PDF format, viewing various statistics, employees ratings, shifts management, vacations management, diseases management, employees languages management, reports management, injuries management.

Klíčová slova

informační systém, docházka, dovolené, nemocenské, nahlášení, zranění, PHP, HTML, CSS, JavaScript, Laravel, OLAP, ETL, správa zaměstnanců, správa směn, MySQL, zaměstnanec, firma, Google Drive, Bootstrap

Keywords

information system, attendance, vacation, disease, report, injury, PHP, HTML, CSS, Javascript, Laravel, OLAP, ETL, employees management, shifts management, MySQL, employee, company, Google Drive, Bootstrap

Citace

SKLENÁŘ, Pavel. *Informační systém pro řízení zaměstnanců firmy.* Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vladimír Bartík, Ph.D.

Informační systém pro řízení zaměstnanců firmy

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Vladimíra Bartíka, Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Pavel Sklenář
4. května 2021

Poděkování

Rád bych poděkoval panu Ing. Vladimíru Bartíkovi, Ph.D. za vedení mé bakalářské práce, poskytnutí rad a kontroly mé práce.

Obsah

Kapitola 1

Úvod

Informační systémy nás dnes doslova obklopují takřka kdekoliv, kam se podíváme. Ať už jsou informační systémy realizovány ve formě sociálních sítí (Facebook, Whatsapp), či ve formách systémů pro správu projektů (Trello), produktů, výroby a plno dalších. Všechny tyto informační systémy sdílí jednu zcela esenciální vlastnost, která spočívá v přístupu k datům za pomocí databází online, tedy narozdíl od papírové archivace, která může být už z hlediska principu chaotická. Další velmi užitečnou vlastností informačních systému je komfortnost jejich užívání. Příkladem může být komunikace s lidmi, s kterými je nemožné mluvit osobně z důvodu velké vzdálenosti. Díky informačním systémům není potřeba jít na pobočku pošty a odeslat obálku se známkou na nějakou adresu, tedy informační systémy obecně ulehčují lidem práci.

Motivací mi pro tuto práci byla absence takového produktu v prostředí internetu (existují do určité míry systémy podobné). Tato aplikace vyniká díky propojení samotného informačního systému s Google Drive, tím umožňuje firmám spravovat jednotlivé soubory zaměstnanců (smlouvy, fotky, ...) přímo v jejich složkách, taktéž vyniká v možnostech správ dovolených, nemocenských a různých nahlášení zaměstnanců.

Cílem této bakalářské práce je vytvořit informační systém ve formě webové aplikace pro efektivní správu zaměstnanců ve firmě za účelem usnadnění administrativy zaměstnanců a zároveň pro možnost plánování zaměstnaneckých směn, správy docházek, dovolených, nemocenských a různých nahlášení. Informační systém umožňuje jednotlivé zaměstnance hodnotit, díky této funkci systém pomáhá firmám v mapování a evidování excelentních, a naopak nevýkonných zaměstnanců, zároveň každá firma po registraci obdrží na základě emailové adresy, uvedené v registraci, přístup do Google Drive firmy v rámci informačního systému. Informační systém také podporuje přímo z domovské stránky (dashboardu) vytváření složek, mazání složek či souborů a samotné nahrávání souboru do Google Drive složky firmy. Tyto funkcionality byly do informačního systému přidány pomocí Google Drive API, která k těmto účelům slouží.

Díky implementaci ve formě webové aplikace je informační systém zcela nezávislý na platformě zařízení. Pro přístup k webové aplikaci je zapotřebí mít pouze nainstalovaný webový prohlížeč a přístup k internetu.

Informační systém je implementován pomocí PHP frameworku s názvem Laravel. Pro účel ukládání dat a také pro účel manipulace s daty, je použita databáze MySQL.

Pro definici vzhledu informačního systému byl použit framework Bootstrap, neboť umožňuje vytváření responsivních webových stránek, díky této volbě je informační systém výborně použitelný i na mobilních zařízeních.

Obsah této práce je rozčleněn do několika kapitol.

Kapitola 2 pojednává o principech tvorby webových aplikací. Další kapitolou je kapitola 3 pojednávající o použitých technologiích při vývoji informačního systému pro správu zaměstnanců ve firmách.

Po představení použitých technologií následuje kapitola 4 pojednávající o analýze a specifikaci požadavků kladených na informační systém. V závěru této kapitoly se nachází diagram případů užití pro jednotlivé role v informačním systému.

Následující kapitolou je kapitola 5 pojednávající o návrhu informačního systému. V závěru této kapitoly se nachází ER diagram.

Po návrhu informačního systému následuje kapitola 6 pojednávající o implementaci informačního systému. V této kapitole je popsána implementace nejdůležitějších částí systému.

Předposlední kapitolu 7 tvoří testování informačního systému. Poslední kapitolou je kapitola 8, kde jsou zhodnoceny dosažené výsledky a nastíněn budoucí vývoj informačního systému.

Kapitola 2

Principy tvorby webových aplikací

Za účelem tvorby informačního systému je nejprve nutné vysvětlit dva pojmy. Prvním pojmem je webová aplikace a pojmem druhým je samotný pojem informačního systému.

Dále se tato kapitola zaměřuje na architektury webových aplikací a na technologie, které se běžně využívají pro vývoj webových aplikací.

2.1 Webová aplikace

Webová aplikace je typem aplikace, která je uložena na vzdáleném webovém serveru a je přístupná v prostředí internetu. K webovým aplikacím se přistupuje skrze webové prohlížeče, příkladem webového prohlížeče může být Google Chrome, Mozilla Firefox nebo Microsoft Edge. Některé webové aplikace jsou dostupné pouze ze specifických webových prohlížečů [74].

Samotné uložení webové aplikace na vzdáleném webovém serveru nestačí. Pro plnou funkčnost webová aplikace potřebuje mít také tzv. doménové jméno. Doménové jméno je způsob adresace na aplikační vrstvě TCP/IP modelu. Na základě doménových jmen je možné přistupovat k webovým aplikacím pomocí určitých slovních řetězců.

Svět bez doménových jmen si dnes jen stěží představit. Praktickým příkladem, jak by svět vypadal bez doménových jmen, může být pouhé zobrazení webové stránky společnosti Seznam.cz, kdy do vyhledávací lišty webového prohlížeče stačí zadat `www.seznam.cz` a webová stránka je nám následně zobrazena, bez doménového jména bychom museli zadávat adresu 77.75.74.172. Několik málo adres by si člověk zapamatovat dokázal, avšak fungovat takto komplexně je prakticky nemožné.

Historicky byla používána architektura typu Klient-server, která je již dnes architekturou zastaralou [35]. V současnosti značný počet webových aplikací používá tzv. Třívrstvou architekturu [34]. Na klientské straně je spuštěn tzv. tenký klient (webový prohlížeč), který se stará o zobrazování webových stránek webové aplikace [81].

Existují dva typy webových stránek. Prvním typem webových stránek jsou webové stránky statické, které jsou neměnné, to v praxi znamená, že webový server odešle klientovi webovou stránku bez jakékoliv změny. Pod tímto termínem si lze představit webovou stránku, která je vytvořena pouze za pomocí Front-End technologií (HTML, CSS), případně pomocí nějakých Front-End frameworků, příkladem může být framework Bootstrap, který je použit i v rámci této práce [53].

Druhým typem webových stránek jsou webové stránky dynamické, které mění svůj vzhled a obsah například v závislosti na čase a aktivitě uživatelů. Webový server před sa-

motným posíláním výsledných webových stránek uživatelům jednotlivé stránky modifikuje na základě nějakých kódů programů přímo pro konkrétní uživatele [54].

Tenký klient komunikuje se serverem pomocí HTTP (Hypertext Transfer Protocol), nebo pomocí HTTPS (Hypertext Transfer Protocol Secure) [81].

Protokol HTTP už by se prakticky neměl v prostředí internetu vyskytovat, kvůli své nezabezpečenosti. Každý, kdo by chtěl nějakým způsobem analyzovat tok paketů na síti, například pomocí aplikace Wireshark nebo pomocí tcpdump se zapnutým promiskuitním módem, by totiž viděl celý obsah komunikace. V poslední době například společnost Google cílí na vlastníky webových stránek, aby protokol HTTPS používali, a postupně to reflektují v SEO (optimalizace pro vyhledávače). To v praxi znamená, že pokud se nějaká webová stránka vyskytovala například na prvním místě v rámci vyhledávání a pořád nepoužívá šifrovanou komunikaci, tak se propadne v žebříčku vyhledávání.

Ještě je nutno dodat, že protokoly SSL a TLS, které se používají pro šifrovaná spojení, šifrují pouze přenos dat mezi aplikacemi, pokud se nám tedy někdo dostane do například emailového klienta, tak uvidí naše emailové zprávy v holé textové podobě. Z toho vyplývá, že žádný protokol SSL ani TLS nemůže garantovat stoprocentní bezpečnost.

Zabezpečování má dle mého názoru ale také svou nevýhodu, neboť čím více se obecně zabezpečuje, tím více se ztrácí anonymita uživatelů v prostředí internetu.

Webové aplikace jsou velmi oblíbené, neboť jsou snadno přístupné. Uživatelé pro přístup k webovým aplikacím potřebují pouze webový prohlížeč a připojení k internetu [74]. Na druhou stranu rychlost a přístupnost samotné webové aplikace může být velice variabilní, ať už kvůli zatížení webového serveru nebo kvůli rychlosti připojení klienta.

2.1.1 Architektura klient-server

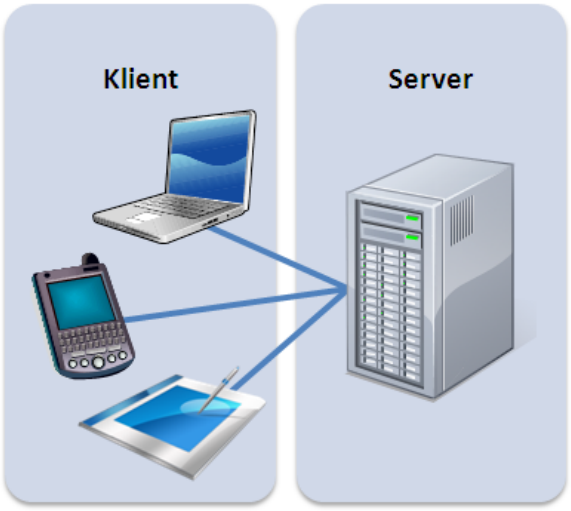
Architektura klient-server je dvouvrstvá, skládá se ze dvou částí: klienta a serveru. Klient se k serveru připojuje, pakliže chce používat libovolnou službu nabízenou serverem, tedy komunikace je iniciována ze strany klienta, po připojení klienta mezi sebou klient a server vzájemně komunikují a předávají si vzájemně data [35]. Server pro každého klienta vytváří tzv. relace [58].

Klient obsahuje uživatelské rozhraní a aplikační logiku, na serveru se nachází relační databáze. Klient má za úkol překládat uživatelské požadavky do podoby, která bude srozumitelná pro server a čeká od serveru na odpověď, kterou překládá zpět tak, aby byla srozumitelná klientovi a následně mohla být zobrazena na obrazovce uživateli. Server slouží pro zpracovávání dotazů nad relační databází a klient je prezentuje, zajišťuje aplikační logiku a zprostředkovává rozhraní pro uživatele [35].

V současnosti je architektura klient-server spíše historická, hlavně kvůli bezpečnosti a také kvůli tomu, že klient obsahuje většinu aplikační logiky, se složitostí aplikace rostou nároky na klientské počítače či zařízení. Další nevýhodou je také obtížnější aktualizace aplikace na straně klienta a také je problematické zpřístupnit část aplikace pomocí webového prohlížeče [35].

Další nevýhodou této architektury je, že server musí pracovat tzv. konkurentně, tedy musí umět pracovat s několika klienty najednou a ještě k tomu navíc udržovat relaci se všemi klienty [58].

Z architektury typu klient-server se vyvinula tzv. třívrstvá architektura [35].



Obrázek 2.1: Ukázka architektury klient-server [35].

2.1.2 Třívrstvá architektura

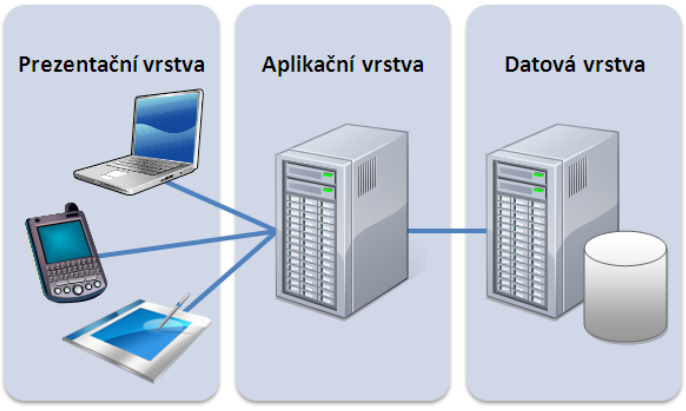
Třívrstvá architektura je v současnosti nejpoužívanější architekturou webových aplikací.

První vrstvou této architektury je vrstva prezentační, kterou reprezentuje webový prohlížeč, tato vrstva je tedy viditelná pro uživatele a je taktéž závislá na platformě [34].

Druhou vrstvou této architektury je vrstva aplikační, která je tvořena nástroji pro dynamické generování stránek [50]. Aplikační vrstva, která se taktéž nazývá aplikačním serverem, realizuje samotné výpočty a operace mezi vstupně-výstupními požadavky a daty [34].

Třetí a poslední vrstvou této architektury je vrstva datová a tvoří jí databáze [50]. Jedná se o nejnižší vrstvu Třívrstvé architektury a zajišťuje práci s daty, konkrétně například ukládání, výběr a agregaci dat [34].

Aplikační vrstvě jsou posílány požadavky od webového prohlížeče, tyto požadavky jsou aplikační vrstvou obsluhovány pomocí dotazování nad konkrétní databází, poté dochází k samotnému generování uživatelského rozhraní [50].

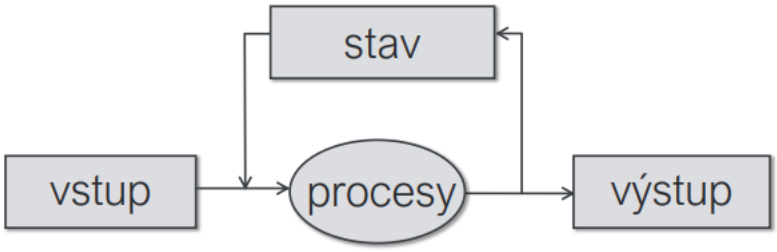


Obrázek 2.2: Ukázka třívrstvé architektury [34].

2.2 Informační systém

Informační systém je takový systém, kde jsou vzájemně propojeny informace a procesy. Procesy jsou funkce, které pracují s informacemi, které se do systému dostávají, následně na základě těchto vstupních informací je provedena transformace na informace výstupní. Informace jsou data, na základě, kterých lze vykonávat různé rozhodování, zatímco procesy sbírají, přenáší, zpracovávají a distribuují informace [85].

Informační systém má i své okolí, které tvoří všechny objekty, které libovolnou změnou svých vlastností nějakým způsobem ovlivňují systém, mezi okolí se ale také počítají objekty, které mění své vlastnosti na základě systému [85].



Obrázek 2.3: Schéma informačního systému [45].

2.3 Front-End

Termín Front-End se váže na grafický výstup aplikace. Front-End popisuje část aplikace, kterou poté uživatel reálně uvidí. Přívětivost, intuitivnost a jednoduchost, to jsou vlastnosti, které by měl každý Front-End libovolné aplikace splňovat v co největší možné míře.

2.3.1 HTML

HTML, v plném znění HyperText Markup Language, je jazyk, který se používá pro vytváření obsahu webových stránek, samotný obsah webových stránek poté tvoří tzv. HTML elementy [48].

Struktura webu je pomocí jazyka HTML popsána značkami, tzv. tagy. Tagy obalují text, a tím určují jeho význam (sémantiku) [82]. HTML elementy se skládají ze značek (tagů), případně atributů a obsahu [86].

Jazyk HTML vznikl v roce 1991, v té době obsahoval pouze 18 tagů, v roce 2019 obsahoval jazyk HTML už 140 tagů. Jazyk HTML spravuje a vyvíjí organizace W3C. Za tvůrce jazyku HTML je považován Tim Berners-Lee. HTML není programovacím jazykem.

V roce 2014 vyšel jazyk HTML ve verzi 5, který přinesl největší počet změn [48].

V jazyku HTML existují značky dvojího typu, buď jsou párové, nebo nepárové. Jednotlivé názvy tagů se uzavírají mezi znaky `<` a `>`. Pro koncový tag platí, že se uzavírá pomocí tvaru `</tag>`, počáteční tag má tvar `<tag>` [82].

Každý HTML dokument musí obsahovat tzv. základní kostru jazyka HTML, která vypadá následovně:

```
<!DOCTYPE html>
<html>
    <head>
        <title> Název webové stránky </title>
    </head>
    <body>
        Text na webové stránce, který se zobrazí uživateli v prohlížeči.
    </body>
</html>
```

V hlavičce `<head>` jsou obsažena různá metadata (odkaz na CSS soubor, titulek, klíčová slova, jméno autora, typ kódování, ...) [30].

Párový tag `<title>` slouží pro určení názvu webové stránky. Pokud chceme používat jazyk HTML ve verzi 5, je nutné v dokumentu uvést `<!DOCTYPE html>`. Párový tag `<body>` obaluje obsah stránky, která se následně zobrazuje uživatelům [82].

2.3.2 CSS

Pro popis vzhledu webové stránky napsané v jazyce HTML se používá jazyk CSS (Cascading Style Sheets), samotný popis vzhledu probíhá pomocí tzv. CSS pravidel, které určují onen styl HTML dokumentu [29]. Jazyk CSS vznikl v roce 1994 a za jeho tvůrce je považován Håkon Wium Lie [57]. Pravidla jazyka CSS mají následující syntax [29]:

```
selektor { vlastnost: hodnota; }
```

Každé CSS pravidlo se skládá ze selektoru a z bloku deklarací (část pravidla ve složených závorkách). Selektor se používá pro vyhledávání HTML elementů na základě jejich jména, identifikátoru a třídě [29].

Vlastnosti určují konkrétní styl (barvu pozadí, textu, velikost textu, zarovnání, ...), tyto vlastnosti poté mají konkrétní hodnoty, příkladem může být červená barva pozadí. [29].

Příkladem konkrétního pravidla může být:

```
p { font-size: 25px;color: blue; }
```

Výše uvedené pravidlo určuje, že text obsažený v odstavcích HTML dokumentu bude modrý a bude mít velikost 25 pixelů. Toto pravidlo se dá přepsat pravidlem tzv. specifičtějším. Například pokud by v odstavci `<p>` byl text obalen tagem `<span>` nebo `<div>`, kterým by se určila třída pomocí HTML atributu class (`<div class="test"><p>Text</p></div>`) tak poté po napsání:

```
p .test { color:yellow; }
```

se barva textu v tagu `<span>`, nebo `<div>` změní na žlutou, neboť použité pravidlo je specifičtější, než jen samotné pravidlo se selektorem `p`.

Existují tři způsoby přidání CSS stylů do HTML dokumentu [29]:

- použitím externího souboru s příponou `.css` (nejběžnější),

- použitím elementu `<style>` v hlavičce HTML dokumentu,
- použitím tzv. řádkového (inline) zápisu, kdy je definovaný styl použit pouze pro konkrétní element.

V současné době se jazyk CSS nachází ve verzi 3 [51].

2.3.3 Javascript

Jazyk JavaScript patří do rodiny interpretovaných programovacích jazyků [72]. Byl vytvořen v roce 1995 Brendanem Eichem [56]. Javascript je jazykem dynamicky typovaným, který obsahuje prvky funkcionálního a objektově orientovaného programování [72].

Je to jazyk, který pracuje nejen na straně klienta tzv. Client-side jazyk, ale potenciálně i na straně serveru tzv. Server-side jazyk, takto se například používá v Node.js[1] prostředí. Kód napsaný v Client-side jazyce je zpracováván na straně uživatele (webový prohlížeč), zatímco kód napsaný v Server-side jazyce je zpracováván na straně serveru. [62].

JavaScript lze rozdělit na dvě části. První částí je část prohlížečová a druhá je samotné jádro jazyka, které lze používat i mimo webový prohlížeč, neboť neobsahuje žádné metody pro práci s elektronickými dokumenty. Jádro jazyku Javascript se nazývá ECMAScript a je spravováno mezinárodní organizací ECMA International [72].

Jazyk Javascript dovoluje dynamicky aktualizovat obsah stránky, například na základě zakliknutí tlačítka uživatelem, tedy obecně umožňuje uživatelům interagovat s webovými stránkami [36]. Může být použit například pro [36]:

- Zobrazení či schování nějaké informace kliknutím tlačítka.
- Změnu barvy tlačítka po najetí myší na dané tlačítko.
- Přibližování či oddalování obrázku.
- Přehrání videa a audia na webové stránce.
- Zobrazování animací.

JavaScript lze přidat do HTML dokumentu dvěma způsoby [62]:

- použitím elementu `<script>` přímo v HTML dokumentu,
- použitím externího souboru s příponou `.js`.

Mezi výhody jazyka Javascript patří [37]:

- rychlost,
- jednoduchost,
- všestrannost.

Mezi nevýhody jazyka JavaScript patří [37]:

- různorodá podpora prohlížečů (každý prohlížeč může interpretovat JavaScript kód jinak),
- potenciální bezpečnostní rizika na straně klienta.

Pro usnadnění práce s JavaScriptem lze použít knihovnu jQuery[2], jejímž účelem je zjednodušit zápis (syntaxi) a také zaručuje funkčnost kódu v různých prohlížečích [31].

[1] Dostupné z: https://nodejs.org/en/.
[2] Dostupné z: https://jquery.com/.

2.3.4 AJAX

AJAX, v plném znění Asynchronous Javascript and XML, je soubor vývojářských technik, které dovolují webové aplikaci pracovat asynchronně, tedy zpracovávat libovolné požadavky na server v pozadí [49].

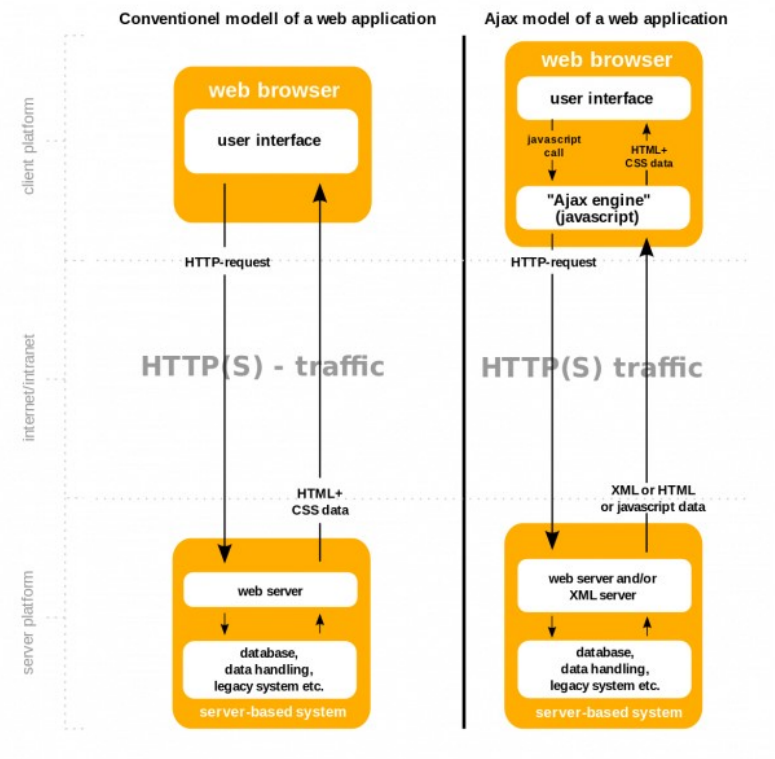
Díky AJAXu není potřeba stránky manuálně obnovovat. Javascript byl popsán výše. XML je variantou značkovacího jazyku, podobně jako jazyk HTML. Rozdíl mezi jazykem XML a HTML je ten, že jazyk HTML slouží pouze pro zobrazování dat, zatímco jazyk XML byl navržen pro uložení a přenos dat [49].

```xml
<?xml version = '1.0' encoding = 'UTF-8'?>
<Movies>
    <Movie id = "1">
        <Title> Pán prstenů: Společenstvo Prstenu </Title>
        <Released> 2001 </Released>
        <Director>
            <First_name> Peter </First_name>
            <Last_name> Jackson </Last_name>
        </Director>
    </Movie>
    <Movie id = "2">
        <Title> Interstellar </Title>
        <Released> 2014 </Released>
        <Director>
            <First_name> Christopher </First_name>
            <Last_name> Nolan </Last_name>
        </Director>
    </Movie>
</Movies>
```

Obrázek 2.4: Ilustrace formátu XML, vytvořil: autor.

Z obrázku lze vypozorovat, že jazyk XML neobsahuje žádné předdefinované tagy (jako například HTML), ale nechává samotného uživatele vytvořit tagy na základě jeho potřeb. Téměř každý uživatel už se někdy s AJAXem v praxi setkal, neboť i samotný Google vyhledávač AJAX používá [49]. AJAX se také hojně používá v aplikacích určeným pro vzájemnou komunikaci, příkladem může být Facebook [49].
AJAX není programovacím jazykem, je to systém, který se skládá z: [49]

- jazyka HTML, nebo XHTML pro definici struktury stránky,
- objektového modelu dokumentu (HTML DOM), který se používá pro dynamické zobrazování dat společně s jejich interakcí,
- jazyka XML pro výměnu dat,
- XMLHttpRequest objektu, který slouží pro asynchronní komunikaci,
- jazyka Javascript, který spojuje všechny výše uvedené technologie dohromady.



Obrázek 2.5: Porovnání konvenčního modelu webové aplikace s modelem webové aplikace používající AJAX [49].

AJAX funguje následovně [32]:

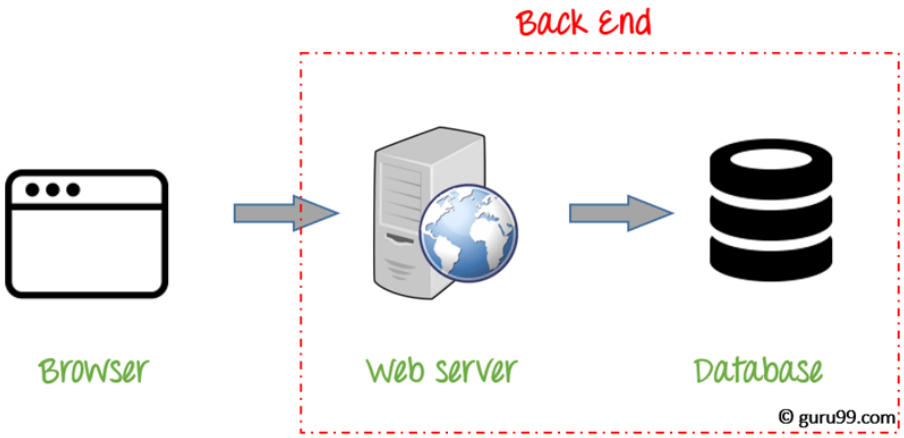
1. Objeví se událost, například uživatel stiskl tlačítko.
2. Následně JavaScript vytvoří XMLHttpRequest objekt.
3. Objekt XMLHttpRequest zašle požadavek na webový server, server následně požadavek zpracuje a zašle odpověď zpátky na webovou stránku.
4. Odpověď je přečtena Javascriptem a následně je vykonána akce (například aktualizace stránky).

2.4 Back-End

Back-End webové aplikace se skládá ze serveru, samotné aplikace a databáze. Aby spolu mohli server, aplikace a databáze komunikovat, tak je potřeba použít nějaký programovací jazyk určený pro programování serverové části (server-side programovací jazyk) aplikace (například PHP) [75].

Uživatelé Back-End aplikace nevidí, Back-End aplikace obecně vytváří výsledné webové stránky pro konkrétního uživatele na základě jeho konkrétních akcí.

Například při přihlašování do libovolného informačního systému nejprve uživatel musí vyplnit jeho konkrétní přihlašovací údaje do přihlašovacího formuláře (Front-End), následně, po stisknutí tlačítka pro odeslání formuláře, se údaje z formuláře odešlou na server, ten ověří, zdali byly ve formuláři zadány korektní hodnoty, například pomocí PHP (validace), pokud ano, server pomocí například PHP ověří, zdali se zadané údaje shodují s nějakým záznamem v databázi, pokud se zadané údaje s nějakým záznamem v databázi shodují, tak je uživatel přesměrován na jeho úvodní stranu v rámci informačního systému, kde má svá data, tedy server vygeneruje webovou stránku (Front-End) přímo pro konkrétního uživatele s jeho konkrétními daty a pošle ji zpět klientovi (webovému prohlížeči).



Obrázek 2.6: Ilustrace Back-Endu [12].

2.4.1 PHP

PHP je univerzální programovací jazyk, který byl původně navržen pro vývoj webů. Jazyk PHP vyvinul Rasmus Lerdorf v roce 1995 pro účely své osobní webové stránky. Zkratka PHP původně vznikla ze tří slov, konkrétně z: Personal Home Page, nyní zkratka PHP stojí za názvem Hypertext Preprocessor [70].

Jazyk PHP se používá pro vytváření dynamických webových stránek a webových aplikací, lze ho ale použít i k vývoji aplikací konzolových a desktopových. Jazyk PHP může být používán ve většině operačních systémů [22].

Jazyk PHP funguje na straně serveru a je jazykem dynamickým. Operace napsané v jazyce PHP provádí vždy server, který posílá výsledky oněch operací do uživatelova prohlížeče ve formátu HTML kódu [6].

Výhodou PHP je zejména malá náročnost na technické vybavení klientské strany a bezpečnost kódu. Díky popularitě je jazyk PHP podporován a předem nainstalován u většiny poskytovatelů webhostingů. Napojení na databázi je jednoduché a podporuje naprostou většinu typů databázových systémů [6].

Nejnovější podporovaná verze jazyka PHP nese označení 8.0 [21]. Pro vývoj webových aplikací existuje mnoho frameworků, které vývoj webových aplikací v jazyce PHP podporují.

Mezi aktuálně nejoblíbenější PHP frameworky patří: [64]

- Laravel,
- CodeIgniter,
- Symfony.

2.4.2 Python

Python je vysokoúrovňovým, interpretovaným a objektově orientovaným jazykem s dynamickým typováním [25]. Autorem jazyka Python je Guido Van Rossum, navrhl ho v roce 1991 [33].

Mezi výhody jazyka Python patří: jednoduchá syntaxe, dynamické typování, automatická alokace paměti a podpora mnoha knihoven. Mezi nevýhody patří zejména potenciální nepraktičnost a pomalost u větších a zároveň komplexnějších aplikací [46].

Je to jazyk poměrně jednoduchý a elegantní, pro svou jednoduchost je také nabízen lidem, kteří s programováním teprve začínají. V rámci programování webových aplikací se používají ve spojení s různými frameworky, v současnosti jsou třemi nejpoužívanějšími Python frameworky [52]:

- Django,
- CherryPy,
- Pyramid.

Dle mého názoru výhoda jazyka Python tkví také v tom, že je šířen jako open source. Nejaktuálnější verze jazyka Python nese označení 3.9.2 [24].

2.4.3 Java

Jazyk Java původně vznikl jako projekt pod názvem Oak, už v roce 1991 [7]. V roce 1995 došlo k přejmenování jazyka z názvu Oak právě na název Java, neboť název Oak už byl zaregistrován firmou Oak Technologies [13].

Autorem jazyka je James Gosling a syntaxe jazyka samotného do značné míry vychází z jazyka C++ [7].
Hlavní výhody Javy pro webový vývoj aplikací jsou následující [39]:

- Java je multiplatformní (přenositelná).
- Java je vysoce zabezpečená.
- Java je objektově orientovaná.

Tři nejpoužívanější frameworky s použitím Javy jsou následující [63]:

- Spring,
- Hibernate,
- JSF (JavaServer Faces).

2.5 Databáze

Problém s daty je takový, že mohou být velmi objemná, proto různé organizace hledají nejlepší způsob, jak tyto data řídit a spravovat. Zatím nejlepší způsob pro řízení a správu dat je použití databází [76].

Samotná databáze je organizovaná kolekce informací, která je lehce přístupná, spravovatelná a aktualizovatelná. Počítačové databáze typicky obsahují záznamy, nebo soubory obsahující informace o daných entitách konkrétní databáze [68].

Transakce jsou posloupnosti operací, které přistupují k datům a také ona data aktualizují [80]. Po dokončení transakce musí být databáze vždy v konzistentním stavu [80].

Co ale když najednou nastane výpadek proudu nebo zkolabuje systém? Z těchto důvodů, a nejen z těchto musí databázový systém zajistit integritu dat [80]. Pro samotné zajištění integrity dat musí databázový systém zaručovat tzv. ACID vlastnosti [80]. ACID vlastnosti garantují spolehlivost databázových transakcí [43].

Zkratka ACID se skládá ze čtyř slov, každé písmeno zkratky je první znak konkrétního slova, ACID vlastnostmi konkrétně jsou [80]:

- Atomičnost - Operace v transakci se provedou buďto všechny, nebo ani jedna.
- Konzistence - Transakce zachovává konzistenci databáze.
- Izolovanost - Více souběžně probíhajících transakcí se nesmí vzájemně ovlivňovat, tedy jednotlivé transakce o sobě nesmí navzájem vědět.
- Trvanlivost - Jakmile je transakce úspěšně dokončena, tak jsou změny, provedené v databázi, trvale uchovány. Na uchování změn nemá vliv ani případný výpadek systému.

Pro ukládání, modifikování, mazání dat a provádění dotazů existuje software, který se nazývá SŘBD (Systém řízení báze dat) [19]. SŘBD je mezivrstvou mezi aplikacemi a daty, které jsou uložené v databázích [19]. V této práci se využívá SŘBD MySQL.

Existuje několik typů databází, konkrétněji [68]:

- Relační databáze,
- NoSQL databáze,
- Objektově orientované databáze,
- Grafové databáze.

V této práci je použita Relační databáze.

Relační databáze se sestávají z jedné či více tabulek s tou vlastností, že tabulky mohou být navzájem propojeny. Do jednotlivých tabulek se poté ukládají hodnoty vlastností (atributy) objektů (entit). Za účelem získání nějaké informace v databázi vytváříme mezi tabulkami tzv. relace (propojení), díky kterým je nám umožněno obdržet data, která požadujeme [26].

Tabulky se skládají z řádků, jednotlivé řádky tabulky se nazývají záznamy. V relačních databázích existují dva typy klíčů: primární a cizí [26].

Zatímco primární klíč je atributem, který v tabulce jednoznačně identifikuje jednotlivé záznamy, tak cizí klíč je atributem, který na základě rovnosti hodnot s klíčem primárním vytváří vztahy mezi jednotlivými záznamy v tabulkách. Primárním klíčem nemusí být pouze jeden atribut, nýbrž i kombinace atributů, pokud se zajistí ona jednoznačnost [78].

2.5.1 MySQL

MySQL[3] patří mezi nejpopulárnější databáze vůbec. MySQL bylo vytvořeno v roce 1995 společností MySQL AB [67].

Jedná se o databázový systém relačního typu SŘBD (Systém řízení báze dat) [2].

Mezi výhody MySQL patří [44]:

- dostupnost zdarma,
- nabízí širokou škálu funkcí na to, že je poskytován zdarma,
- možnost práce s ostatními databázemi jako je například Oracle.

Mezi nevýhody MySQL patří [44]:

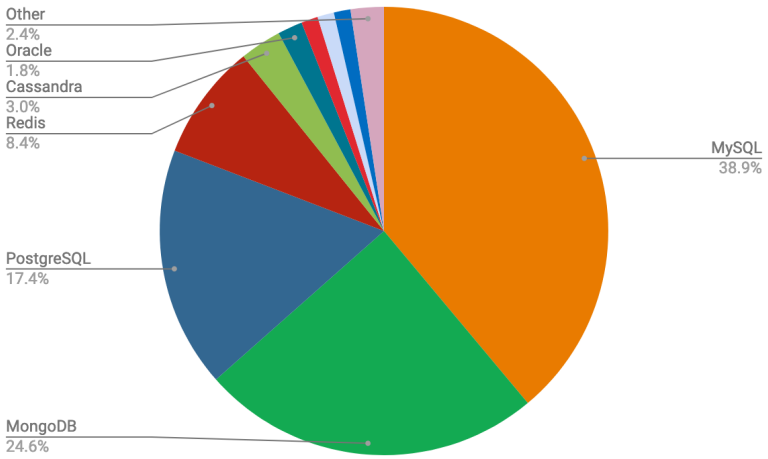
- nemá zabudovanou podporu XML a OLAP,
- pro verzi nabízenou zdarma existuje podpora, za kterou už je potřeba si zaplatit,
- potenciální strávení mnoha času nad věcmi, které ostatní databáze dělají automaticky.

MySQL využívá tzv. dvojí licencování, první licencování je GPL (bezplatné) a druhé komerční. MySQL databáze je přenositelná napříč různými platformami, jedná se tedy o multiplatformní databázi. Pro manipulaci s daty nad databází MySQL používá jako prostředek jazyk SQL [2].

GPL verze je zaměřena na rychlost a spolehlivost za cenu méně dostupných funkcí [44].

Pro správu MySQL databáze lze například používat open source nástroj s názvem PhpMyAdmin [2].

MySQL ukládá data do tabulek, každá tabulka představuje jednu entitu (například: student, učitel, ...).



Obrázek 2.7: Zobrazení popularity MySQL oproti konkurenčním databázovým systémům [38].

[3]Dostupné z: https://www.mysql.com/.

2.5.2 Ostatní používané databáze

PostgreSQL

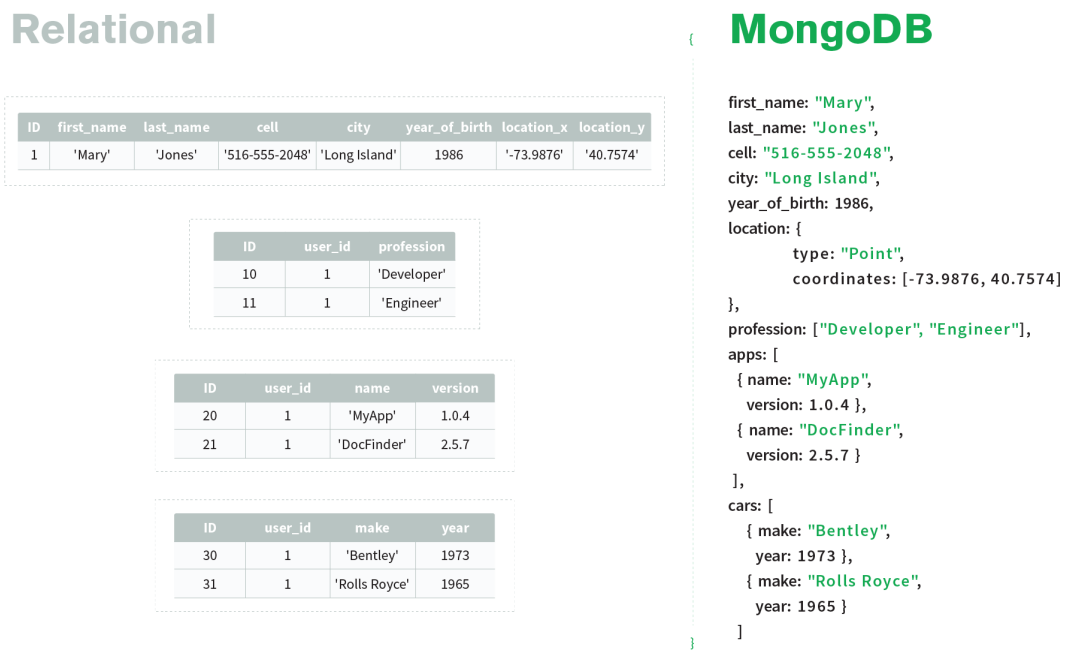
PostgreSQL je open source relační databáze, která podporuje jak SQL (relační), tak JSON (nerelační) dotazování. Patří mezi vysoce stabilní databázi [1].

Počátky PostgreSQL se datují na rok 1986, kdy to byla část projektu POSTGRES na University of California v Berkeley a aktivně se vyvíjí už více než 30 let. PostgreSQL je oblíbený databázový systém díky jeho osvědčené architektuře, spolehlivosti a zachování integrity dat. PostgreSQL splňuje ACID vlastnosti od roku 2001 [23].

Oproti MySQL je PostgreSQL rychlejší při práci s masivními soubory dat, či komplikovanými dotazy, zatímco MySQL je rychlejší při příkazech pouze pro přečtení. Obecně se MySQL hodí spíše pro obyčejné weby a online transakce, zatímco PostgreSQL je lepší pro rozsáhlé a komplikované analytické procesy. PostgreSQL podporuje více programovacích jazyků, nežli MySQL [71].

MongoDB

MongoDB je jeden z nejvíce populárních open-source NoSQL databází. Byl vyvinut společností 10gen, která je známa jako MongoDB Inc. MongoDB je dokumentově orientovaná databáze, která ukládá data ve formátu JSON. U MongoDB tedy lze ukládat záznamy bez obávání se o strukturu našich dat, jako je například počet atributů, nebo typy jednotlivých atributů. MongoDB byl naprogramován v programovacím jazyce C++. Počátky MongoDB se datují na rok 2007, kdy se firma původně jmenovala 10gen. V roce 2009 byla MongoDB uvedena jako open source pod názvem MongoDB 1.0 [69].



Obrázek 2.8: Srovnání relačních databází s NoSQL databázemi, konkrétně s MongoDB [20].

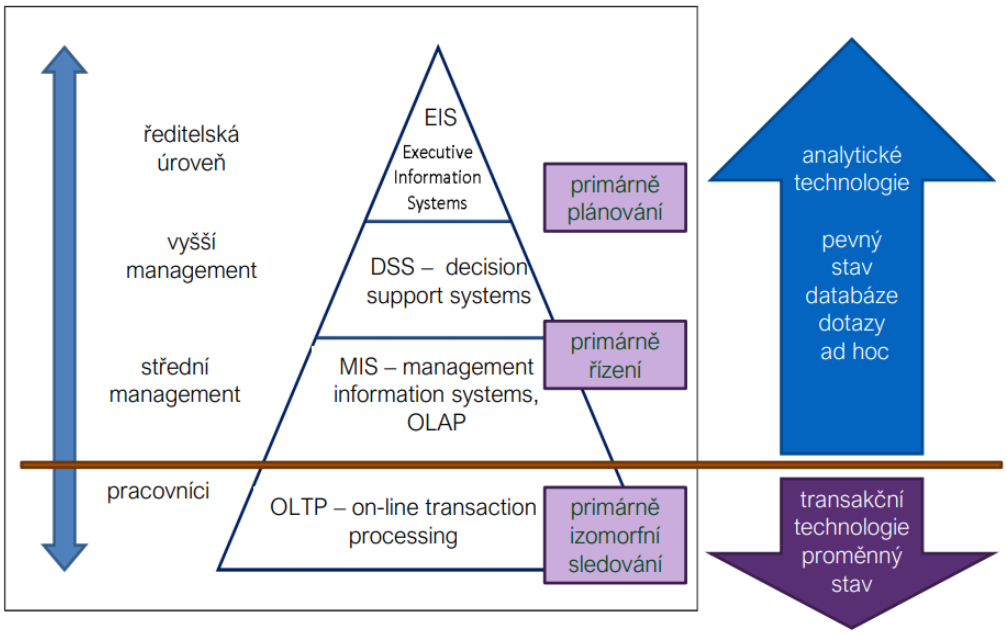
Poslední verzí MongoDB je verze 4.4, která byla vydána v červenci roku 2020 [42]. Samotné ukládání dat narozdíl od relačních databází neprobíhá v tabulkách, ale přímo v dokumentech. MongoDB vyniká svým výkonem (je výkonnější než jakákoliv relační databáze [59]), dostupností a lehkou škálovatelností [69]. V MongoDB neexistuje žádné spojování dat, jako u relačních databází, zároveň v MongoDB nejsou vůbec žádné vztahy mezi daty [59].

Mezi hlavní nevýhody MongoDB patří fakt, že nezvládá komplexní transakce [69], také používá velké množství paměti pro samotné ukládání dat, v neposlední řadě existuje limit pro velikost dokumentu, který činí 16 MB [59].

V relačních databázích musíme nastavit tabulce primární klíč, zatímco v MongoDB existuje atribut _id, který je automaticky vytvořen s každým dokumentem a chová se jako primární klíč [59].

2.6 Klasifikace informačních systémů na základě úrovně rozhodování

Schéma klasifikace informačních systému podle úrovně rozhodování je vyobrazeno na obrázku 2.9. Nejnižší vrstvou tohoto schématu je OLTP, nad vrstvou OLTP jsou systémy OLAP neboli systémy určené pro analýzu dat.



Obrázek 2.9: Pyramidové schéma klasifikace informačních systémů na základě úrovně rozhodování [45].

2.6.1 Systémy OLTP

Systémy OLTP (Online Transaction Processing) umožňují provádění několika transakcí od mnoha uživatelů online, typicky přes internet. Databázová transakce buď data mění, vkládá, maže, nebo se na ně dotazuje v databázi. OLTP systémy pohání mnoho peněžních transakcí, které skoro každý den využíváme, například: internetové bankovnictví, nákupy v e-shopech, rezervace hotelu, letenky a plno dalších. v OLTP systémech funguje atomicita transakce, buď se provede transakce celá, nebo se neprovede vůbec. Systémy OLTP se nepoužívají pouze v bankovnictví, ale používají se například i pro obyčejné transakce, příkladem takové transakce může být změna hesla či změna nějakého textu v databázi [40].

Výhody OLTP systémů jsou následující [40]:

- Zpracování velkého množství jednoduchých transakcí.
- Povoluje více uživatelům přístup ke stejným datům, zatímco zajišťuje integritu dat.
- Poskytuje indexované soubory dat.

Příklady OLTP systémů [40]:

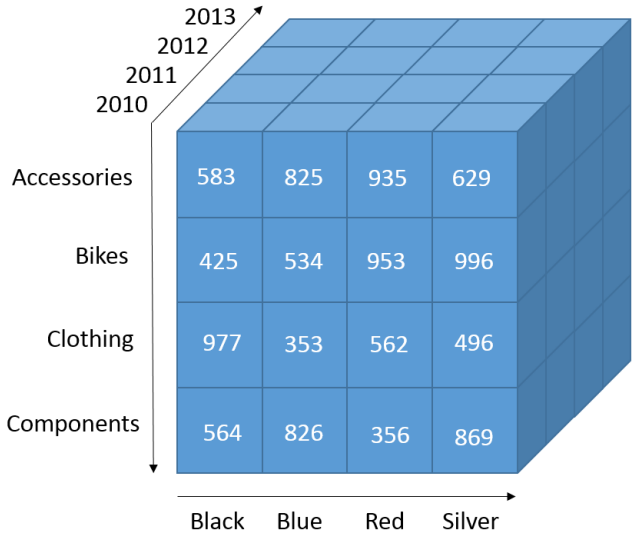
- bankomaty,
- procesy placení kreditní kartou,
- online rezervace všeho druhu,
- systémy, kde chceme obecně ukládat nějaká data (informace o zaměstnancích, zdravotní záznamy, ...).

2.6.2 Systémy OLAP

OLAP (Online Analytical Processing) jsou nástroje pro zpracování analýz. Výsledky analýzy slouží jako podklady pro manažérské rozhodování (OLAP je součástí nástrojů Bussiness Intelligence). OLAP technologie používá multidimenzionální data, multidimenzionální data tvoří tzv. datovou kostku [47].

Pro účel modelování dat existuje datová kostka, díky které mohou být data modelována a vnímána jako data multidimenzionální. Datová kostka se skládá z dimenzí a faktů. Dimenze jsou entity nebo pohledy a fakta jsou měrné jednotky číselného charakteru. Na základě množství, které reprezentují jednotlivé fakty, lze provádět analýzu vztahů mezi dimenzemi. Na základě dimenzí chce typicky organizace ukládat záznamy. Kostka je organizována kolem nějakého základního tématu, příkladem mohou být prodeje, to je reprezentováno onou tabulkou faktů. Příkladem faktu může být celková cena nákupu. Tabulka faktů kromě samotných fakt obsahuje cizí klíče do jednotlivých dimenzí [79].

Většina business dat má více dimenzí, například datová kostka pro analýzu prodejů by se skládala ze tří dimenzí: lokace, čas a konkrétní produkt. Lze ale přidávat více dimenzí. V datovém skladu jsou data uložena do tabulek, každá tabulka má pouze dvě dimenze, proto by byla v obyčejných relačních databázích analýza dat velmi pomalá, analýza dat pomocí OLAP je velmi rychlá a efektivní. OLAP prvně extrahuje data z několika tabulek (které spolu souvisí) a reorganizuje je do multidimenzionálního formátu [41].



Obrázek 2.10: Ukázka datové kostky [61].

Schémata OLAP

OLTP databáze jsou modelovány pomocí ER-diagramu. Datové sklady jsou modelovány použitím multidimenzionálních modelů. Multidimenzionální model nabízí tři formy. První formou je schéma hvězdy, druhou schéma sněhové vločky a třetí formou schéma souhvězdí [79].

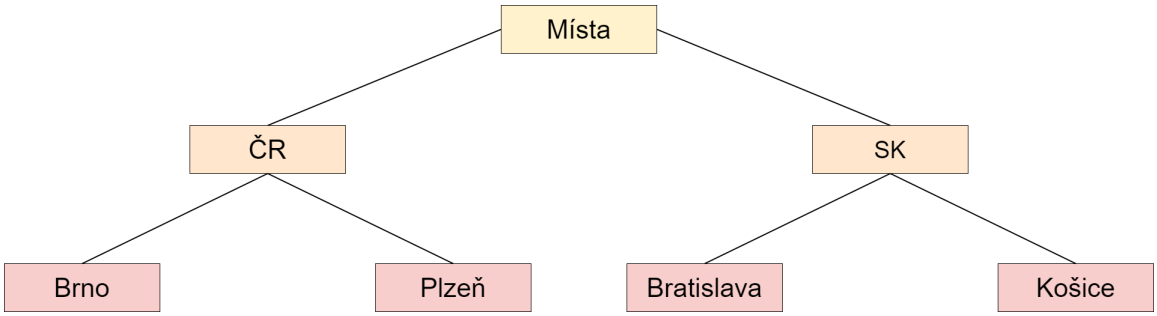
Schéma hvězdy obsahuje jednu centrální tabulku, která se nazývá tabulkou faktů. V tabulce faktů se vyskytují data bez jakékoliv redundance. Okolo této centrální tabulky se nachází tabulky jednotlivých dimenzí [79].

Schéma sněhové vločky představuje vylepšenou variantu schématu hvězdy. V tomto schématu jsou některé tabulky dimenzí normalizovány. Hlavní výhodou tohoto schématu je především snížení redundance, díky snížení redundance je u rozsáhlých tabulek možné zaznamenat značné ušetření prostoru. Ve výsledku je ale celková úspora minimální, neboť největší prostor zabere tabulka faktů. U tohoto schématu je komplikovanější dotazování kvůli nutnosti více spojení, proto se toto schéma moc nepoužívá [79].

Schéma souhvězdí je kolekcí schémat hvězd, tím pádem se v tomto schématu vyskytuje více tabulek faktů sdílejících některé z tabulek dimenzí. V tomto schématu existují nejen datové sklady, ale i datové trhy. Datový sklad je typicky objemný a váže se na data z celé společnosti, zatímco datové trhy se naopak týkají například pouze jednoho oddělení společnosti. Datové trhy jsou typicky modelované ve formě schématu hvězdy nebo sněhové vločky [79].

OLAP Konceptuální hierarchie

„Představuje sekvenci mapování z množiny konceptů nižší úrovně do konceptů vyšší úrovně [79].“ Příklad konceptuální hierarchie je vyobrazen na obrázku 2.11, který pojednává o dimenzi místo [79]:



Obrázek 2.11: Ukázka konceptuální hierarchie, vytvořil: autor.

Operace v OLAP

Díky OLAP operacím můžeme data sledovat z různých úhlů pohledu. Zde se zaměřím na operace: Roll-Up, Drill-Down, Slice and Dice a Pivot.

Roll-Up - slouží k agregaci na datové kostce, způsobí posunutí z dané hierarchie o úroveň výše. Příkladem může být výše zmíněná dimenze místo, v tomto případě bychom se posunuli od měst ke krajům. Díky této operaci můžeme jednotlivé dimenze dokonce i redukovat, tedy pokud nás nezajímají konkrétní místa prodeje, můžeme se od nich zcela abstrahovat, následně datová kostka bude zobrazovat součet všech prodejů bez ohledu na místa prodejů [79].

Drill-Down - zatímco u operace Roll-Up se posouváme v dané hierarchii o úroveň výše, u operace Drill-Down je tomu naopak. Tedy v rámci této operace se posouváme o úroveň níže k tzv. detailnějším úrovním. Pokud bychom měli například dimenzi čas, tak by to znamenalo přejití od roků k měsícům nebo od měsíců k jednotlivým dnům [79].

Slice & Dice - operace **Slice** slouží k selekci v rámci jedné dimenze, zatímco operace **Dice** slouží k selekci v rámci několika dimenzí, výsledkem selekcí je podkostka [79].

Pivot - jedná se o vizualizační operaci, díky které lze měnit datové osy. Smyslem této operace je změna prezentace dat v rámci datové kostky. Konkrétně může jít například o rotaci os ať už v 2D či 3D kostce. Je zde také možnost zobrazit 3D kostku jako kolekci 2D kostek [79].

Typy OLAP serverů

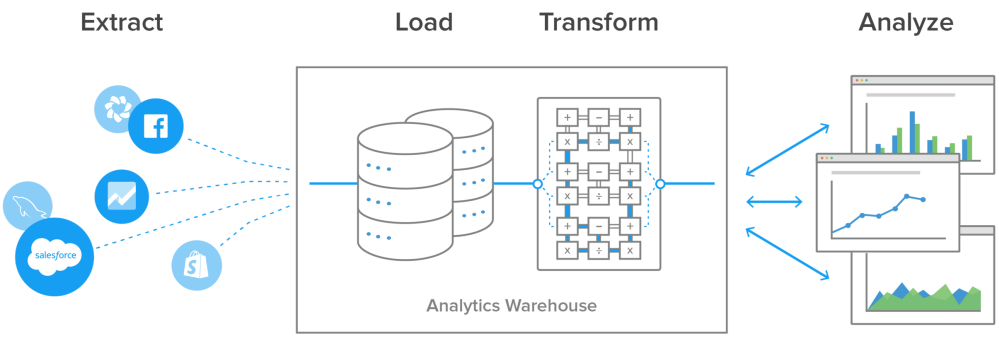
Existují čtyři typy OLAP serverů [79]:

- Relační (ROLAP) - využívá relační databázové systémy pro uložení dat.
- Multidimenzionální (MOLAP) - využívají multidimenzionální uložiště založené na polích.
- Hybridní (HOLAP) - kombinace MOLAP a ROLAP technologie.
- Specializované OLAP servery - patří mezi ně relační databázové servery nabízející pokročilé dotazovací jazyky. Tyto dotazovací jazyky lze poté použít pro vytváření dotazů u schémat hvězdy nebo sněhové vločky.

Pro účel přenosu dat z transakčních systémů (OLTP) do systémů analytických (OLAP) existují tzv. ETL procesy. ETL je zkratka skládající se ze tří znaků, první znak reprezentuje slovo Extract, druhý Transform a třetí Load [28].

Postup procesu je následující [28]:

1. Extrakce dat z OLTP systému, ale pouze těch dat, které nás zajímají, neděláme kopii.
2. Transformace, například pokud jsou data reprezentující nějakou částku v jiných měnách, tak je nejprve potřeba provést transformaci na měnu společnou.
3. Uložení dat do OLAP sekce systému pro následné získávání znalostí z dat.



Obrázek 2.12: Ukázka ETL procesu [28].

Kapitola 3

Použité technologie

Tato kapitola se zabývá technologiemi použitými při vývoji informačního systému pro správu zaměstnanců ve firmách. Pro samotný vývoj informačního systému byl zvolen PHP framework s názvem Laravel. Hlavními důvody pro vybrání právě tohoto frameworku tkví v jeho oblibě, efektivitě a velmi pěkné dokumentaci.

3.1 Front-End technologie

Pro vývoj Front-Endu informačního systému je použit framework Bootstrap společně se značkovacím jazykem HTML ve verzi 5. Pro specifické úpravy vzhledu je použit jazyk CSS ve verzi 3. Dále je použit jazyk Javascript společně s knihovnou jQuery. Pro načítání pouze částí stránek je použita technologie AJAX. Validace údajů z formulářů se provádí na straně klientské i serverové. Validace na klientské straně je realizována pomocí HTML atributů. Validace na straně serverové je realizována jazykem PHP.

3.1.1 Bootstrap

Bootstrap[4] je nejoblíbenějším open source frameworkem pro HTML, CSS a JavaScript vůbec. Díky Bootstrapu lze vytvářet responzivní webové stránky, které budou vypadat dobře i na mobilních zařízení (tzv. mobile-first). Používání Bootstrapu je jednoduché, například pro realizaci responzivního obrázku stačí pouze k obrázku samotnému přidat třídu `img-responsive`. Bootstrap nabízí k použití hned několik komponent, příkladem mohou být: navigační lišty, tlačítka, rozbalovací nabídky a další. Bootstrap byl vytvořen Markem Ottem a Jacovem Thorntonem [66].

[4] Dostupné z: https://getbootstrap.com/.

3.1.2 Chart.js

Chart.js[5] je open source Javascriptová knihovna. Používá se za účelem vizualizace dat pomocí grafů. Chart.js podporuje 8 různých typů grafů, konkrétně grafy: paprskové, plošné, spojnicové, sloupcové, polární, prstencové, bublinové a bodové. Chart.js k samotnému vykreslování grafů používá HTML5 canvas. V Chart.js lze vykreslovat více grafů do jednoho canvasu. Chart.js nabízí také mnoho možností pro animace. Jednotlivé grafy je možné vytvořit po každé změně velikosti okna, tedy je možné dosáhnout responzivity jednotlivých grafů [3].

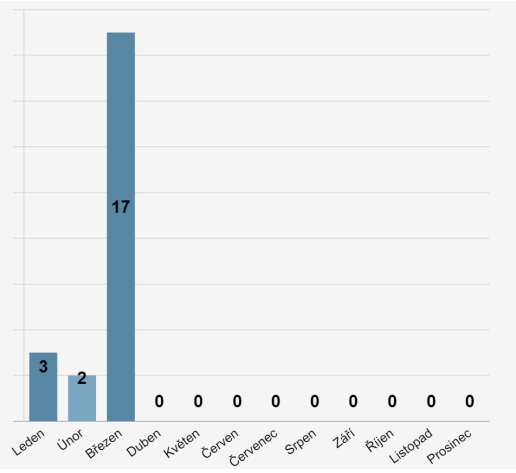
3.1.3 Chart.js Doughnutlabel plugin

Chart.js Doughnutlabel plugin[6] byl v práci použit u prstencových grafů (tzv. Doughnut grafy), díky tomuto modulu je možné zobrazovat hodnotu prstencového grafu jako text přímo uprostřed grafu.

Obrázek 3.1: Příklad použití pluginu Chart.js Doughnutlabel, vytvořil: autor.

3.1.4 Chart.js Datalabels plugin

Chart.js Datalabels plugin[7] umožňuje zobrazit jednotlivé hodnoty v grafu jako text přímo v něm.



Obrázek 3.2: Příklad použití pluginu Chart.js Datalabels, vytvořil: autor.

[5]Dostupné z: https://www.chartjs.org/.
[6]Odkaz na Github: https://github.com/ciprianciurea/chartjs-plugin-doughnutlabel.
[7]Odkaz na Github: https://github.com/chartjs/chartjs-plugin-datalabels.

3.2 Back-End technologie a nástroje

3.2.1 Laravel

Laravel[8] patří v současnosti mezi nejpoužívanější PHP frameworky vůbec [73]. Vytvořil ho Taylor Otwell, k prvnímu oficiálnímu vydání došlo v červnu roku 2011 [65]. Laravel je zejména pohodlný na práci s databázemi. Webové aplikace vytvořené za pomocí Laravelu fungují na principu Model-View-Controller architektury (MVC) [73].

Pro tvorbu Front-Endu Laravel používá šablonovací systém Blade, který dovoluje psát i různé konstrukce jazyka PHP přímo do HTML kódu. Pro využívání šablonovacího systému Blade je potřeba, aby měli soubory příponu `.blade.php`, soubory s touto příponou jsou následně zkompilovány do čistého PHP kódu a jsou následně kešovány do doby, dokud se v nich neprovede nějaká libovolná změna [16].
Mezi výhody Laravelu zejména patří [73]:

- nástroj Artisan,
- vestavěný autentifikační a autorizační balíček,
- Eloquent ORM,
- MVC architektura,
- reverse routing,
- propracovaná dokumentace.

3.2.2 Architektura MVC

Model-View-Controller je architektonický vzor, který se stal velmi oblíbeným, zejména v prostředí webových aplikací. Tuto architekturu používají různé webové frameworky, například i výše zmíněný Laravel. Tato architektura rozděluje aplikace do komponent 3 typů. Prvním typem je Model, druhým View a třetím Controller. Díky této architektuře se aplikace stávají přehlednějšími [84].

Model

Model je typem komponenty obsahující veškerou logiku aplikace. Může obsahovat například databázové dotazy a různé výpočty. Model neví o existenci controlleru nebo view [84].

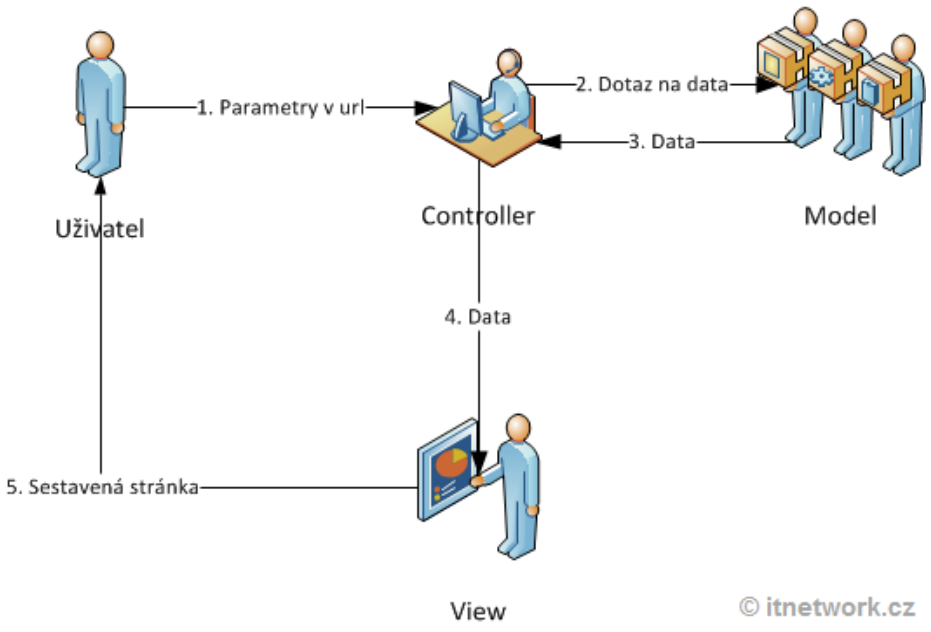
View

View je typem komponenty, která slouží k zobrazování požadovaných dat uživateli. Zobrazování jednotlivých view je většinou realizováno přes šablonovací systémy. View nemá samo o sobě možnost požádat o nějaké data, data jsou view vždy poskytnuta [84].

Controller

Controller je typem komponenty, která slouží jako prostředník mezi uživatelem, modelem a view. Propojuje jednotlivé komponenty, a tím drží systém pohromadě [84].

[8] Dostupné z: https://laravel.com/.



Obrázek 3.3: Architektura Model-View-Controller [84].

3.2.3 Artisan

Artisan[9] je konzolová aplikace, která je dodávána jako součást Laravelu. Poskytuje mnoho užitečných příkazů, které mohou pomoci při vytváření webové aplikace. Pro zobrazení všech dostupných Artisan příkazů existuje příkaz list, viz níže [14].

```
php artisan list
```

Každý příkaz mimo jiné obsahuje nápovědu, která popisuje argumenty a možnosti jednotlivých příkazů, například pro migraci databáze [14]:

```
php artisan help migrate
```

Pro vytváření například modelů, kontrolerů, notifikací a dalších se používá [14]:

```
php artisan make
```

Sadu příkazů nástroje Artisan lze rozšiřovat o příkazy nové, například instalací různých balíčků. Existuje možnost vytvářet si i vlastní příkazy [14].

[9] Dostupné z: https://laravel.com/docs/8.x/artisan.

3.2.4 Composer

Composer[10] (Dependency Manager) je nástroj určený pro správu knihoven a také pro správu závislostí samotných knihoven navzájem. Používá se v aplikacích, které jsou naprogramovány pomocí PHP. Umožňuje nám snadnou a automatickou instalaci knihoven. V souboru composer.json, která je typicky kořenovém adresáři projektu, se zadávají jednotlivé knihovny, které chce uživatel nainstalovat [5].

Composer se ovládá z příkazové řádky. Příkaz `composer install` spustí instalaci všech knihoven uvedených v souboru `composer.json`. Pro aktualizaci knihoven existuje příkaz `composer update` [4].

Po instalaci či aktualizaci jsou jednotlivé knihovny umístěny do složky s názvem vendor [4].

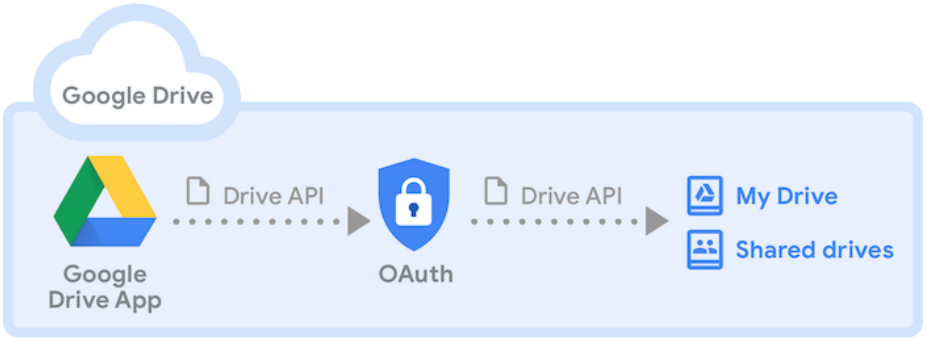
3.2.5 DOMPDF Wrapper for Laravel

Knihovna DOMPDF Wrapper for Laravel[11] je v práci použita pro generování souborů ve formátu PDF.

Nejprve bylo nutné tuto knihovnu přidat do Laravel projektu pomocí composer příkazu require, konkrétně: `composer require barryvdh/laravel-dompdf`.
Následně se knihovna musela zaregistrovat do souboru app.php do pole providers pomocí: `Barryvdh\DomPDF\ServiceProvider::class` [55].

3.2.6 Google Drive API

Díky Google Drive API[12] je možné vytvářet aplikace, kterými lze ovládat samotný úložný prostor Google Drive konkrétně zvoleného Google účtu [9].



Obrázek 3.4: Ilustrace vztahů mezi Google Drive aplikací, Google Drive a Google Drive API [9].

Nyní si rozebereme jednotlivé komponenty z obrázku 3.4.

[10] Dostupné z: https://getcomposer.org/.
[11] Odkaz na Github: https://github.com/barryvdh/laravel-dompdf.
[12] Odkaz na Github: https://github.com/googleapis/google-api-php-client.

Google Drive

Google Drive je služba poskytující úložný prostor v rámci Google účtů. Po založení Google účtu je nabízen Google Drive s velikostí 15 GB zdarma [11]. V rámci Google Drive je nejen možnost spravovat své vlastní složky a soubory, nýbrž je možnost i jednotlivé soubory, či složky sdílet, tedy Google Drive nabízí i formu spolupráce mezi uživateli. V Google Drive existují dokonce i role, které určují, jaké práva má uživatel k přístupu k určité složce, či souboru. Rolí je dohromady pět a jsou následující [10]:

- owner - vlastník má veškerá práva, která Google Drive nabízí.
- fileOrganizer - organizátor souborů a složek, má podobná práva jako vlastník, ale nemůže složky či soubory mazat, nebo je přesouvat mimo Google Drive.
- writer - uživatel s právy pro čtení a zápis, může například soubory vytvářet, modifikovat a číst je. Oproti roli fileOrganizer této roli není umožněno přesouvat položky do koše, a také reorganizovat položky v Google Drive.
- commenter - uživatel s právy číst soubory, nemůže vytvářet složky. Této roli je umožněno si zobrazit obsah složek, v kterých si může zobrazit obsah libovolného soubor. Role commenter také dovoluje psát do jednotlivých souborů komentáře.
- reader - uživatel s právy pouze pro čtení, tedy má ty stejná práva jako už zmíněná role komentátora (commenter), tedy až na psaní oněch komentářů do souborů.

Google Drive API

Komponenta, díky které můžeme manipulovat s Google Drive přes naše aplikace [9].

Google Drive Aplikace

Námi vytvořená webová aplikace, která bude komunikovat přes Google Drive API s Google Drive úložištěm [9].

My Drive

Tato komponenta je podmnožinou Google Drive úložiště, jedná se o úložiště Google Drive konkrétního uživatele. V rámci Google Drive lze složky, či jednotlivé soubory sdílet [9].

OAuth 2.0

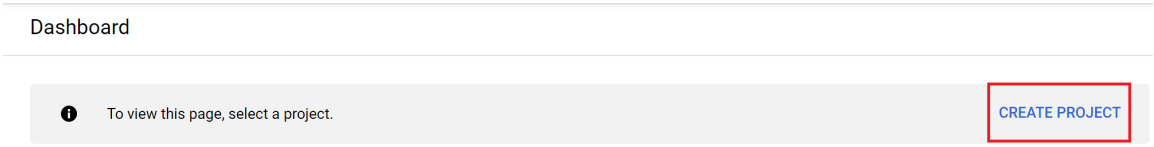
OAuth 2.0 poskytuje klíč, přes tento klíč se připojujeme, když chceme Google Drive API používat. Používá se pro autentizaci uživatele. V této práci je používán tzv. servisní účet, díky tomuto účtu se uživatelé nemusí autorizovat [9].

Google Drive API lze použít například k následujícím činnostem [9]:

- stahování a nahrávání souborů na Google Drive
- hledání souborů a složek v Google Drive
- sdílení složek, souborů i samotných disků za účelem spolupráce s ostatními uživateli

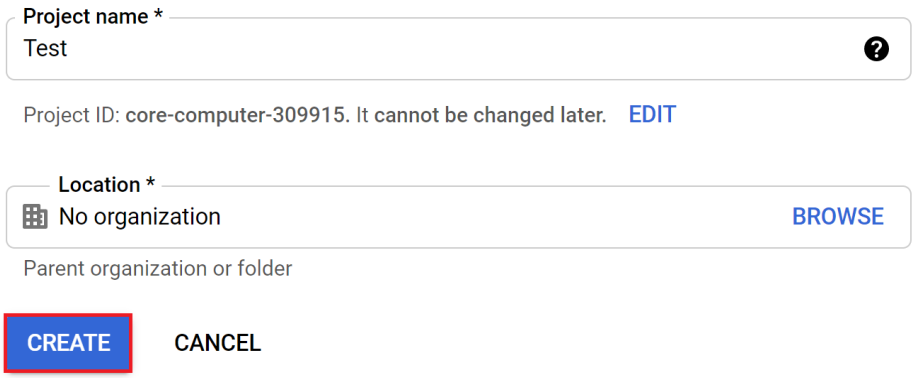
Připojení k Google Drive prakticky

Na začátku je potřeba vytvořit projekt. Projekt se vytváří na domovské stránce Google API Console[13], viz obrázek 3.5 [60].



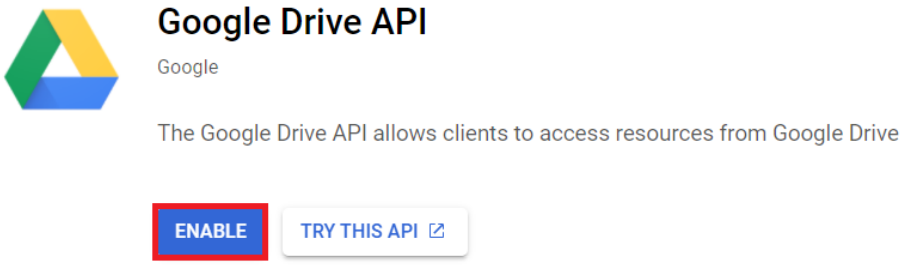
Obrázek 3.5: Vytvoření projektu na domovské stránce Google API Console [8].

Po stisknutí tlačítka pro vytvoření projektu (create project) se objeví formulář, kde vyplníme název projektu a případně jméno organizace, viz obrázek 3.6 [60].



Obrázek 3.6: Formulář pro vytvoření projektu [8].

Po vytvoření projektu musíme pro daný projekt povolit používání Google Drive API. Na domovské stránce se nachází tlačítko s názvem `ENABLE APIS AND SERVICES`, po stisknutí tohoto tlačítka se nám zobrazí vyhledávací lišta. Do lišty zadáme řetězec Google Drive API, následně je potřeba kliknout na ikonku Google Drive API, následně se nám zobrazí webová stránka, kde lze Google Drive API povolit, viz obrázek 3.7 [60].



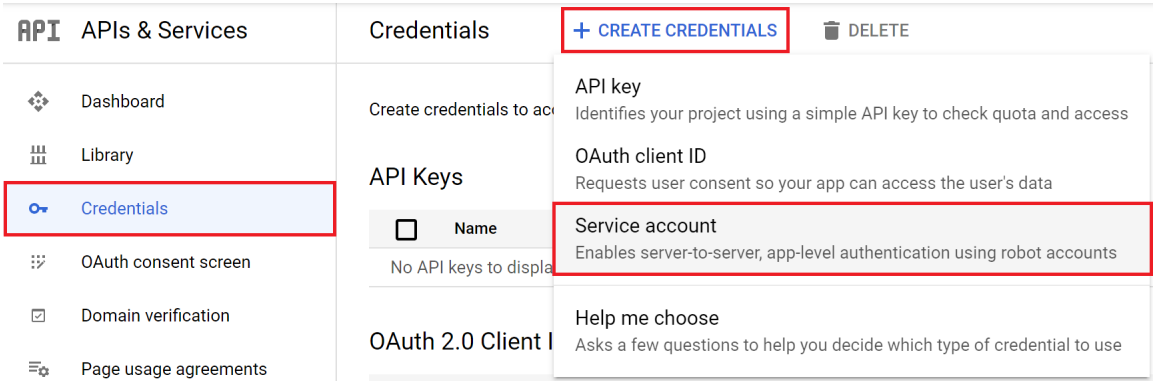
Obrázek 3.7: Povolení Google API v rámci projektu [8].

[13] Dostupné z: https://console.developers.google.com/.

Po povolení Google Drive API v projektu už nám pouze zbývá vytvořit tzv. Servisní účet (Service account). Tento účet lze vytvořit na domovské stránce Google API Console v záložce `Credentials`. V této záložce se nachází tlačítko s názvem `CREATE CREDENTIALS`. Po stisknutí tohoto tlačítka se objeví nabídka, ze které vybereme možnost `Service account` [60].

Po vytvoření servisního účtu nám bude k dispozici soubor `credentials.json` a samotný název servisního účtu. Soubor `credentials.json` slouží jako klíč, díky tomuto klíči nám bude umožněno přihlašování do Google Drive bez nutnosti autorizace [60].

Následně náš Google Drive musíme nasdílet našemu servisnímu účtu, po tomto kroku už je Google Drive API připraveno k používání [60].



Obrázek 3.8: Vytvoření servisního účtu [8].

3.3 Databáze

Webová aplikace používá databázi MySQL. Jako nástroj pro administraci databáze je použit phpMyAdmin, který umožňuje správu MySQL databází a současně je jeden z nejrozšířenějších nástrojů, pro administraci databází, vůbec.

Kapitola 4

Analýza a specifikace požadavků

Analýza a specifikace požadavků je obecně alfou a omegou všech projektů a aplikací. Od zákazníka dostaneme požadavky na aplikaci, kterou by chtěl realizovat. Poté je na nás tyto požadavky nějakým způsobem analyzovat a validovat. Musíme zjistit, zdali jsou požadavky na aplikaci vůbec reálné nebo jestli časový rámec, pro vývoj dané aplikace, stanovený zákazníkem není příliš krátký. Potenciálním problémem může být fakt, že sám zákazník pořádně neví, co od dané aplikace očekává nebo to nedokáže exaktně popsat. V průběhu tvorby aplikace může navíc dojít ke změně požadavků na danou aplikaci.

Pro eliminaci nejednoznačností, které vznikají při používání běžné řeči, se používají různé diagramy, příkladem může být Use-Case Diagram.

V informačním systému existují celkově tři role uživatelů:

- firma,
- zaměstnanec,
- admin.

V následujících sekcích budou popsány jednotlivé požadavky na informační systém, včetně popsání výše uvedených rolí v rámci jednotlivých požadavků.

4.1 Správa firem

Firma a Admin

Firmám je umožněna registrace dvěma způsoby. Prvním způsobem je registrace za pomocí registračního formuláře. Způsobem druhým je vytvoření účtu firmy adminem informačního systému. Počet registrací je v informačním systému neomezený.

Po registraci či vytvoření firmy adminem je firmě automaticky zaslána zpráva o sdílení prostoru na Google Drive na emailovou adresu zadanou při registraci, následně je automaticky vytvořena složka v Google Drive informačního systému (to vše při aktivaci Google Drive v rámci registrace), která nese název firmy a emailovou adresu zástupce firmy. Firmě jsou následně udělena práva role writer pouze pro její nově vytvořenou složku v Google Drive. Firma vidí pouze svoji složku, nevidí složky ostatních firem. Firmě je také zaslána emailová zpráva sloužící k ověření emailové adresy účtu. Bez ověření emailové adresy není umožněn vstup do informačního systému.

V informačním systému je možné měnit různé údaje firmy, tyto údaje může případně měnit i admin, příkladem může být změna názvu firmy, která se reflektuje i v Google

Drive názvu složky firmy. Název firemní složky v Google Drive je tedy vždy aktuální s názvem a emailovou adresou zadanou v účtu firmy. Dále lze změnit IČO, město, či ulici sídla firmy, křestní jméno a příjmení zástupce firmy a telefonní číslo. Změna hesla účtu je taktéž umožněna. V informačním systému je umožněno nahrávat a odstraňovat profilový obrázek firmy. Firma má také možnost smazat svůj účet, tato možnost je poskytnuta i adminovi. Při zapomenutém heslu má firma možnost resetování hesla. Do informačního systému se lze přihlašovat buďto za pomocí uživatelského jména nebo emailové adresy.

4.2 Správa jazyků

Správa jazyků se týká pouze uživatelů s rolí firmy.

Firma

V informačním systému lze vytvářet a odstraňovat jazyky. Libovolná firma si může vytvořit své vlastní jazyky, které budou viditelné jen pro onu firmu, ostatní firmy jazyky vytvořené jinou firmou neuvidí. Tedy pokud si například firma A vytvoří jazyky: japonština, němčina a angličtina, tak firma B i firmy ostatní tyto jazyky neuvidí, uvidí jen své vlastní vytvořené jazyky.

Každý nový účet firmy ve výchozím nastavení nemá vytvořený žádný jazyk. Vytvořené jazyky lze následně přiřazovat zaměstnancům, díky tomu je možné evidovat, jaký zaměstnanec ovládá, jaké jazyky.

4.3 Správa dovolených

Dovolené se skládají ze dvou datumů, a to z datumu začátku dovolené a z datumu konce dovolené, k dovolené lze přidat poznámku, která je volitelná.

Firma

Může dovolené vytvářet, upravovat a odstraňovat bez jakýchkoli omezení. U jednotlivých dovolených je zobrazen jejich stav a aktuálnost. Aktuálnost vyjadřuje, zdali už dovolená proběhla, probíhá, nebo teprve proběhne. Stav dovolené vyjadřuje, v jaké fázi se daná dovolená nachází. Stav se skládá z 5 možných hodnot:

- nezažádáno,
- odesláno,
- schváleno,
- neschváleno,
- přečteno.

Po vytvoření dovolené firmou má dovolená automaticky stav odesláno. U dovolených je možné průběžně libovolně měnit stavy, žádný stav se neuzamyká.

Zaměstnanec

Po vytvoření dovolené zaměstnancem je daná dovolená pro firmu neviditelná, nachází se ve stavu nezažádáno. Po zažádání se stane dovolená pro firmu viditelná.

Zaměstnanec může jednotlivé své dovolené upravovat a odstraňovat pouze pokud se nachází ve stavu nezažádáno. Pokud se dovolená nachází ve stavu odesláno, lze žádost zrušit a následně případně odstranit celou dovolenou. Pakliže se dovolená nachází ve stavu schváleno, neschváleno, nebo přečteno, tak zrušení žádosti o dovolenou není možné.

4.4 Správa nahlášení

Nahlášení se skládají z názvu a popisu, případně důležitosti, která je volitelná.
Důležitost nahlášení může být následující:

- zásadní,
- naléhavá,
- důležitá,
- normální,
- nedůležitá,
- nespecifikováno.

Firma

Může nahlášení vytvářet, upravovat a odstraňovat bez jakýchkoli omezení. U jednotlivých nahlášeních je zobrazen jejich stav. Stav nahlášení vyjadřuje, v jaké fázi se dané nahlášení nachází. Stav se skládá z 5 možných hodnot, viz sekce 4.2.

Po vytvoření nahlášení firmou má dané nahlášení automaticky stav odesláno. U nahlášeních je možné průběžně libovolně měnit stavy tak jako u dovolených.

Zaměstnanec

Po vytvoření nahlášení zaměstnancem je dané nahlášení pro firmu neviditelné, nachází se ve stavu nenahlášeno. Po odeslaní nahlášení se stane dané nahlášení viditelným i pro firmu.

Zaměstnanec může své nahlášení upravovat a odstraňovat pouze pokud se nachází ve stavu nenahlášeno. Pokud se nahlášení nachází ve stavu odesláno, lze žádost zrušit a následně případně odstranit nahlášení celé. V případě, že se nahlášení nachází ve stavu schváleno, neschváleno, nebo přečteno, tak zrušení nahlášení není možné.

4.5 Správa nemocenských

Nemocenské se skládají z názvu a z dvou datumů, konkrétně z datumu začátku a konce nemocenské a z poznámky k nemocenské. Poznámka k nemocenské je volitelná.

Firma

Může nemocenské vytvářet, upravovat a odstraňovat bez jakýchkoli omezení. U jednotlivých nemocenských je zobrazen jejich stav. Stav nemocenských vyjadřuje, v jaké fázi se dané nemocenské nachází. Stav se skládá z 5 možných hodnot, viz sekce 4.2.

Po vytvoření nemocenské firmou má daná nemocenská automaticky stav odesláno. U nemocenských je možné průběžně libovolně měnit stavy tak jako u dovolených a nahlášení.

Zaměstnanec

Po vytvoření nemocenské zaměstnancem je daná nemocenská pro firmu neviditelná, nachází se ve stavu nezažádáno. Po odeslaní nemocenské se stane daná nemocenská viditelná i pro firmu.

Zaměstnanec může své nemocenské upravovat a odstraňovat pouze pokud se nachází ve stavu nezažádáno. Pokud se nemocenská nachází ve stavu odesláno, lze žádost zrušit a následně případně odstranit nemocenskou celou, v případě, že se nemocenská nachází ve stavu schváleno, neschváleno, nebo přečteno, tak zrušení nemocenské není možné.

4.6 Správa zranění

Zranění se skládají ze čtyř částí, a to z konkrétního zaměstnance a směny, dále z datumu zranění a popisu zranění.

Firma

Zranění na směnách může vytvářet, upravovat a případně odstraňovat. Vytvářet zranění lze maximálně 20 minut před začátkem dané směny.

Zaměstnanec

Nemůže si sám vytvářet, upravovat ani odstraňovat žádná zranění. V účtu zaměstnance je pouze možnost zobrazení historie zranění.

4.7 Správa směn

Směny se skládají ze dvou datumů, a to z datumu začátku a konce směny, dále se skládají z místa směny, důležitosti směny a z poznámky ke směně. Důležitost a poznámka ke směně jsou volitelné.
Důležitost směny může být následující:

- fatální,
- důležitá,
- normální,
- zaučení,
- nedůležitá,
- nespecifikováno.

Firma

Může směny vytvářet, odstraňovat a upravovat. Po smazání směny zmizí daná směna i z historie směn zaměstnanců, kteří danou směnu měli přiřazenou, taktéž zmizí z analytické části OLAP, pokud už ji nějaký zaměstnanec přiřazenou měl.

Směna nesmí být delší než 12 hodin. Firma po vytvoření směny může danou směnu přiřazovat zaměstnancům v seznamu směn anebo také může konkrétnímu zaměstnanci přiřazovat směny v seznamu zaměstnanců. V seznamu zaměstnanců lze přiřazovat zaměstnancům pouze směny budoucí, a to z důvodu zamezení zobrazování zbytečných směn, tedy směn, které už proběhly. Přiřadit zaměstnanci směnu z minulosti je možné v seznamu směn.

Zaměstnanec

Nespravuje směny, směny mu jsou přiděleny firmou. Zaměstnanec si může pouze zobrazit jaké směny ho čekají, taktéž si může zobrazit i historii všech jeho směn.

4.8 Správa docházek

Docházky se skládají ze 7 částí, a to z indikátorů příchodů, příchodů a odchodů zapisovanými firmami, příchodů a odchodů zapisovanými zaměstnanci, statusů a poznámek ke konkrétním docházkám zaměstnanců na konkrétních směnách. Docházky se vztahují na konkrétní směny konkrétních zaměstnanců. Docházky fungují na bázi priorit, nejnižší prioritou jsou příchody a odchody zapisované zaměstnanci, vyšší prioritou jsou příchody a odchody zapisované firmami a nejvyšší prioritou jsou statusy docházek. Status docházky může nabývat 5 různých hodnot, těmito hodnotami jsou:

- nemocný,
- nepřišel,
- odmítl,
- zpoždění,
- čekající,
- OK - používá se pouze, pokud je docházka v pořádku.

Pokud tedy není pro firmu čas příchodu a odchodu důležitý a chce používat správu docházek stylem zaměstnanec přišel, nebo nepřišel, a když nepřišel, tak z jakého důvodu, tak je to v tomto informačním systému zcela možné. Pakliže je příchod v pořádku, automaticky se vyplní status hodnotou OK. Pakliže je zapsaný příchod později než samotný začátek směny, tak se status vyplní hodnotou zpoždění. Z docházek jsou vypočítávány celkově odpracované hodiny, které jsou následně dostupné v analytické části OLAP. Celkově odpracované hodiny v rámci směny jsou prioritně vypočítávány na základě zápisu příchodu a odchodu firmy, pokud firma příchod, nebo odchod nevyplnila, tak se celkově odpracované hodiny počítají ze zapsaného příchodu a odchodu zaměstnanců.

Firma

Může vyplňovat docházky jednotlivým zaměstnancům nebo také jednotlivým směnám, pokud firma nevyplní příchod, ani odchod, tak je směna vedena jako směna s nezapsaným příchodem nebo odchodem, podobně to platí pro status konkrétní docházky. Pakliže firma nevyplní status docházky, tak samotný status zůstává ve stavu čekající.

V seznamu zaměstnanců lze vyplňovat docházku zaměstnancům pouze u směn, které jsou z aktuálního měsíce, a to z důvodu zamezení zobrazování zbytečných směn.

Zaměstnanec

Může si zapsat svůj příchod maximálně 20 minut před začátkem dané směny, zapsání odchodu není nijak omezeno.

4.9 Správa hodnocení

Hodnocení zaměstnanců se skládá ze 3 dílčích částí, a to ze:

- spolehlivosti,
- dochvilnosti,
- pracovitosti.

Každá část hodnocení má minimální hodnotu 0 a maximální hodnotu 5. Na základě vyplněných částí hodnocení je vypočítáváno celkové skóre zaměstnance. Celkové skóre zaměstnance je vypočítáváno jako průměr ze všech tří dílčích částí hodnocení.

Hodnotit zaměstnance je umožněno pouze těm uživatelům informačního systému s rolí firmy.

4.10 Správa zaměstnanců

Firma

Může zaměstnance vytvářet, upravovat a odstraňovat, dále jim může přiřazovat směny, taktéž je může hodnotit a vyplňovat jim docházku. Dále jim může přiřazovat jazyky, vytvářet jim dovolené, nemocenské, zranění a nahlášení. Firma si může nechat vygenerovat seznam zaměstnanců, nebo také aktuální směny konkrétního zaměstnance, či všechny zaměstnancovy směny, dále může spravovat Google Drive složku jednotlivých zaměstnanců.

Po vytvoření zaměstnance je zaměstnanci automaticky vytvořena složka na Google Drive, jejíž název je ve formátu zaměstnancova jména a příjmení. Při vytváření zaměstnance lze zvolit možnost pro automatické nasdílení vytvořené složky s emailovou adresou zaměstnance. Název Google Drive složky je vždy aktuální, neboť se synchronizuje s případnou změnou jména a příjmení v účtu zaměstnance. Firma také může spravovat profilové fotky zaměstnanců, tedy může je nahrávat, případně i odstraňovat.

Zaměstnanec

Spravuje pouze svůj účet, kde si může upravovat své údaje, včetně změny hesla. Dále si může zobrazit aktuální směny[14], kde si zaměstnanec zapisuje své odchody a příchody v rámci směn, dále si může zobrazit seznam všech směn, zranění na směnách a statistiky. Může vytvářet a odesílat žádosti o dovolené a nemocenské, případně může vytvářet a odesílat jednotlivá nahlášení. Po jejich vytvoření jsou pro firmu žádosti neviditelné, neboť ihned po vytvoření se nachází ve stavu nenahlášeno, či nezažádáno. Po zakliknutí tlačítka zažádat, či odeslat se stane dovolená, nemocenská, či nahlášení viditelné pro firmu. Zaměstnanec může jednotlivé dovolené, nemocenské a nahlášení upravovat a odstraňovat pouze, pokud se nachází ve stavu nezažádáno či nenahlášeno.

Dále si zaměstnanec může vygenerovat seznam jeho dovolených, nemocenských a nahlášení. Existuje také možnost vygenerovat si aktuální směny, historii směn, údaje zaměstnance a seznam jeho zranění. Zaměstnanec si taktéž může nahrát či odstranit svou stávající profilovou fotku. Taktéž má možnost si případně smazat i celý svůj účet.

[14] Aktuální směny jsou všechny směny od pondělí do neděle v daném týdnu.

4.11 Statistiky

Admin

Může si zobrazit celkový počet firem, zaměstnanců, obsazených směn, budoucích směn (obsazených) a počet vypsaných směn celkově, dále si může zobrazit počet nově zaregistrovaných firem dle měsíců, počet nových zaměstnanců firem dle měsíců, počet vypsaných směn dle měsíců a počet obsazených směn dle měsíců.

Sloupcové grafy mají možnost změnit rok, ve výchozím nastavení je vždy aplikován rok aktuální.

Firma

Může si zobrazit celkový počet zaměstnanců, vypsaných směn, obsazených směn a odpracovaných směn, dále si může zobrazit počet obsazených směn, nezapsaných docházek, směn s absencí, společně s počtem směn, které proběhly v pořádku, tedy bez absence, to znamená, že zaměstnanec se buďto dostavil na směnu včas, nebo měl případně zpoždění.

Taktéž si může zobrazit počet nových zaměstnanců dle měsíců, počet vypsaných směn dle měsíců, celkový počet zpoždění na směnách, počet nepříchodů na směnu, počet odmítnutí směn a počet nepříchodů kvůli nemoci. Dále se ve statistikách nachází průměrné celkové skóre zaměstnance, taktéž průměrná spolehlivost zaměstnance spolu s průměrnou dochvilností a pracovitostí zaměstnance.

Dále se v informačním systému nachází statistika, díky níž lze provádět analýzu docházky, neboť tato statistika zobrazuje počet směn, které proběhly v pořádku, počet směn se zpožděním, taktéž počet směn s nepříchody, včetně nepříchodů kvůli nemoci a nepříchodů na základě odmítnutí směny.

Dále se v rámci statistik nachází sekce, která byla vytvořena za účelem sledování počtu hodin směn, průměrné délky směny, nejdelší směny a taktéž směny nejkratší. Informační systém také nabízí statistiky o počtu celkově odpracovaných hodin a o počtu hodin absence způsobenými zpožděním zaměstnanců, dále se v systému nachází statistika o počtu celkových zpoždění, v rámci všech směn. Dále si lze zobrazit statistiku, která pojednává o průměrném skóre ze všech směn.

Dále je možné si zobrazit počet směn zaměstnanců dle měsíců, celkový počet hodin směn dle měsíců, počet odpracovaných hodin na směnách dle měsíců, počet celkových hodin zpoždění dle měsíců, počet zpoždění dle měsíců a počet zranění na směnách dle měsíců.

Dále mezi statistiky patří celkový počet zranění, nahlášení, dovolených, nemocenských, počet dovolených dle měsíců, počet nemocenských dle měsíců, počet nahlášení dle měsíců a vývoj průměrného skóre zaměstnanců v čase.

Sloupcové grafy mají možnost změnit rok, ve výchozím nastavení je vždy aplikován rok aktuální.

Firma si také může zobrazit statistiky konkrétního zaměstnance. U konkrétního zaměstnance si může zobrazit jeho spolehlivost, dochvilnost a pracovitost, spolu s jeho celkovým skóre. Dále se u zaměstnance nachází pravděpodobnost příchodu na směnu, která je vypočítávána na základě pouze těch směn s vyplněnou docházkou.

Mezi další statistiky konkrétního zaměstnance patří počet vypsaných hodin na aktuální týden, měsíc a celkově. K těmto statistikám jsou doplněny statistiky o počtu odpracovaných hodin za aktuální týden, měsíc a celkově. Dále je možné si zobrazit celkový počet hodin způsobený zpožděními na směnách, včetně celkového počtu zpoždění. Dále je možné si zobrazit zaměstnancovo průměrné skóre ze všech směn, počet celkově vypsaných směn,

počet směn s vyplněnou docházkou, počet směn nadcházejících, počet nepříchodů celkově, celkový počet zranění, dovolených a nemocenských.

V rámci statistik konkrétního zaměstnance je možné si dále zobrazit počet směn dle měsíců, celkový počet hodin směn dle měsíců, počet celkově odpracovaných hodin dle měsíců, počet celkových hodin zpoždění dle měsíců, počet celkových zpoždění dle měsíců, počet zranění na směnách dle měsíců a vývoj průměrného skóre zaměstnance v čase.

Zaměstnanec

Může si zobrazit počet dovolených, nemocenských, zranění, nahlášení, směn, budoucích směn a počet absencí. Mezi další statistiky patří počet vypsaných hodin směn pro aktuální týden, měsíc a počet vypsaných celkově. Dále se mezi statistikami nachází počet odpracovaných hodin pro aktuální týden, měsíc a počet odpracovaných hodin celkově. Dále si může zobrazit počet celkových zpoždění a počet hodin způsobenými zpožděními.

Dále je možné si v rámci účtu zaměstnance zobrazit počet směn dle měsíců, celkový počet hodin směn dle měsíců, počet celkově odpracovaných hodin dle měsíců, počet celkových hodin způsobenými zpožděními dle měsíců, počet celkových zpoždění dle měsíců a počet zranění na směnách dle měsíců.

4.12 Google Drive

Používání Google Drive je v rámci informačního systému volitelné. Struktura každé Google Drive složky firmy je koncipována tak, že obsah složky samotné, tvoří složky zaměstnanců. Ve složkách zaměstnanců lze mít k zaměstnanci konkrétní soubory. Do složky zaměstnance nemá přístup pouze firma, nýbrž i zaměstnanec, potenciálně i admin. Zaměstnanec vždy vidí pouze a jen svou složku v rámci Google Drive složky firmy, nevidí složky ostatních zaměstnanců. Tuto strukturu bylo možné vybudovat díky identifikátorům Google Drive složek. Každá složka, či soubor mají totožnou část URL, tato část je následující:

```
https://drive.google.com/drive/u/3/folders/
```

Poté při vytvoření složky, či souboru se k této URL připojí pouze ID složky, či souboru, ID má například tento tvar:

```
1K6HNaBjU9URQCeD3jr_3QbRBPlIWjI1D
```

Tedy výsledná URL k této složce či souboru by byla následující:

```
https://drive.google.com/drive/u/3/folders/1K6HNaBjU9URQCeD3jr_3QbRBPlIWjI1D
```

Admin

Může spravovat celou strukturu Google Drive.

Firma

Má možnost nahrávání souborů do své Google Drive složky, a to buď přímo do Google Drive složky firmy, nebo do podsložek Google Drive složky firmy 1. úrovně, taktéž má firma možnost vytváření a odstraňování složek (pouze z Google Drive složky firmy, ne z podsložek této složky). Složky se vytvářejí pouze do složky Google Drive firmy, ne do jejich podsložek. Taktéž je přítomna možnost odstraňování souborů (pouze z Google Drive složky firmy, ne z podsložek této složky). V informačním systému, v účtu firmy, se vyskytuje tlačítko propojující informační systém s Google Drive složkou konkrétní firmy.

Zaměstnanec

Má možnost nahrávání souborů do své Google Drive složky, nebo do podsložek zaměstnanecké složky 1. úrovně. Zaměstnanec může také vytvářet, odstraňovat složky (pouze z Google Drive složky zaměstnance, ne z podsložek této složky). Složky se vytvářejí pouze do složky zaměstnance, ne do jejich podsložek, taktéž je přítomna možnost odstraňování souborů (pouze z Google Drive složky zaměstnance, ne z podsložek této složky). V informačním systému v účtu zaměstnance se vyskytuje tlačítko propojující informační systém s Google Drive složkou konkrétního zaměstnance.

4.13 Generování souborů

V informačním systému lze generovat soubory ve formátu PDF.

Admin

Může si vygenerovat seznam firem.

Firma

Může si vygenerovat seznam zaměstnanců, směn a souhrn hodnocení, dále si může vygenerovat aktuální směny konkrétního zaměstnance, všechny směny konkrétního zaměstnance, zaměstnance v konkrétní směně a své údaje.

Zaměstnanec

Může si vygenerovat seznam svých dovolených, nemocenských, nahlášení a zranění. Dále si zaměstnanec může vygenerovat své aktuální směny a svou historii směn, taktéž si může vygenerovat své údaje.

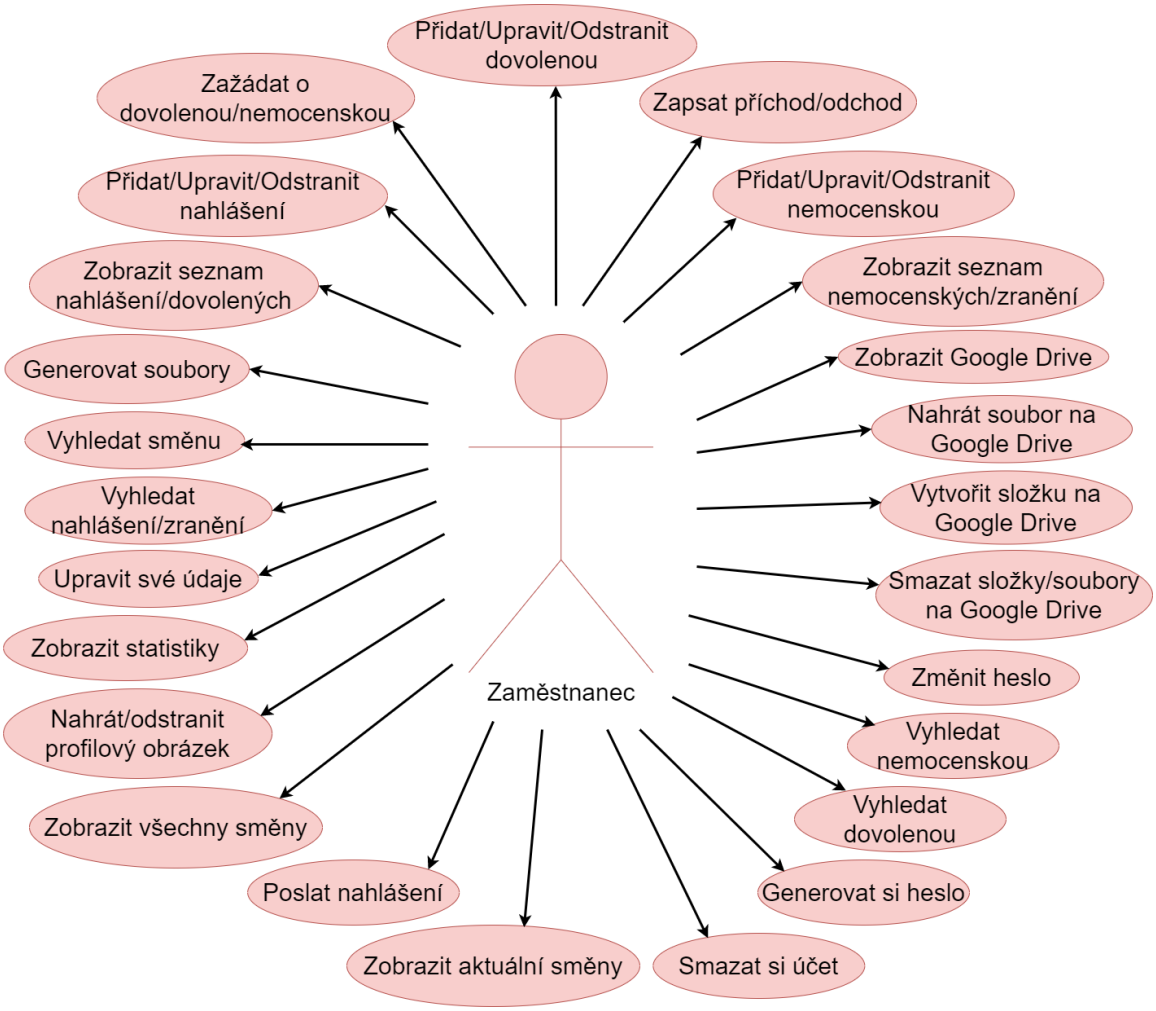
4.14 Diagram případů užití

Diagram případů užití (Use Case Diagram) zobrazuje chování systému z pohledu koncového uživatele. Účelem diagramu je popsat funkcionalitu systému. Diagram popisuje, jaké funkce má systém obsahovat, ale už nám neříká, jak konkrétně tyto funkce budou implementovány. Většinou první diagramem, který se při návrhu informačního systému vytváří je právě Diagram případů užití z důvodu prvotního ujasnění si, co vlastně od systému jako takového očekáváme [83].

Diagram případů užití se skládá ze samotných případů užití (use case), aktérů (actors) a vztahů mezi nimi [83].

4.14.1 Role zaměstnance

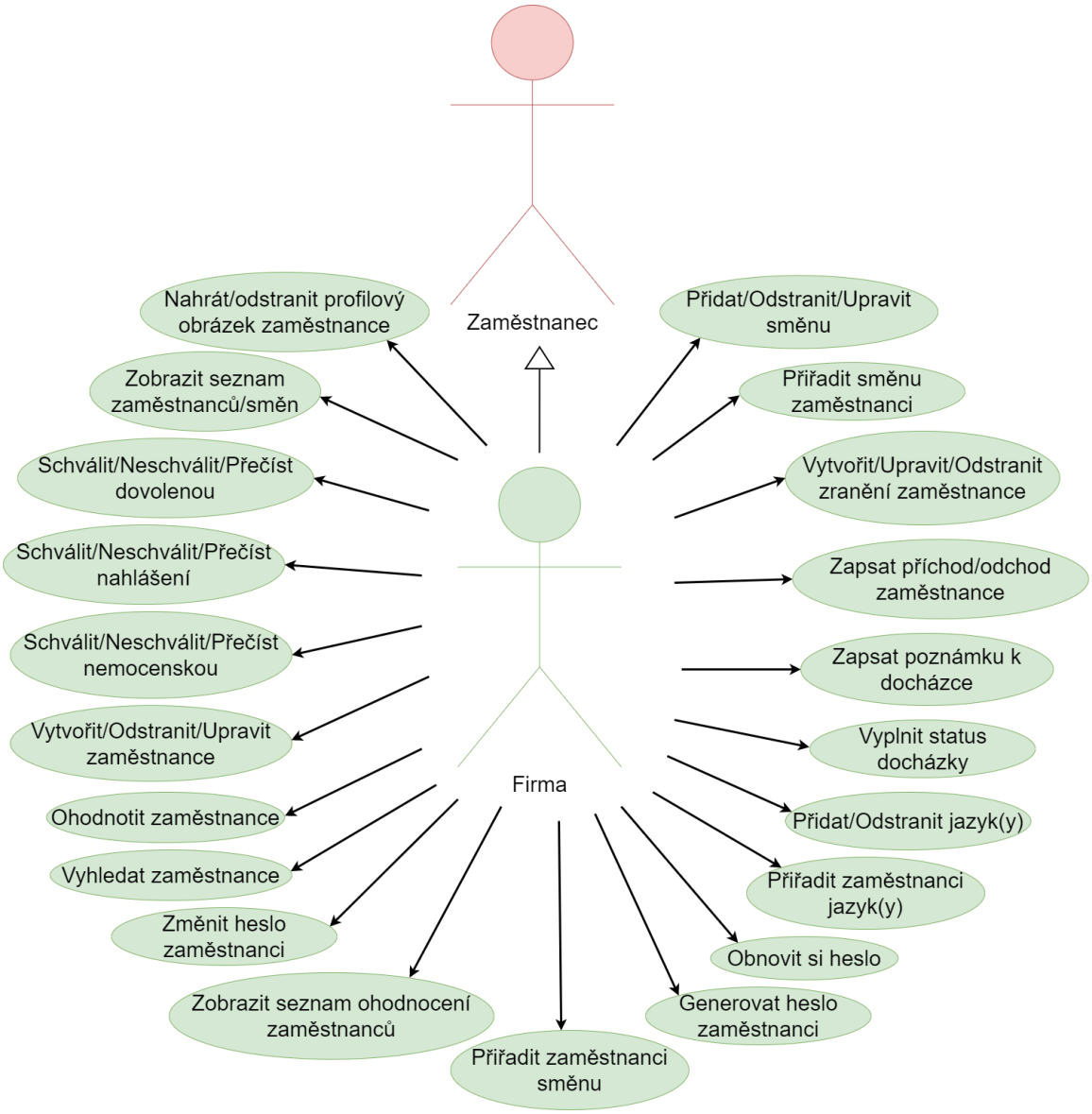
Jednotlivé případy užití v rámci role zaměstnance jsou popsány výše.



Obrázek 4.1: Diagram případu užití pro roli zaměstnance, vytvořil: autor.

4.14.2 Role firmy

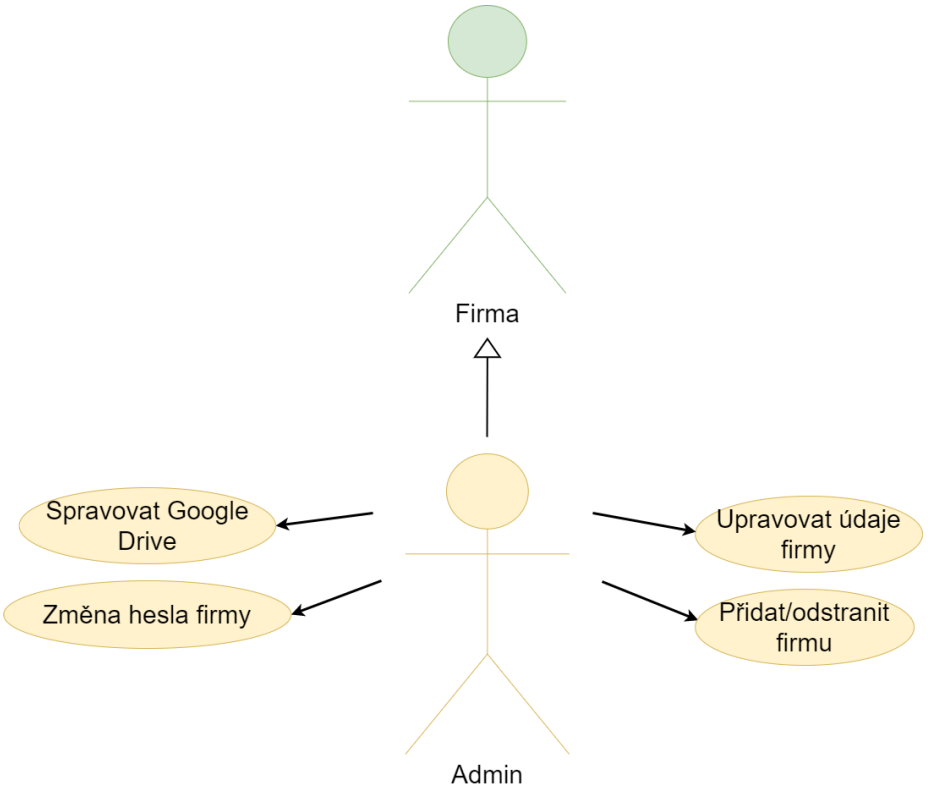
Jednotlivé případy užití v rámci role firmy jsou popsány výše.



Obrázek 4.2: Diagram případu užití pro roli firmy, vytvořil: autor.

4.14.3 Role admina

Jednotlivé případy užití v rámci role admina jsou popsány výše.



Obrázek 4.3: Diagram případu užití pro roli admina, vytvořil: autor.

Kapitola 5

Návrh systému

Tato kapitola se zabývá datovým návrhem aplikace, tedy existují požadavky na data a pro ony data je nutné vytvořit konceptuální schéma (ER Diagram), dále je také nutné vytvořit databázové schéma pro analytickou část OLAP.

5.1 ER diagram

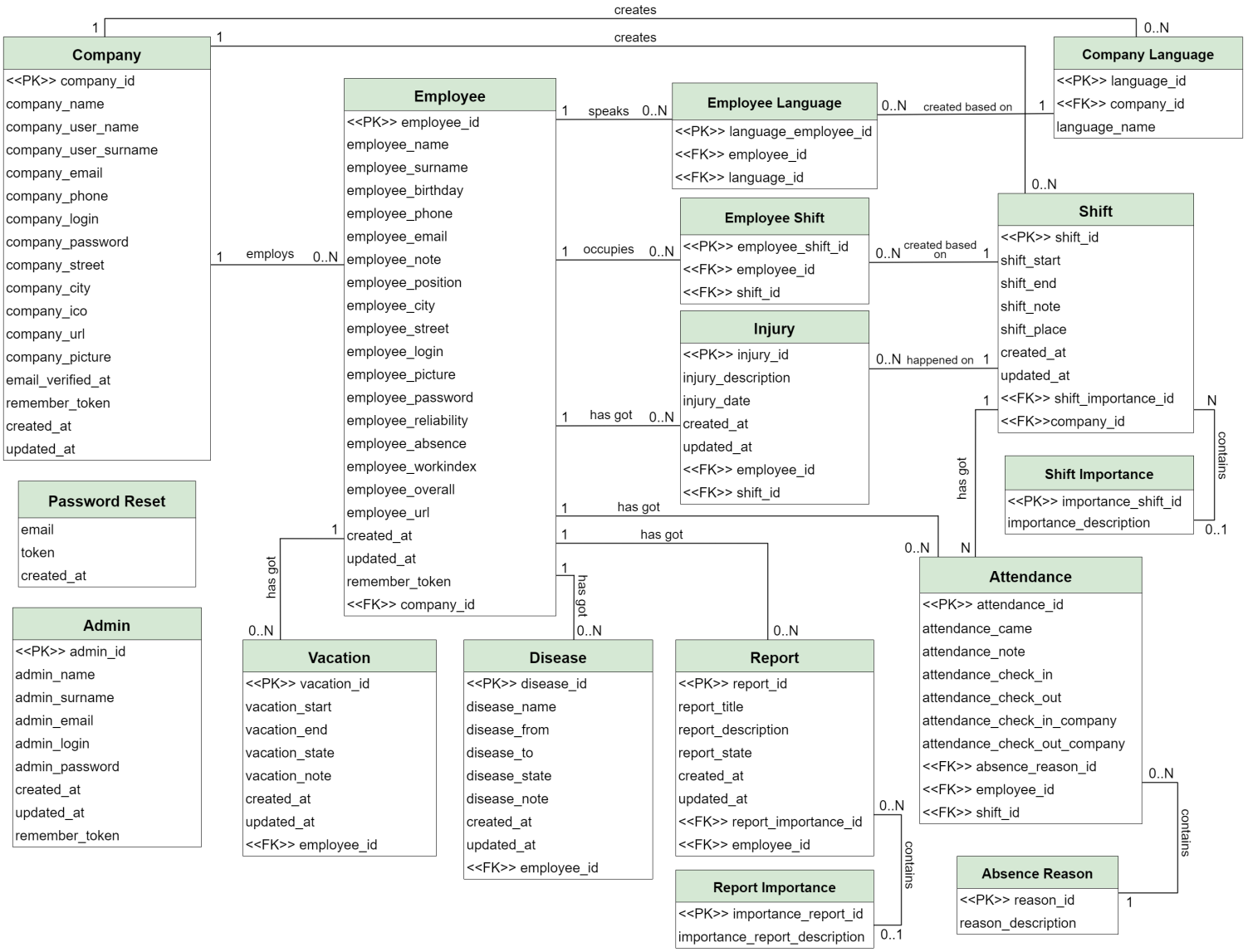
ER diagram, v plném znění Entity Relationship diagram, slouží k návrhu databáze [18]. ER diagram se používá k vyobrazení jednotlivých entitních množin a vztahů mezi nimi. Entitní množina je takový objekt reálného světa, o kterém chceme uchovávat data v databázi [77].

Entitní množinu si lze také představit jako složku pro konkrétní typ dat [27]. Entitní množinou může být například firma, zaměstnanec, učitel nebo žák.

Každá entitní množina obsahuje atributy, které popisují vlastnosti dané entitní množiny nebo vztahu, které nás zajímají [77]. Například atributy u entitní množiny zaměstnance by se mohli skládat z křestního jména, příjmení, pozice a adresy bydliště.

ER diagram spadá do konceptuálního modelování, po dokončení ER diagramu lze pokračovat další fází v rámci návrhu databáze, která se zaobírá návrhem logickým (tabulky), poté následuje fáze poslední, která se zaobírá návrhem fyzickým [77].

ER diagram se tedy skládá ze tří základních elementů, konkrétně z entitních množin, atributů a vztahů mezi jednotlivými entitními množinami [18].



Obrázek 5.1: ER diagram, vytvořil: autor.

5.1.1 Admin

Entitní množina Admin slouží pro ukládání dat o adminech a je přímo svázaná s rolí admina. Admin se do informačního systému může přihlásit pomocí emailové adresy (`admin_email`). Emailová adresa každého admina musí být, v rámci entitní množiny Admin, unikátní. Admin se do informačního systému může přihlásit i pod svým uživatelským jménem (`admin_login`). Uživatelské jméno musí taktéž splňovat unikátnost, v rámci dané entitní množiny. V rámci přihlašování je nutné zadat také heslo, k tomu slouží atribut `admin_password`.

5.1.2 Company

Entitní množina Company, dále v textu uváděná jako Firma, je entitní množinou sloužící pro ukládání dat o firmách a je přímo svázaná s rolí firmy v informačním systému. Firma se do informačního systému může přihlásit pomocí emailové adresy (`company_email`). Emailová adresa každé firmy musí být, v rámci entitní množiny Firma, unikátní. Firma se do informačního systému může přihlásit i pod svým uživatelským jménem (`company_login`). Uživatelské jméno musí taktéž splňovat unikátnost v rámci dané entitní množiny. V rámci přihlašování je nutné zadat také heslo, k tomu slouží atribut `company_password`.

Atribut `company_url` slouží pro ukládání identifikátorů Google Drive složek firem.
Atribut `company_picture` slouží pro ukládání názvů profilových fotek firem.
Atribut `email_verified_at` slouží k ověření emailové adresy daných firem.

Atributy IČO (`company_ico`) a ulice sídla (`company_street`) jsou při vytváření nového účtu firmy, v rámci informačního systému, nepovinné.

5.1.3 Employee

Entitní množina Employee, dále v textu uváděná jako Zaměstnanec, je entitní množinou sloužící pro ukládání dat o jednotlivých zaměstnancích konkrétních firem a je přímo svázaná s rolí zaměstnance v informačním systému. Zaměstnanec se do informačního systému může přihlásit pomocí emailové adresy (`employee_email`). Emailová adresa každého zaměstnance musí být, v rámci entitní množiny Zaměstnanec, unikátní. Zaměstnanec se do informačního systému může přihlásit i pod svým uživatelským jménem (`employee_login`). Uživatelské jméno musí taktéž splňovat unikátnost v rámci dané entitní množiny. V rámci přihlašování je nutné zadat také heslo, k tomu slouží atribut `employee_password`.

Atribut `employee_picture` slouží pro ukládání názvů profilových fotek zaměstnanců.
Atribut `employee_reliability` slouží pro ukládání hodnoty spolehlivosti zaměstnance.
Atribut `employee_absence` slouží pro ukládání hodnoty dochvilnosti zaměstnance.
Atribut `employee_workindex` slouží pro ukládání hodnoty pracovitosti zaměstnance.
Atribut `employee_overall` slouží pro ukládání hodnoty celkového skóre zaměstnance.
Atribut `employee_url` slouží pro ukládání identifikátorů Google Drive složek zaměstnanců.

Entitní množina Zaměstnanec také obsahuje atribut `company_id`, jak už sám název atributu napovídá, jedná se o cizí klíč, díky kterému víme, k jakým konkrétním firmám zaměstnanci patří.

Atributy poznámka k zaměstnanci (`employee_note`), ulice bydliště (`employee_street`) a datum narození (`employee_birthday`) jsou při vytváření nového účtu zaměstnance, v rámci informačního systému, nepovinné.

5.1.4 Password Reset

Entitní množina Password Reset, dále v textu uváděná jako Reset Hesla, je entitní množinou sloužící k resetování hesla. Reset Hesla se váže pouze na roli firmy.

5.1.5 Vacation, Disease a Report

Entitní množina Vacation, dále v textu uváděná jako Dovolená, je entitní množinou sloužící pro ukládání dat o dovolených jednotlivých zaměstnanců.

Atribut `vacation_state` je detailněji popsán v sekci 4.3.

Z důvodu vztahu entitní množiny Dovolená k entitní množině Zaměstnanec je atributem entitní množiny Dovolená také cizí klíč (`employee_id`), neboť jeden zaměstnanec může mít vícero dovolených, díky tomuto atributu tedy víme, ke kterým dovoleným patří, jaký zaměstnanec.

Entitní množina Disease, dále v textu uváděná jako Nemocenská, je entitní množinou sloužící pro ukládání dat o nemocenských jednotlivých zaměstnanců.

Atribut `disease_name` slouží pro ukládání názvů jednotlivých nemocí.
Atribut `disease_state` je detailněji popsán v sekci 4.3.

Z důvodu vztahu entitní množiny Nemocenská k entitní množině Zaměstnanec je atributem entitní množiny Nemocenská také cizí klíč (`employee_id`), neboť jeden zaměstnanec může mít vícero nemocenských, díky tomuto atributu tedy víme, ke kterým nemocenským patří, jaký zaměstnanec.

Entitní množina Report, dále v textu uváděná jako Nahlášení, je entitní množinou sloužící pro ukládání dat o nahlášení jednotlivých zaměstnanců.

Atribut `report_title` slouží pro ukládání názvů jednotlivých nahlášení.
Atribut `report_description` slouží pro ukládání popisů jednotlivých nahlášení.
Atribut `report_state` je detailněji popsán v sekci 4.3.

Z důvodu vztahu entitní množiny Nahlášení k entitní množině Zaměstnanec je atributem entitní množiny Nahlášení také cizí klíč (`employee_id`), neboť jeden zaměstnanec může mít vícero nahlášení, díky tomuto atributu tedy víme, ke kterým nahlášení patří, jaký zaměstnanec.
V entitní množině Nahlášení také existuje další cizí klíč (`report_importance_id`), díky kterému lze přiřazovat jednotlivým nahlášení důležitosti, které jsou v informačním systému předem nadefinované v entitní množině Report Importance (viz sekci 4.4).

5.1.6 Injury

Entitní množina Injury, dále v textu uváděná jako Zranění, je entitní množinou sloužící pro ukládání dat o zranění jednotlivých zaměstnanců na konkrétních směnách.

5.1.7 Employee Language a Company Language

Entitní množina Company Language, dále v textu uváděná jako Firemní Jazyk, je entitní množinou sloužící pro ukládání dat o jednotlivých jazycích nadefinovanými firmami. V entitní množině Firemní Jazyk se vyskytuje cizí klíč (`company_id`), díky kterému lze přiřadit konkrétním firmám pouze ty jazyky, které daná firma nadefinovala.

Entitní množina Employee Language, dále v textu uváděná jako Zaměstnanecký Jazyk, je entitní množina sloužící pro ukládání dat o jednotlivých jazykových schopnostech zaměstnanců firem. V entitní množině Zaměstnanecký Jazyk se vyskytují dva cizí klíče (`employee_id` a `language_id`), díky kterým lze zaměstnancovi přiřadit konkrétní jazyk.

5.1.8 Shift

Entitní množina Shift, dále v textu uváděná jako Směna, je entitní množinou sloužící pro ukládání dat o jednotlivých směnách konkrétních firem.

Atribut `shift_place` slouží pro ukládání lokací jednotlivých směn.

V entitní množině Směna se vyskytují dva cizí klíče. První slouží k přiřazení konkrétní směny ke konkrétní firmě (`company_id`) a druhý slouží k přiřazení důležitosti směny ke konkrétní směně (`shift_importance_id`).

5.1.9 Employee Shift

Entitní množina Employee Shift, dále v textu uváděná jako Zaměstnancova Směna, je entitní množinou sloužící pro ukládání dat o jednotlivých přiřazených směnách konkrétním zaměstnancům.

V entitní množině Směna se vyskytují dva cizí klíče. První slouží k přiřazení konkrétního zaměstnance (`employee_id`) a druhý slouží k přiřazení konkrétní směny (`shift_id`), tedy dohromady určují jací zaměstnanci obsadili, jaké směny.

5.1.10 Attendance

Entitní množina Attendance, dále v textu uváděná jako Docházka, je entitní množinou sloužící pro ukládání dat o jednotlivých docházkách konkrétních zaměstnanců na konkrétních směnách konkrétních firem.

Atribut `attendance_came` slouží indikaci, zdali zaměstnanec na danou směnu přišel.

Atributy `attendance_check_in` a `attendance_check_out` slouží k zapsání příchodu a odchodu zaměstnancem.

Atributy `attendance_check_in_company` a `attendance_check_out_company` slouží k zapsání příchodu a odchodu zaměstnance firmou.

V entitní množině Docházka se vyskytují tři cizí klíče. První slouží k přiřazení důvodu absence k docházce (`absence_reason_id`), druhý pro přiřazení konkrétního zaměstnance k docházce (`employee_id`) a třetí k přiřazení konkrétní směny k docházce (`shift_id`).

5.2 OLAP

Analýza OLAP funguje nad přiřazenými zaměstnaneckými směnami konkrétních firem, tedy samotná analýza se vždy pojí s konkrétní směnou (včetně její docházky), zaměstnancem a firmou.

Dohromady existují čtyři dimenze, těmito dimenzemi jsou: Shift Info Dimension, Time Dimension, Employee Dimension a Company Dimension.

První dimenze, tedy Shift Info Dimension se pojí na konkrétní směnu, včetně její docházky. Od dimenze Shift Info Dimension je odvozena druhá dimenze s názvem Time Dimension, která slouží pro extrakci dnu (`day`), měsíce (`month`), čtvrtletí (`quarter`) a roku (`year`) konkrétní směny. Time Dimension obsahuje také jednoznačný identifikátor (`time_id`), díky kterému lze zjistit, k jakým směnám se jednotlivé záznamy v dimenzi Time Dimension pojí.

Třetí dimenze, tedy Employee Dimension slouží pro ukládání dat o zaměstnancích, kteří mají přiřazenou alespoň jednu směnu, pokud zaměstnanci byla směna odebrána a byla zároveň jeho směnou jedinou, je z dimenze následně odstraněn.

Čtvrtá a poslední dimenze, tedy Company Dimension slouží pro ukládání dat o firmách, kteří přiřadili alespoň jednomu zaměstnanci libovolnou směnu. Pokud firma smaže všechny své zaměstnance ze směn, tak je následně odebrána z dimenze Company Dimension.

Za účelem analýzy vznikla tabulka faktů (Shift Facts), která obsahuje data, která jsou v rámci analýzy směn a docházky důležitá pro účely informačního systému. Tabulka faktů obsahuje délku směny (`shift_total_hours`), počet odpracovaných hodin na směně (`total_worked_hours`), počet hodin zpoždění na směně (`late_total_hours`), dále tabulka faktů obsahuje příznaky, díky kterým lze zjistit, zdali došlo na dané směně k zranění nebo zpoždění (`employee_injury_flag` a `employee_late_flag`). Dále obsahuje průměrné skóre zaměstnance (`employee_overall`), v době, kdy mu byla směna přiřazena. Tabulka faktů také obsahuje indikátor, který slouží pro indikaci, zdali zaměstnanec na směnu přišel (`attendance_came`) a důvod absence k dané směně (`absence_reason`). Počet hodin není uložen ve formátu celých čísel, nýbrž ve formátu čísel desetinných, tedy v OLAP sekci systému je například 30 minut rovno 0,5 hodiny. S tabulkou faktů mohou manipulovat pouze účty s rolí firmy, tedy například atribut `total_worked_hours` společně s atributem `late_total_hours` se pojí pouze na docházky zapsanými firmami.

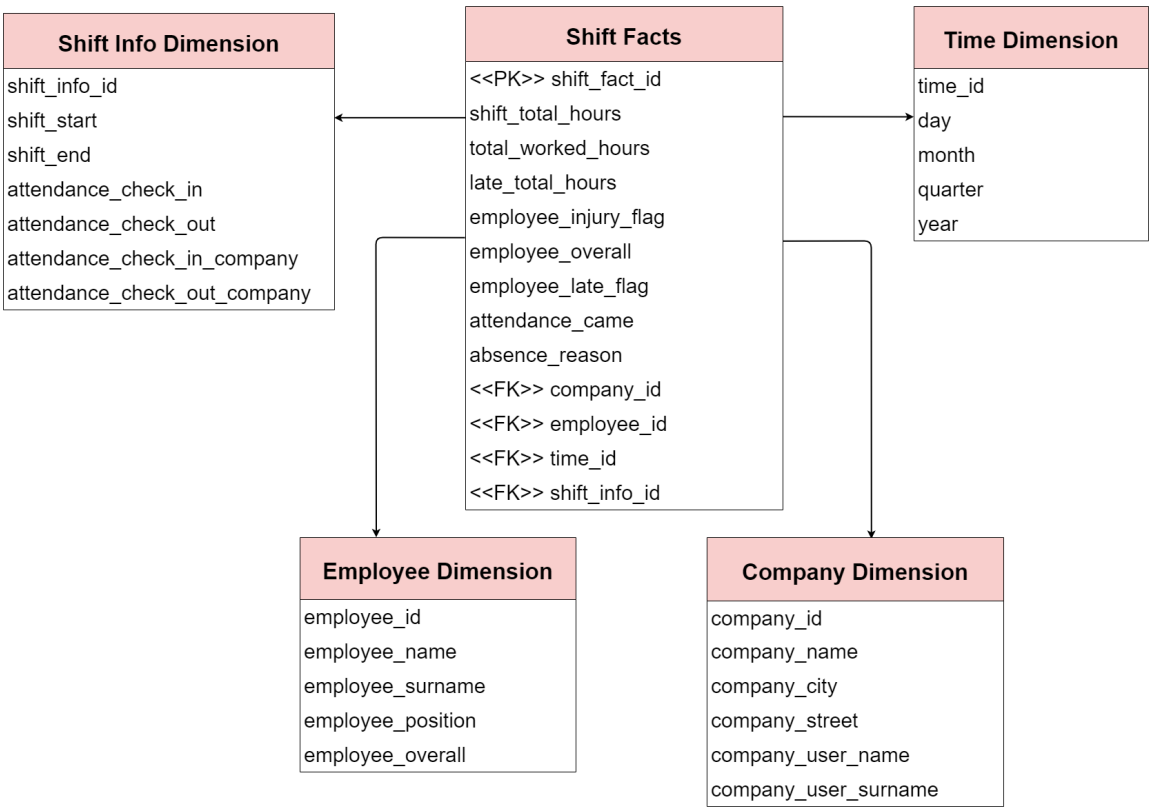
Do OLAP sekce systému je potřeba jednotlivé data nějakým způsobem nahrát, v této práci jsou použity procesy ETL (Extract, Transform, Load), při kterých se data z OLTP sekce systému uloží do OLAP sekce systému. Tento proces ukládání je spuštěn automaticky po přiřazení libovolné směny libovolnému zaměstnanci.

Prvním krokem je samotná extrakce dat (Extract), to v praxi znamená, že z OLTP sekce systému si vezmeme jen ty atributy entitních množin, které nás zajímají, a které chceme uchovávat v OLAP sekci systému. Po uložení dat do dimenzí a tabulky faktů (pouze u těch atributů, u kterých není potřeba transformace) je proveden druhý krok, kterým je vykonání transformací v tabulce faktů Shift Facts (Transform), konkrétně pro výpočet délky směny, počtu odpracovaných hodin na konkrétní směně a počtu hodin zpoždění na konkrétní směně.

Posledním krokem každého procesu ETL je Load, tedy uložení dat samotných do celé OLAP sekce systému. Data, která nejsou k dispozici se v OLAP sekci systému uloží s hodnotou NULL.

OLAP sekce systému je automaticky zaktualizována po jakékoliv změně v OLTP sekci systému, která se pojí s OLAP sekcí systému. Pokud se jedná o změnu, po které je potřeba znovu provést transformace v tabulce faktů, tak se nejprve daná data extrahují a až poté se znovu vykonají dané transformace nad daty novými.

Schéma OLAP je vyobrazeno na obrázku 5.2, jedná se o schéma hvězdy.



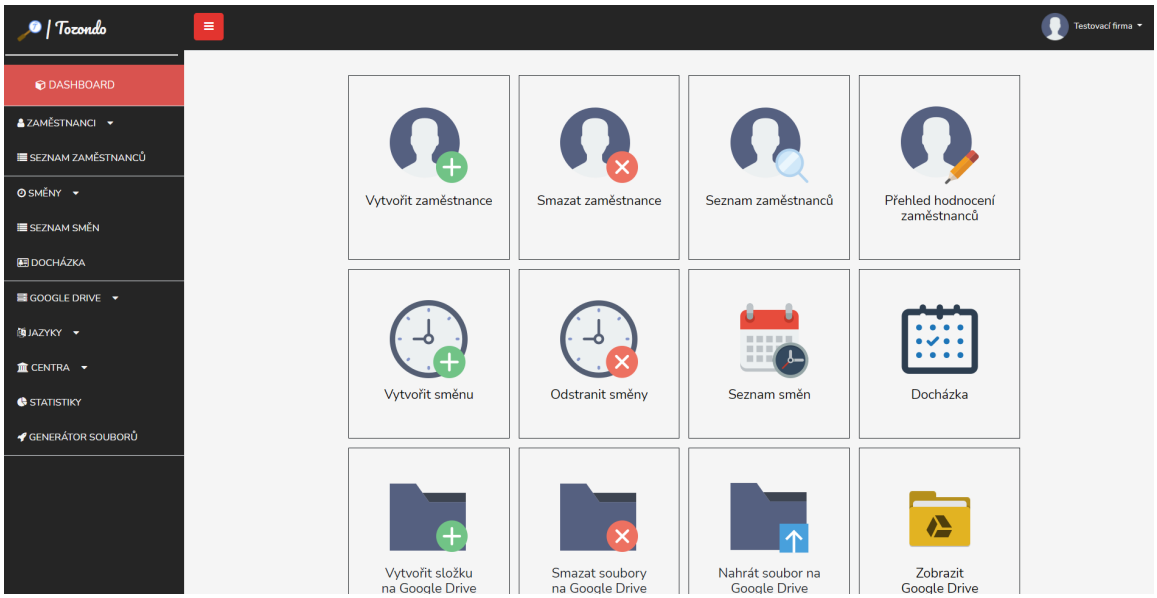
Obrázek 5.2: Schéma databáze pro OLAP sekci informačního systému, vytvořil: autor.

5.3 Návrh uživatelského rozhraní

Nedílnou součástí každé webové aplikace je její uživatelské rozhraní. Uživatelské rozhraní by mělo být snadno pochopitelné, efektivní a hlavně přehledné. Tyto tři aspekty dohromady tvoří celkový dojem z webové aplikace, proto je nutné se uživatelskými rozhraními zabývat.

V informačním systému pro správu zaměstnanců byl vytvořen tzv. Dashboard. V Dashboardu se vyskytují všechny funkcionality informačního systému. Dashboard je domovskou stránkou informačního systému. Dashboard byl postaven na existující základní šabloně Simple Sidebar[15]. Šablona byla značně upravena a změněna pro účely informačního systému.

Domovská stránka informačního systému je vyobrazena na obrázku 5.3.



Obrázek 5.3: Ukázka domovské stránky informačního systému pro správu zaměstnanců, vytvořil: autor.

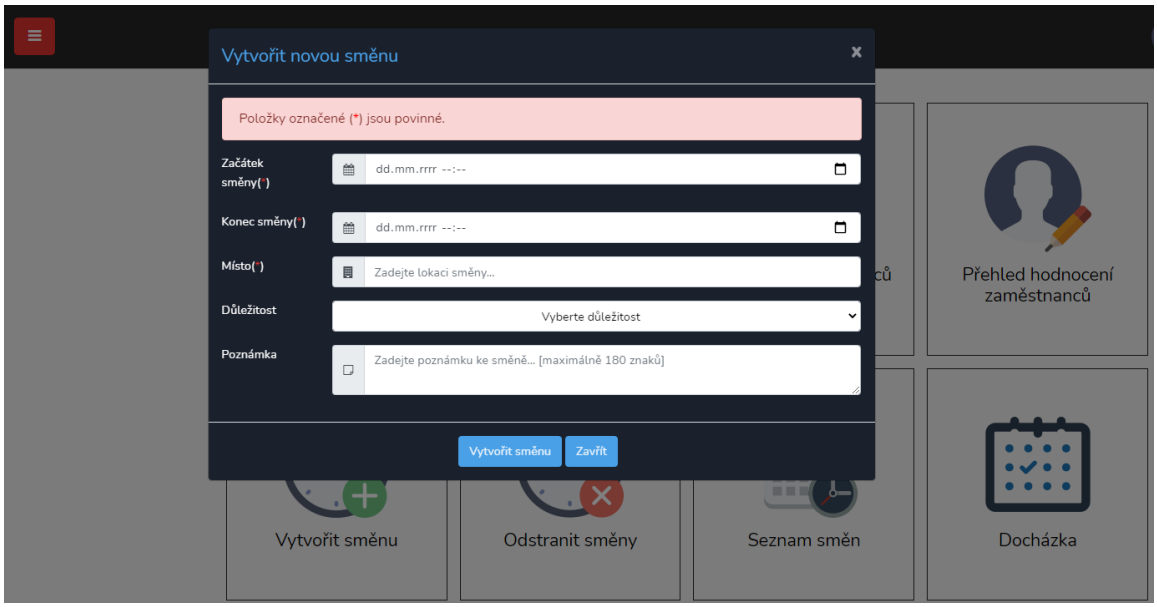
V levé části domovské stránky se nachází postranní panel. V prostřední části se nachází samotné funkcionality informačního systému a v horní pravé části se nachází rozklikávací kontextové menu, v kterém je možnost si zobrazit profil účtu nebo se odhlásit z informačního systému.

[15] Dostupné z: https://startbootstrap.com/template/simple-sidebar.

5.4 Postranní panel a funkcionality

Po stisknutí na jednotlivé funkcionality je uživatel buďto přesměrován na webovou stránku dané funkcionality nebo se mu zobrazí tzv. modální okno. Modální okna se typicky týkají vytváření a odstraňování libovolného obsahu v rámci informačního systému.

Ukázkou modálního okna se zabývá obrázek 5.4.



Obrázek 5.4: Ukázka modálního okna při vytváření směny, vytvořil: autor.

Postranní panel byl vytvořen pro zjednodušení práce s informačním systémem. Díky němu se lze přepínat do různých sekcí systémů, nebo vyvolat modální okna i mimo domovskou stránku. Postranní panel lze skrývat či odkrývat za pomocí tlačítka, to je zejména důležité pro ty uživatele, kteří používají informační systém z mobilních zařízení.

Kapitola 6

Implementace

Tato kapitola se zabývá nejdůležitějšími sekcemi implementace informačního systému pro správu zaměstnanců. Implementaci předchází ona analýza a specifikace požadavků společně s návrhem systému, na jejichž základě lze samotnou implementaci realizovat.

6.1 Adresářová struktura

Framework Laravel je dodáván s určitou adresářovou strukturou. Adresářovou strukturu projektu je možné libovolně měnit, ale za účelem budoucí správy adresářové struktury je lepší dodržovat zavedené konvence frameworku Laravel.

Mezi nejdůležitější soubory v rámci kořenového adresáře patří:

- **.env** - jedná se o soubor obsahující proměnné prostředí. Umožňuje například nastavení přihlašovacích údajů pro připojení k databázi, a také k mailovým serverům. Dále umožňuje webovou aplikaci pojmenovat a přiřadit ji URL.
- **artisan** - jedná se o samotný nástroj Artisan.
- **composer.json** a **composer.lock** - jedná se o konfigurační soubory nástroje Composer.
- **package.json** a **package-lock.json** - jedná se o konfigurační soubory nástroje npm.
- **server.php** - jedná se o lokální webový server, který slouží pouze k vývoji webové aplikace. Webový server se spustí příkazem `php artisan serve`.

Mezi nejdůležitější složky kořenového adresáře patří [17]:

- **app** - obsahuje samotné jádro, ve smyslu kódu, webové aplikace. Tato složka obsahuje tři důležité složky.

  První složkou je složka Http obsahující Controllery, Middlewary a požadavky na formuláře.

  Druhou složkou je složka Models obsahující jednotlivé třídy modelů. Každý model koresponduje s nějakou tabulkou v databázi. Modely obecně slouží pro manipulace s tabulkami. Dovolují například realizovat dotazy a vkládat nové záznamy.

  Třetí složkou je složka Console obsahující které lze používat v rámci nástroje Artisan.
- **config** - obsahuje konfigurační soubory webové aplikace.

- **database** - obsahuje databázové migrace, seedy a továrny, které jsou svázány s jednotlivými modely.
- **resources** - obsahuje všechny pohledy (views).
- **routes** - obsahuje veškeré nadefinované cesty, které slouží k samotnému směrování v rámci webové aplikace.
- **public** - obsahuje soubor index.php, který slouží jako počáteční bod pro všechny požadavky vstupující do webové aplikace. Tato složka také slouží k ukládání různých obrázků a CSS nebo Javascript souborů.
- **vendor** - obsahuje závislosti nástroje Composer.

6.2 Databáze

Před samotnou prací s databází je nejprve nutné se k ní připojit. Připojení k dané databázi umožňuje soubor **.env**, který byl popsán výše.

Ve webové aplikaci se nachází dohromady 20 modelů. Každý model se váže na jednu tabulku databáze. V rámci každého modelu je nutné vyplnit jednotlivé atributy daných tabulek, aby s nimi bylo možné manipulovat. V modelech se také vyskytují funkce, které realizují dotazy na databázi.

Následně Controllery používají ony modely, které potřebují, a volají z nich jednotlivé funkce pro získání požadovaných dat. Příkladem funkce může být:

```
public static function ziskatDovolene($id_zamestnance){
     return DB::table('table_vacations')
            ->where(['table_vacations.employee_id' => $id_zamestnance])
            ->orderBy('table_vacations.vacation_start', 'asc')
            ->get();
}
```

Příkladem volání funkce modelu v Controlleru může být:

```
$dovolene = Vacation::ziskatDovolene($uzivatel->id_zamestnance);
```

Po provedení tohoto řádku kódu bude proměnná `$dovolene` obsahovat všechny dovolené konkrétního zaměstnance, který byl určen na základě jeho identifikátoru.
V rámci Controllerů lze nad tabulkami provádět i operace INSERT, UPDATE a DELETE, to vše za pomocí modelů.

6.3 Autentizace a autorizace

Pro účel používání informačního systému je nutné se do samotného informačního systému přihlásit. Přihlašování je realizováno formou formuláře, který obsahuje textové pole sloužící pro emailovou adresu či uživatelské jméno a textové pole pro heslo.

V rámci autentizace a autorizace byl použit autorizační a autentifikační balíček, který je součástí frameworku Laravel. Pro instalaci tohoto balíčku je vyžadováno použít příkazovou řádku, přičemž je nutné se nacházet v kořenovém adresáři webové aplikace, kde se nachází onen nástroj `artisan`, poté stačí zadat příkaz `php artisan ui vue --auth`.

Po úspěšné instalaci balíčku webová aplikace bude obsahovat jednotlivé cesty (routes) sloužící k autorizaci a autentizaci.

Dále bude vytvořena složka Auth ve složce Controllers. Mezi nejdůležitější soubory v této složce patří [15]:

- **LoginController** - stará se o samotnou autentizaci.
- **ForgotPasswordController** - stará se o emailove zprávy za účelem obnovy hesla.
- **RegisterController** - stará se o nové registrace uživatelů.
- **ResetPasswordController** - stará se o samotnou obnovu hesla za pomocí formuláře.
- **VerificationController** - stará se o ověření emailových adres uživatelských účtů.

Jednotlivé soubory byly upraveny tak, aby splňovaly požadavky kladené na informační systém. Některé soubory musely být upraveny značně (LoginController a RegisterController), zatímco u některých došlo ke změnám poměrně malým nebo žádným.

Pro účely samotné autorizace a autentizace existují v databázi tři tabulky, které se přímo vážou na jednotlivé role informačního systému.

První tabulkou je `table_companies`, která se váže na roli firmy. Druhou tabulkou je `table_employees`, která se váže na roli zaměstnance a třetí tabulkou je `table_admins`, která se váže na roli admina.

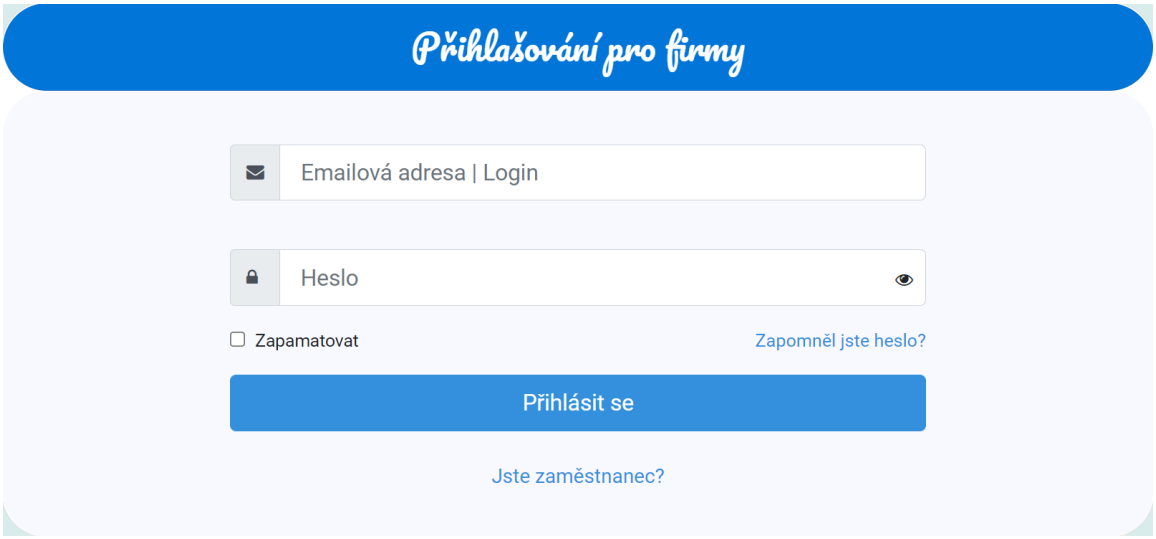
Kontroly přístupů k jednotlivým cestám (routes), v informačním systému, jsou realizovány pomocí Middlewarů, které definují, jaká role má právo na zobrazení, jakých částí informačního systému. Ve webové aplikaci existují tři Middlewary, a to pro firmu, zaměstnance a admina. Tímto je docíleno toho, že například si libovolný zaměstnanec nemůže zobrazit tabulku směn z účtu jeho firmy.

Příkladem autorizace cesty pro roli firmy v informačním systému může být následující kód:

```
Route::group(['middleware' => 'auth:firma'], function () {
    Route::get('/dovolene',[DovoleneController::class,'index']);
});
```

Cesta `/dovolene` tedy bude přístupna pro uživatele s rolí firmy. Pokud se libovolný uživatel s rolí zaměstnance pokusí načíst webovou stránku, která se na dané cestě nachází, tak bude přesměrován na stránku, kde se před zapsáním cesty nacházel.

Ukázka přihlašovacího formuláře je na obrázku 6.1. Pro zobrazení přihlašovacího formuláře pro roli admina je potřeba jít na adresu `www.tozondo.com/login/admin`.



Obrázek 6.1: Ukázka přihlašovacího formuláře, vytvořil: autor.

6.4 Vytváření uživatelů

V informačním systému se nachází registrační formulář pro firmy. Firmy mohou být zaregistrováni i manuálně adminem v seznamu firem. Zaměstnanci jsou do systému registrováni jejich firmami. Firma může vytvořit zaměstnance na domovské stránce nebo v seznamu zaměstnanců.

V registračním formuláři pro uživatele s rolí firmy se nachází možnost pro aktivaci Google Drive k danému účtu. Tedy používání Google Drive je volitelné, to se hodí zejména pro uživatele s jinou emailovou adresou než tou od společnosti Google (gmail). V případě aktivace Google Drive i přes nevlastnění emailové adresy od společnosti Google se sice vytvoří ona firemní složka, do které lze v rámci informačního systému například nahrávat soubory, ale nebude možné si ji zobrazit. V případě neaktivace Google Drive nebudou zobrazeny jednotlivé funkcionality, vytvořené pro manipulaci s Google Drive složkou firmy, na domovské stránce účtu s rolí firmy.

V registračním formuláři pro vytváření uživatelů s rolí zaměstnance se nachází možnost pro sdílení vytvořené složky zaměstnance s emailovou adresou zadanou ve formuláři. V případě zaškrtnutí této možnosti bude mít zaměstnanec možnost si zobrazit svou Google Drive složku, taktéž do ní bude moci nahrávat soubory, vytvářet nové složky, ale také bude mít možnost jednotlivé složky či soubory odstraňovat.

Jednotlivá hesla ať už zaměstnance, firmy nebo admina jsou před uložením do samotné databáze zahashována pomocí algoritmu Blowfish, který je využíván metodou `make` ze třídy `Hash`. Praktický příklad použití hashování může být následující:

```
$firma->heslo = Hash::make($request->heslo);
```

Heslo firmy je tímto zahashováno.

Vytváření firmy

Práci s registračním formulářem obstarává soubor `RegisterController.php`. Definice vzhledu registračního formuláře je popsána v souboru `register.blade.php`.

V registračním formuláři firma zadá její název, ulici (volitelné) a město sídla, IČO (volitelné), jméno a příjmení zástupce firmy, emailovou adresu, telefonní číslo, uživatelské jméno a heslo. Po stisknutí tlačítka *Registrovat* je zavolána metoda `validator` ze souboru `RegisterController.php`. V rámci této metody se provede validace zadaných údajů ve formuláři, pokud jsou zadané údaje validní, tak se pokračuje samotnou metodou `create`.

V rámci metody `create` se prvotně vytvoří firmě složka, ve formátu emailové adresy a názvu firmy, v Google Drive informačního systému. Poté následuje úsek kódu, díky němuž se na emailovou adresu firmy odešle zpráva o sdílení prostoru na Google Drive (při zaškrtnutí volby pro aktivaci Google Drive). Po sdílení se samotný účet vytvoří i v databázi.

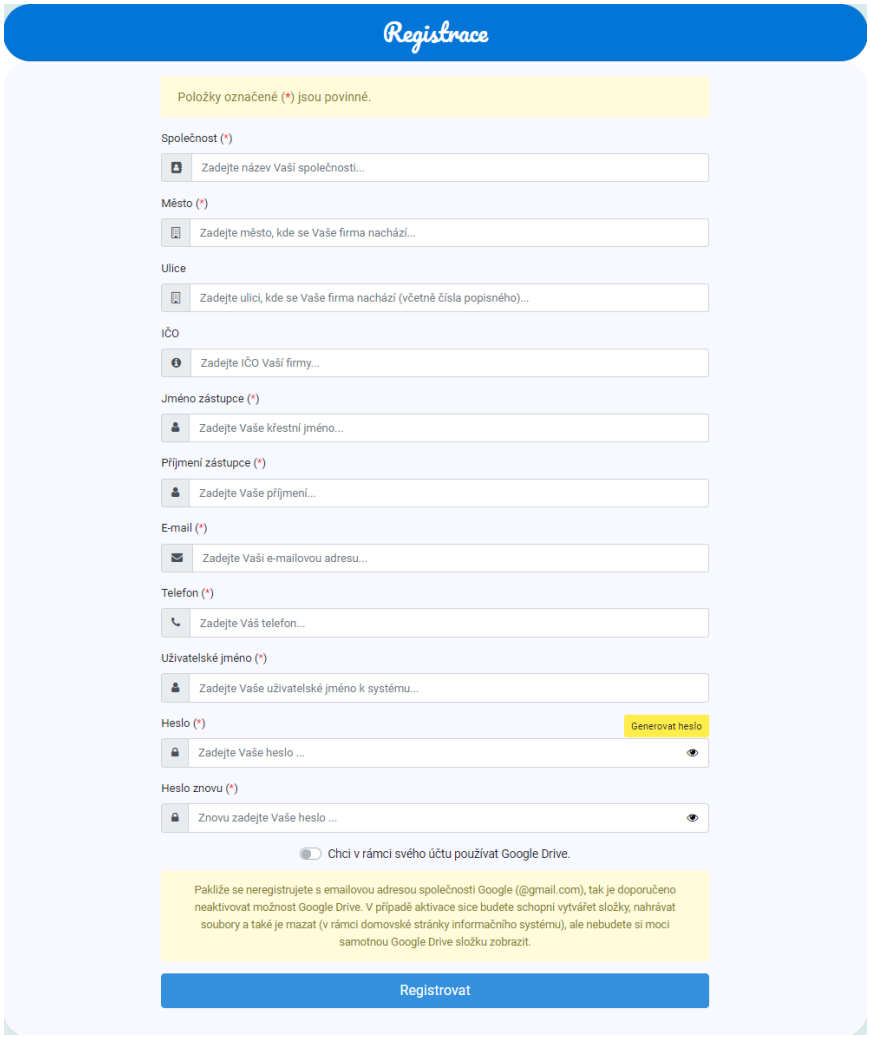
Po vytvoření účtu v databázi je na emailovou adresu firmy zaslána emailová zpráva, která slouží k ověření dané emailové adresy. Následně je firma přesměrována na webovou stránku, která je bránou do samotného informačního systému. Firma tedy nejprve musí svoji emailovou adresu ověřit, jinak ji nebude umožněn přístup do informačního systému. V případě manuální registrace firmy adminem je princip naprosto stejný. Jediným rozdílem je, že v případě manuální registrace se využívá AJAX technologie.

Vytváření zaměstnance

Práci s registračním formulářem obstarává soubor `EmployeeDatatableController.php` a definice vzhledu registračního formuláře je popsána v souboru `employee_list.blade.php` v případě vytváření zaměstnance v seznamu zaměstnanců. V případě, kdy libovolný účet s rolí firmy přidá zaměstnance skrze domovskou stránku či postranní panel, tak práci s registračním formulářem obstarává soubor `UserCompanyController.php` a definice vzhledu registračního formuláře je popsána v souboru `company_dashboard.blade.php`.

Vytváření zaměstnance na domovské stránce oproti jeho vytváření v seznamu zaměstnanců obsahuje možnost přímo nahrát profilový obrázek zaměstnance v rámci registrace.

V registračním formuláři firma zadá jméno a příjmení zaměstnance, jeho emailovou adresu, telefonní číslo, pozici, ulici (volitelné) a město bydliště, uživatelské jméno, poznámku (volitelné) a heslo. Po stisknutí tlačítka *Vytvořit zaměstnance* je zavolána metoda ze souboru `EmployeeDatatableController.php`, kde se prvotně provede validace zadaných údajů ve formuláři. Pokud jsou zadané údaje validní, tak se vytvoří účet zaměstnance v databázi. Následně se zaměstnanci vytvoří složka, ve formátu jména a příjmení zaměstnance, v Google Drive složce firmy a zároveň se mu nasdílí (při zaškrtnutí volby nasdílení).



Obrázek 6.2: Ukázka registračního formuláře, vytvořil: autor.

6.5 Správa uživatelů

Správa firem

V informačním systému mohou spravovat firmy pouze účty s rolí admina. Ke správě firem se lze dostat za pomocí kliknutí na *Seznam firem*, na domovské stránce, v účtu admina nebo pomocí kliknutí na *Seznam firem*, který se nachází na postranním panelu.

O zobrazení datové tabulky v *Seznamu firem* se stará metoda `getCompanies` ze souboru `AdminCompaniesDatatable.php`, který se používá pro všechny operace, v rámci celé webové stránky, reprezentující *Seznam firem*. Definice vzhledu celé webové stránky pro správu firem, včetně modálních oken, je v souboru `companies_list.blade.php`.

V rámci datové tabulky je u každé firmy možnost onu firmu upravovat a odstraňovat. Je zde i možnost manuální registrace firmy, která byla popsána výše. Každé tlačítko *Zobrazit* a *Smazat* je s konkrétní firmou svázáno za pomocí atributu `data-id` v kódu jazyka HTML. Hodnoty atributů `data-id` jsou samotné identifikátory firem.

Pro zobrazení profilu firmy je potřeba zakliknout tlačítko *Zobrazit*. Po stisknutí onoho tlačítka dojde k zavolání metody `edit`, která zobrazí modální okno s vyplněnými údaji dané firmy. V profilu firmy lze upravovat údaje obecného charakteru, měnit heslo a lze si také zobrazit statistiky k dané firmě. Po stisknutí tlačítka *Aktualizovat* dojde k zavolání metody `update`, kde dojde ke zvalidování zadaných údajů. Pakliže jsou údaje validní, tak se pokračuje se synchronizací názvu a emailové adresy firmy s její složkou v Google Drive. Následně dojde k samotné aktualizaci údajů v databázi a také k aktualizaci údajů firmy v analytické sekci OLAP.

Pro odstranění účtu firmy je potřeba zakliknout tlačítko *Smazat*. Po stisknutí onoho tlačítka dojde k zobrazení modálního okna, které slouží k potvrzení smazání firmy. V modálním okně se vyskytuje tlačítko *Ano*. Po jeho stisknutí dojde k zavolání metody `destroy`, která firmě nejprve odstraní její Google Drive složku, poté firmu odstraní z analytické sekce OLAP a na konec dojde k samotnému odstranění firmy z databáze.

Správa zaměstnanců

V informačním systému mohou zaměstnance spravovat pouze účty s rolí firmy. Ke správě zaměstnanců se lze dostat za pomocí kliknutí na *Seznam zaměstnanců*, na domovské stránce, v účtu firmy nebo pomocí kliknutí na *Seznam zaměstnanců*, který se nachází na postranním panelu.

O zobrazení datové tabulky v *Seznamu zaměstnanců* se stará metoda `getEmployees` ze souboru `EmployeeDatatableController.php`, který se používá pro všechny operace, v rámci celé webové stránky, reprezentující *Seznam zaměstnanců*. Definice vzhledu celé webové stránky pro správu zaměstnanců, včetně modálních oken, je v souboru `employee_list.blade.php`. V rámci datové tabulky je u každého zaměstnance možnost mu upravit údaje, změnit heslo, přiřadit jazyky a ohodnotit ho. Mezi další možnosti patří odstranění zaměstnance, přiřazení směn zaměstnanci nebo vyplnění jeho docházky v rámci jeho směn. Je zde i možnost vytváření zaměstnance, která byla popsána výše. Každé tlačítko *Zobrazit*, *Smazat*, *Hodnotit*, *Přiřadit* a *Docházka* je s konkrétním zaměstnancem svázáno za pomocí atributu `data-id` v kódu jazyka HTML. Hodnoty atributů `data-id` jsou samotné identifikátory zaměstnanců.

Pro zobrazení profilu zaměstnance je potřeba zakliknout tlačítko *Zobrazit*. Po stisknutí onoho tlačítka dojde k zavolání metody `edit`, která zobrazí modální okno s vyplněnými údaji daného zaměstnance. V profilu zaměstnance lze upravovat údaje obecného charakteru, měnit heslo a lze si také zobrazit statistiky k danému zaměstnanci. Po stisknutí tlačítka *Aktualizovat* dojde k zavolání metody `update`, kde dojde ke zvalidování zadaných údajů, pakliže jsou údaje validní, tak se pokračuje se synchronizací jména a příjmení zaměstnance s jeho složkou v Google Drive. Následně dojde k samotné aktualizaci údajů v databázi a také k aktualizaci údajů zaměstnance v analytické sekci OLAP.

Pro odstranění účtu zaměstnance je potřeba zakliknout tlačítko *Smazat*. Po stisknutí onoho tlačítka dojde k zobrazení modálního okna, které slouží k potvrzení smazání zaměstnance. V modálním okně se vyskytuje tlačítko *Ano*. Po jeho stisknutí dojde k zavolání metody `destroy`, která zaměstnanci nejprve odstraní jeho Google Drive složku, poté zaměstnance odstraní z analytické sekce OLAP a na konec dojde k samotnému odstranění zaměstnance z databáze.

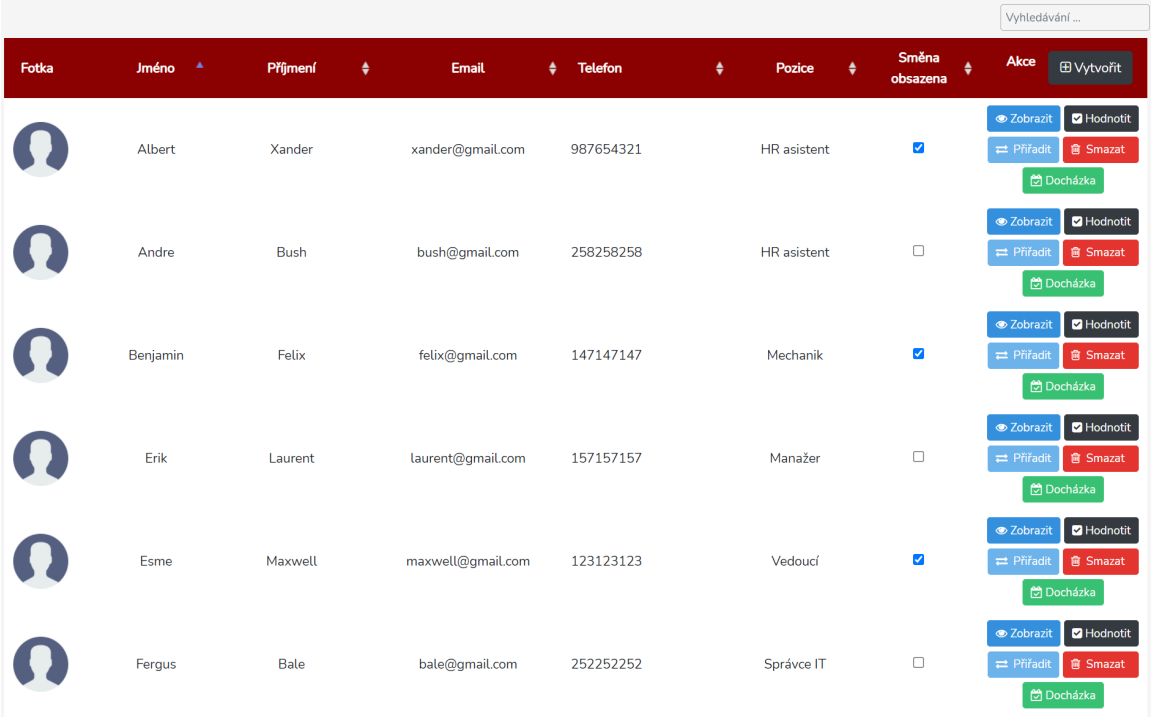
Pro ohodnocení zaměstnance je potřeba zakliknout tlačítko *Hodnotit*. Po jeho stisknutí dojde k zavolání metody `editRate`, která zobrazí dané modální okno společně se třemi posuvníky. Jednotlivé posuvníky mají minimální hodnotu 0 a maximální hodnotu 5. Po-

suvníky reprezentují pracovitost, dochvilnost a spolehlivost. V modálním okně se nachází tlačítko *Hodnotit*. Po jeho stisknutí dojde k zavolání metody `updateRate`, která zaktualizuje zaměstnanci jeho hodnocení v databázi.

Pro přiřazení směny je potřeba zakliknout tlačítko *Přiřadit*. Po jeho stisknutí je zavolána metoda `assignShift`, která zobrazí modální okno společně s dostupnými směnami pro zaměstnance. Po vybrání směn a následném stisknutí tlačítka *Přiřadit* je zavolána metoda `updateassignShift`, která vytvoří záznamy do tabulky zaměstnaneckých směn `table_employee_shifts` a následně vytvoří také záznamy v analytické sekci OLAP. Směny, které zaměstnanec původně měl přiřazené a firma mu je poté změnila na směny jiné jsou smazány z tabulky `table_employee_shifts` i z analytické sekce OLAP.

Pro vyplnění docházky daného zaměstnance je nutné zakliknout tlačítko *Docházka*. Po jeho stisknutí je zavolána metoda `getAttendanceOptions`, která slouží k zobrazení modálního okna, které obsahuje jednotlivé možnosti docházky. Jednotlivá tlačítka v modálním okně jsou detailněji popsána v sekci 6.7.

Odstraňovat zaměstnance z informačního systému lze i skrze domovskou stránku či postranní panel v rámci účtu firmy.



Obrázek 6.3: Ukázka správy zaměstnanců, vytvořil: autor.

6.6 Správa směn

V informačním systému mohou spravovat směny pouze účty s rolí firmy. Ke správě směn se lze dostat za pomocí kliknutí na *Seznam směn*, na domovské stránce, v účtu firmy nebo pomocí kliknutí na *Seznam směn*, který se nachází na postranním panelu.

O zobrazení datové tabulky v *Seznamu směn* se stará metoda `getShifts` ze souboru `ShiftDatatableController.php`, který se používá pro všechny operace, v rámci celé webové stránky, reprezentující *Seznam směn*. Definice vzhledu celé webové stránky pro správu směn, včetně modálních oken, je v souboru `shift_list.blade.php`.

V rámci seznamu směn je možné vytvářet i směny nové. Pro účel vytváření nových směn slouží tlačítko *Vytvořit*. Po jeho stisknutí se zobrazí modální okno, ve kterém se nachází tlačítko *Vytvořit směnu*. Po stisknutí tohoto tlačítka se zavolá metoda `store`, která údaje zadané ve formuláři zvaliduje. Pokud validace dopadne v pořádku je nová směna uložena do databáze.

V rámci datové tabulky je u každé směny možnost onu směnu upravovat, odstraňovat, přiřadit k ní zaměstnance nebo jí vyplnit docházku. Každé tlačítko *Zobrazit*, *Smazat*, *Přiřadit* a *Docházka* je s konkrétní směnou svázáno za pomocí atributu `data-id` v kódu jazyka HTML. Hodnoty atributů `data-id` jsou samotné identifikátory směn.

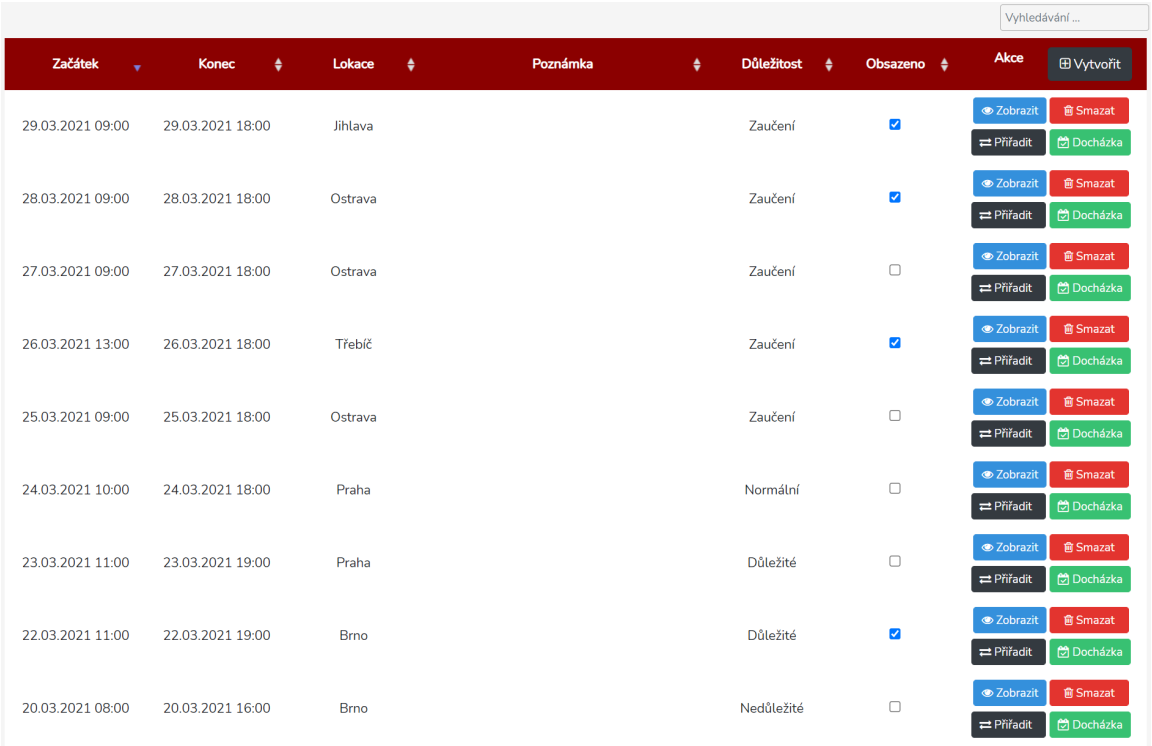
Pro zobrazení detailu směny je potřeba zakliknout tlačítko *Zobrazit*. Po stisknutí onoho tlačítka dojde k zavolání metody `edit`, která zobrazí modální okno s vyplněnými údaji dané směny. V detailu směny lze upravovat údaje obecného charakteru, příkladem může být začátek a konec směny. Po stisknutí tlačítka *Aktualizovat* dojde k zavolání metody `update`, kde dojde ke zvalidování zadaných údajů. V případě validních údajů následně dojde k samotné aktualizaci údajů v databázi a také k aktualizaci údajů směny v analytické sekci OLAP.

Pro odstranění směny je potřeba zakliknout tlačítko *Smazat*. Po stisknutí onoho tlačítka dojde k zobrazení modálního okna, které slouží k potvrzení smazání směny. V modálním okně se vyskytuje tlačítko *Ano*. Po jeho stisknutí dojde k zavolání metody `destroy`, která odstraní danou směnu z databáze a z analytické sekce OLAP.

Pro přiřazení směny je potřeba zakliknout tlačítko *Přiřadit*. Po jeho stisknutí je zavolána metoda `assignEmployee`, která zobrazí modální okno společně s dostupnými zaměstnanci. Po vybrání zaměstnanců a následném stisknutí tlačítka *Přiřadit* je zavolána metoda `updateassignEmployee`, která vytvoří záznamy do tabulky zaměstnaneckých směn `table_employee_shifts` a následně vytvoří také záznamy v analytické sekci OLAP. V případě odebrání zaměstnance ze směny se daná zaměstnanecká směna odstraní z tabulky `table_employee_shifts` a také z analytické sekce OLAP.

Pro vyplnění docházky daného zaměstnance je potřeba zakliknout tlačítko *Docházka*. Po jeho stisknutí je zavolána metoda `getAttendanceOptions`, která slouží k zobrazení modálního okna, které obsahuje jednotlivé možnosti docházky. Jednotlivá tlačítka v modálním okně jsou detailněji popsána v sekci 6.7.

Směny lze také vytvářet a odstraňovat na domovské stránce či na postranním panelu informačního systému.



Obrázek 6.4: Ukázka správy směn, vytvořil: autor.

6.7 Správa docházek

V informačním systému mohou spravovat jednotlivé docházky pouze účty s rolí firmy. Účty s rolí zaměstnance se mohou na směny pouze zapisovat v čas, kdy směna probíhá nebo maximálně 20 minut před začátkem směny. Spravovat docházku je možné z mnoha částí informačního systému, a to ze *Seznamu zaměstnanců*, *Seznamu směn* a z možnosti *Docházka*. K těmto částem se lze dostat skrze domovskou stránku či z postranního panelu informačního systému.

Správa zaměstnanců a směn byla již popsána výše. U možnosti *Docházka* je nejprve nutné vybrat zaměstnance. Po jeho výběru je zobrazena datová tabulka, kterou vytvořila metoda `getAttendance` ze souboru `EmployeeAttendanceController.php`. Jednotlivá tlačítka modálního okna docházky používají metody ze souboru `ShiftDatatableController.php`. Definice vzhledu celé webové stránky pro možnost *Docházka*, včetně modálních oken, je v souboru `attendances.blade.php`.

V rámci datové tabulky je u každé směny zaměstnance možnost mu vyplnit docházku pomocí tlačítka *Docházka*. Každé tlačítko *Docházka* je s konkrétní zaměstnaneckou směnou svázáno za pomocí atributu `data-id` v kódu jazyka HTML. Hodnoty atributů `data-id` jsou samotné identifikátory zaměstnaneckých směn.

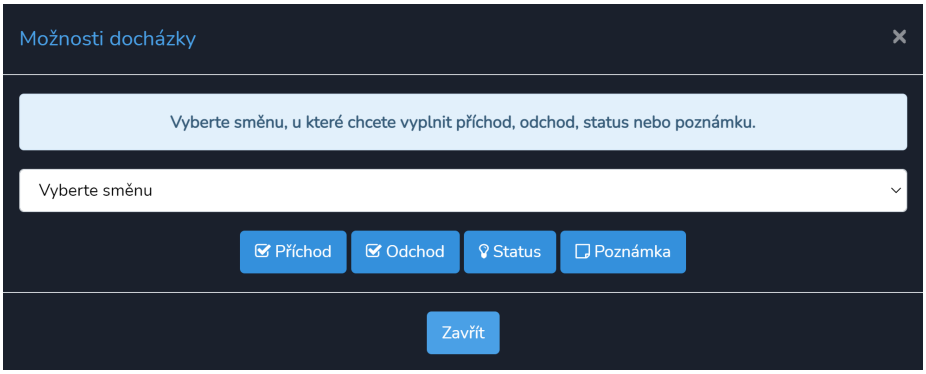
Pro vyplnění docházky daného zaměstnance je potřeba zakliknout tlačítko Docházka. Po jeho stisknutí je zavolána metoda `getAttendanceOptions`, která slouží k zobrazení modálního okna, které obsahuje jednotlivé možnosti docházky. Modální okno docházky obsahuje celkově čtyři tlačítka, a to *Příchod*, *Odchod*, *Status* a *Poznámka*.

První zmíněné tlačítko slouží pro zápis příchodu zaměstnance na směnu. Po jeho stisknutí se zavolá metoda `showCheckin`, díky níž se zobrazí další modální okno, v němž lze vyplnit datum a čas příchodu. Po vyplnění datumu a času příchodu je nutné stisknout tlačítko *Uložit*, které zavolá metodu `updateCheckIn`. Tato metoda nejprve daný datum a čas zvaliduje, pakliže validace proběhla bez chyb, tak se příchod zapíše do databáze a také se zkontroluje, zdali už je zapsaný odchod. Pokud ano, tak se vypočítá odpracovaná doba na směně, která se uloží do OLAP sekce informačního systému. Pokud zaměstnanec přišel pozdě tak se do OLAP sekce informačního systému také uloží počet hodin zpoždění a indikátor zpoždění se nastaví na hodnotu 1.

Druhé zmíněné tlačítko slouží pro zápis odchodu zaměstnance na směnu. Po jeho stisknutí se zavolá metoda `showCheckout`, díky níž se zobrazí další modální okno, v němž lze vyplnit datum a čas odchodu. Po vyplnění datumu a času odchodu je nutné stisknout tlačítko *Uložit*, které zavolá metodu `updateCheckOut`. Tato metoda nejprve daný datum a čas zvaliduje, pakliže validace proběhla bez chyb, tak se příchod zapíše do databáze a také se zkontroluje, zdali už je zapsaný příchod. Pokud ano, tak se vypočítá odpracovaná doba na směně, která se uloží do OLAP sekce informačního systému.

Třetí zmíněné tlačítko slouží pro zápis statusu docházky. Po jeho stisknutí se zavolá metoda `showAbsence`, díky níž se zobrazí další modální okno, v němž lze vybrat status docházky. Po vybrání statusu docházky je nutné stisknout tlačítko *Uložit*, které zavolá metodu `updateAbsence`. Tato metoda uloží vybraný status docházky do databáze, společně s indikátorem, zdali zaměstnanec na směnu přišel. Zároveň se status docházky uloží i do OLAP sekce informačního systému.

Poslední zmíněné tlačítko slouží pro zápis poznámky k docházce. Po jeho stisknutí se zavolá metoda `showAttendanceNote`, díky níž se zobrazí další modální okno, v němž lze danou poznámku zapsat. Po zapsání poznámky k docházce je nutné stisknout tlačítko *Uložit*, které zavolá metodu `updateAttendanceNote`. Tato metoda uloží danou poznámku do databáze.



Obrázek 6.5: Ukázka možností docházky, vytvořil: autor.

6.8 Google Drive

V rámci informačního systému si zaměstnanci a firmy mohou zobrazovat svoje Google Drive složky a nejen to. Do své Google Drive složky si mohou také nahrávat soubory či vytvářet nové složky nebo je naopak odstraňovat. Účty s rolí admina mohou spravovat celou Google Drive strukturu. Všechny možnosti pro práci s Google Drive jsou dostupné na domovské stránce informačního systému.

Propojení účtu firmy s Google Drive je realizováno přes atributy `company_url` u účtů firem a `employee_url` u účtů zaměstnanců, kdy se do těchto atributů uloží identifikátor dané složky účtu. Následně na domovské stránce je u možnosti Zobrazit Google Drive odkaz, který vede přímo na danou složku v rámci Google Drive.

Nahrávání souborů na Google Drive zajišťuje metoda `uploadGoogleDrive` ze souboru `UserCompanyController.php` u účtů firem. Stejnojmenná metoda se používá i u účtů zaměstnanců a pochází z `UserEmployeeController`. Po stisknutí možnosti nahrání souboru se prvně odešle AJAXový, který zjistí názvy složek a jejich identifikátory v rámci dané složky účtu. Po vybrání složky, kam chce uživatel obrázek nahrát, jsou zavolány ony metody `uploadGoogleDrive`.
Metody `uploadGoogleDrive` fungují následovně:

1. Dojde k připojení k danému Google Drive přes servisní účet.

2. Vytvoří se nový soubor, konkrétně následovně:
   `$soubor = new Google_Service_Drive_DriveFile();`.

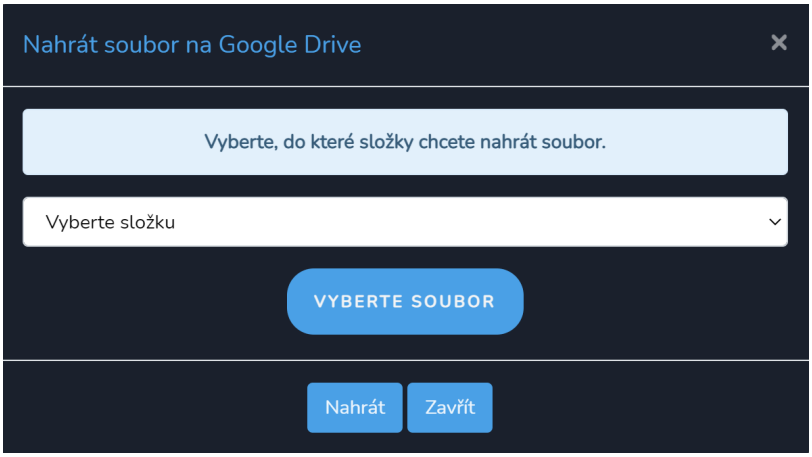
3. Nastaví se mu parametry, příkladem může být nastavení jména souboru nebo nasměrovaní souboru do konkrétní složky:
   `$soubor->setName($request->file('nazevSouboru')->getClientOriginalName());`.
   `$soubor->setParents([$idSlozkyProUlozeniSouboru]);`

4. Soubor je uložen na Google Drive do konkrétní složky vybrané uživatelem, konkrétně následovně:
```
$servis->files->create($soubor,[
    'data' => $obsahSouboru,
    'mimeType' => $mimeType,
    'uploadType' => "resumable"
]);
```

Vytváření složek na Google Drive zajišťují metody `createFolderGoogleDrive` u účtů zaměstnanců i firem. Metody pochází z výše zmíněných souborů. Princip vytváření složky oproti nahrávání souboru je hlavně tzv. typ MIME, který určuje typ dat. V případě vytváření složky je typ MIME vždy `application/vnd.google-apps.folder`.

Odstraňování souborů z Google Drive složky firmy či zaměstnance funguje na stejném principu jako nahrávání souborů nebo vytváření složek. Po stisknutí možnosti *Smazat soubory na Google Drive* se nejprve provede AJAXový požadavek, který zjistí všechny dostupné složky a soubory v rámci složky daného účtu. Samotné odstranění souborů či složek zajišťují metody `deleteFileGoogleDrive`. Metody pochází z výše zmíněných souborů. Samotné mazání vybraných souborů či složek koná následující úsek kódu:

```
foreach($request->seznam_souboru as $soubor){
      $servis->files->delete($soubor);
}
```



Obrázek 6.6: Ukázka modálního okna pro nahrávání souboru na Google Drive, vytvořil: autor.

6.9 Generování souborů

Generování souborů je realizováno knihovnou DOMPDF Wrapper for Laravel. O generování souborů se starají následující soubory:

- `AdminFileGeneratorController.php` pro účty s rolí admina,
- `FileGeneratorController.php` pro účty s rolí firmy,
- `EmployeeFileGenerator.php` pro účty s rolí zaměstnance.

Princip generování souborů je následující:

1. Do proměnné, například proměnné `$out`, se uloží struktura budoucího souboru včetně dat, které v souboru mají být.
2. Proměnná `$out` je použita ke generování souboru, konkrétně následovně:
   ```
   return PDF::loadHTML($out)
       ->setPaper('a4','portrait')
       ->download('soubor.pdf','UTF-8');
   ```
3. Výše uvedeným řádkem kódu se vygeneruje soubor `soubor.pdf`, v kterém bude obsažena struktura a data z proměnné `$out`.

Obrázek 6.7: Ukázka možností generování souborů, vytvořil: autor.

6.10 Statistiky

Statistiky jsou realizovány knihovnou Chart.js. K této knihovně byly použity dva pluginy. Prvním je Chart.js Doughnutlabel plugin a druhým je Chart.js Datalabels plugin. O statistiky se starají následující soubory:

- `AdminStatisticsController.php` pro účty s rolí admina,
- `StatisticsController.php` pro účty s rolí firmy,
- `EmployeeStatisticsController.php` pro účty s rolí zaměstnance.

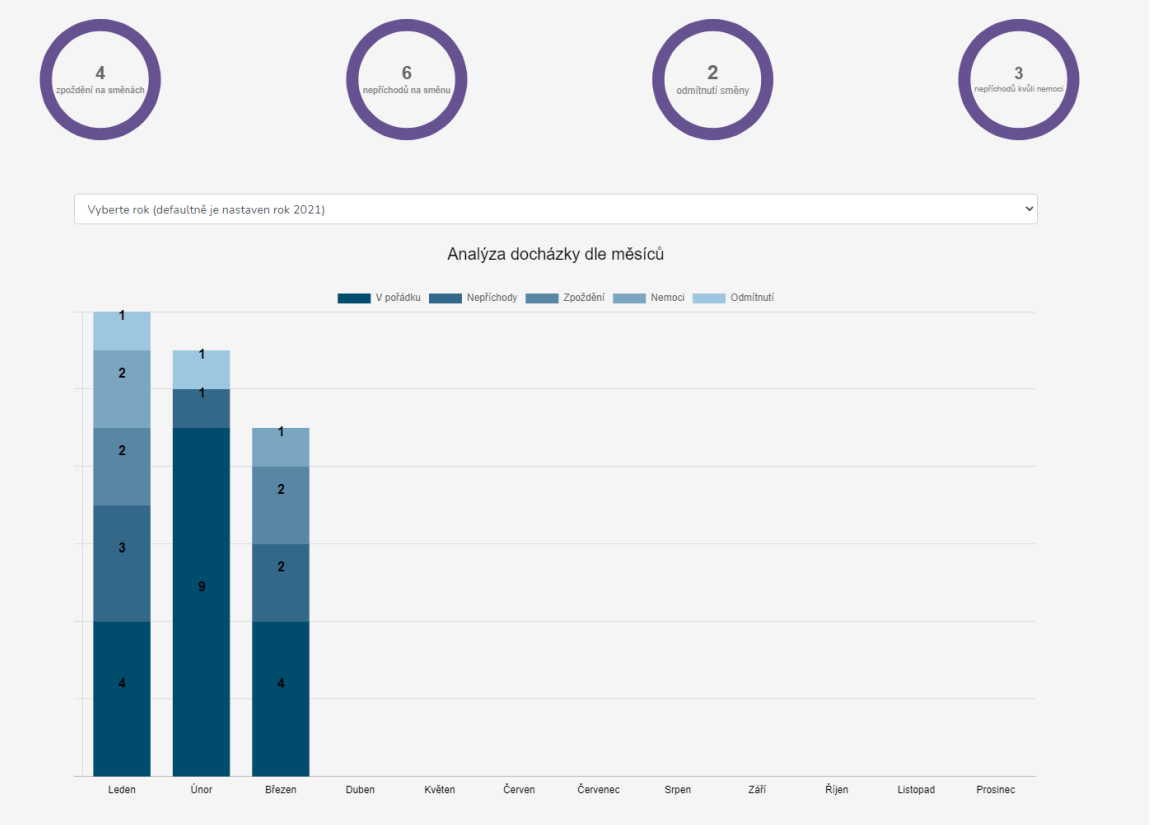
Princip fungování statistik je následující:

1. V jednotlivých výše uvedených souborech se zavolají funkce z modelů informačního systému, výsledky těchto funkcí se uloží do proměnných.
2. Jednotlivé proměnné se pošlou do šablony `statistics.blade.php`.
3. V šabloně jsou jednotlivé proměnné využity k tvorbě grafů za pomocí jazyka Javascript a knihovny Chart.js.

V šabloně `statistics.blade.php` jsou předchystané HTML5 canvasy, kterým jsou přiřazeny identifikátory. Díky těmto identifikátorům lze jednoznačně určit konkrétní HTML5 canvas, do kterého se následně vykreslí graf. Ukázka jednoduchého sloupcového grafu může být následující:

```
var canvasSmeny = $("#barChartSmeny);
barChartSmeny = new Chart(canvasSmeny,{
    type:"bar",
    data:{
      labels:["Leden", "Únor", "Březen", "Duben"],
      datasets:[{
         label: "Testovací popisek",
         data: [1, 2, 3, 4],
         backgroundColor: ["#2E8B57", "#2E8B57", "#2E8B57", "#2E8B57"]
      }]
    },
})
```

V informačním systému, v položce data ve výše zmíněném kódu, jsou použity konkrétní proměnné.



Obrázek 6.8: Ukázka statistik, vytvořil: autor.

6.11 OLAP

O analytickou sekci systému se starají dva soubory. Prvním je soubor `OlapETL.php` a druhým `OlapAnalyzator.php`. První zmíněný soubor slouží pro extrakci dat do analytické sekce systému, druhý soubor slouží k získávání znalostí z dat.

Soubor `OlapETL.php` tedy provádí procesy ETL (Extract, Transform, Load), které jsou detailněji popsány v sekci 2.6. V tomto souboru se nachází 28 funkcí.

Soubor `OlapAnalyzator.php` provádí dotazy na tabulku faktů pomocí funkcí. V tomto souboru se nachází 40 funkcí.

Kapitola 7

Testování

Testování je nedílnou součástí vývoje jakéhokoliv komplexnějšího softwaru. Díky testování zjišťujeme, zdali software, který jsme vytvořili, funguje dle našeho očekávání a zdali neobsahuje nějaké chyby. Testováním tedy zjišťujeme kvalitu daného softwaru.

7.1 Průběžné testování

Po naprogramování větší části softwaru by měl programátor, jenž daný software vyvíjí, pokaždé onu část softwaru řádně otestovat. V rámci této práce byl informační systém testován po každé přidané funkcionalitě hned několikrát. Během průběžného testování bylo zaznamenáno a opraveno několik chyb, které se zejména týkali chybné práce s databází. Mezi časté chyby ale také patřily chyby související se sémantikou samotných kódů a chyby týkající se grafického uživatelského rozhraní.

Testování probíhalo na lokálním serveru, který byl spuštěn nástrojem `artisan` pomocí příkazu `php artisan serve`. Lokální MySQL databáze byla spuštěna nástrojem, který se nazývá `WampServer`[16]. Po naimplementování podstatné části informačního systému byl samotný informační systém přesunut na webový server, který poskytuje společnost Hostinger[17].

7.2 Uživatelské testování

Uživatelské testování započalo po dokončení samotného informačního systému. Účelem testování bylo zjistit, zdali je vytvořený informační systém v praxi kvalitní a také do jaké míry splňuje požadavky na něj kladené. Kvalitní informační systém by měl disponovat hlavně intuitivním grafickým uživatelským rozhraním. Díky uživatelskému testování lze zvýšit pravděpodobnost výskytu neočekávaných chyb.

[16] Dostupné z: https://www.wampserver.com/en/.
[17] Dostupné z: https://www.hostinger.cz/.

Firmy

Pro účely testování uživatelů s rolí firmy byl vytvořen testovací formulář. Formulář obsahuje následující úkony:

1. Zaregistrujte se do informačního systému. V případě registrace s emailovou adresou společnosti Google (gmail) zaškrtněte možnost pro aktivaci Google Drive.
2. Ověřte svoji emailovou adresu.
3. Přihlaste se do informačního systému.
4. Vytvořte jazyk(y).
5. Vytvořte zaměstnance. V případě registrace zaměstnance s emailovou adresou společnosti Google zaškrtněte možnost pro nasdílení vytvořené Google Drive složky s emailovou adresou zaměstnance. Následně zaměstnanci přiřaďte libovolný jazyk.
6. Vytvořte směnu.
7. Přiřaďte zaměstnanci směnu.
8. Vyplňte docházku zaměstnance na dané směně.
9. Ohodnoťte zaměstnance.
10. Změňte zaměstnanci heslo a profilový obrázek.
11. Vytvořte zaměstnanci zranění na směně.
12. Vytvořte zaměstnanci dovolenou, nemocenskou a nahlášení.
13. Schvalte dovolenou, neschvalte nemocenskou a nastavte stav přečteno k danému nahlášení.
14. Vygenerujte si seznam zaměstnanců a směn.
15. Vygenerujte si souhrn hodnocení a aktuální směny zaměstnance.
16. Podívejte se do statistik.
17. Změňte si některé údaje a heslo.
18. Změňte si profilový obrázek a odhlaste se.
19. Změňte si heslo za pomocí možnosti „Zapomněl jste heslo“ v rámci přihlašovacího formuláře.
20. Pakliže jste si aktivoval Google Drive, tak vytvořte novou složku na Google Drive.
21. Pakliže jste si aktivoval Google Drive, tak nahrajte soubor do nově vytvořené složky.
22. Pakliže jste si aktivoval Google Drive, tak si zobrazte firemní Google Drive složku.
23. Pakliže jste si aktivoval Google Drive, tak odstraňte vytvořenou složku.
24. Odstraňte si profilový obrázek.

25. Odstraňte zaměstnance.
26. Odstraňte směnu.
27. Odhlaste se a znovu se přihlaste.
28. Smažte si účet.

Zaměstnanci

Pro účely testování uživatelů s rolí zaměstnance byl vytvořen testovací formulář. Formulář obsahuje následující úkony:

1. Přihlaste se do systému.
2. Změňte si některé údaje a heslo.
3. Změňte si profilový obrázek.
4. Zobrazte si své aktuální směny.
5. Zobrazte si své všechny směny.
6. Vytvořte dovolenou a následně o ni zažádejte.
7. Vytvořte nemocenskou a následně o ni zažádejte.
8. Vytvořte nahlášení a následně ono nahlášení odešlete.
9. Zrušte žádost o dovolenou a následně ji odstraňte.
10. Zobrazte si historii zranění.
11. Zobrazte si statistiky.
12. Vygenerujte si své údaje.
13. Vygenerujte si své aktuální směny a seznam nemocenských.
14. Pakliže Vám firma nasdílela Vaši složku, tak vytvořte složku na Google Drive.
15. Pakliže Vám firma nasdílela Vaši složku, tak nahrajte soubor do nově vytvořené složky.
16. Pakliže Vám firma nasdílela Vaši složku, tak si zobrazte Google Drive.
17. Pakliže Vám firma nasdílela Vaši složku, tak odstraňte vytvořenou složku.
18. Odhlaste se a znovu se přihlaste.
19. Odstraňte si profilový obrázek.
20. Smažte si účet.

Uživatelského testování se účastnila jedna firma[18], další uživatelé byli rodinní příslušníci a přátelé. V rámci uživatelského testování se objevilo pár nedostatků, které byly následně opraveny. V rámci uživatelského testování také došlo k různým návrhům na možná rozšíření informačního systému. Navržená rozšíření jsou popsána v sekci 8.1.

[18] https://www.autodobrovolny.cz/.

Kapitola 8

Závěr

Cílem této bakalářské práce bylo vytvořit informační systém pro správu zaměstnanců ve firmách. Účel tohoto informačního systému tkví zejména v tom, že usnadňuje a zefektivňuje administraci zaměstnanců včetně plánování jejich směn a vyplňování jejich docházek a také poskytuje řadu jiných funkcionalit, příkladem může být generování různých souborů ve formátu PDF. Informační systém vyniká propojením s Google Drive za pomocí Google Drive API. Všechny požadavky kladené na informační systém jsou splněny a informační systém je tak plně funkční.

V rámci této práce byly nejprve představeny a popsány principy tvorby webových aplikací. Poté následovaly sekce o technologiích používaných pro realizaci libovolných webových aplikací. Následovala kapitola, která se zabývala konkrétními použitými technologiemi při vývoji informačního systému.

Po představení použitých technologií následovala kapitola, která se týkala samotných požadavků na informační systém. Na základě oněch požadavků byl vytvořen diagram případů užití pro jednotlivé role v informačním systému.

Následující kapitola se zabývala návrhem informačního systému. V této kapitole byl také představen ER diagram. Jednotlivé entitní množiny, které se nachází v ER diagramu, byly v této kapitole popsány. Na závěr této kapitoly je také představen návrh grafického uživatelského prostředí.

Po návrhu informačního systému následovala kapitola pojednávající o implementaci informačního systému. Na úvod této kapitoly je popsána adresářová struktura webové aplikace. Následně jsou popsány důležité části informačního systému z hlediska implementace.

Po návrhu informačního systému následuje předposlední kapitola pojednávající o testování informačního systému. Informační systém byl průběžně testován programátorem, také byl podroben uživatelskému testování. Chyby nalezené při testování byly opraveny.

Webová aplikace je dostupná na adrese www.tozondo.com.

8.1 Možná rozšíření

V rámci budoucího vývoje informačního systému jsou možná tato rozšíření:

- Integrování výplatního systému.
- Možnost zasílání zpráv mezi zaměstnanci a mezi zaměstnanci a firmou.
- Spojení informačního systému s Google Calendar API.
- Možnost vytváření individuálních rozvrhů směn u zaměstnaneckých účtů. Následně by bylo umožněno zaslat firmě daného zaměstnance daný rozvrh. Na základě tohoto rozvrhu by se firma mohla lépe rozhodovat jakému zaměstnanci přidělit jaké směny.
- Vytvoření jednoduché mobilní aplikace pro zápis příchodu a odchodu zaměstnance.
- Vytvoření tzv. achievementů. Achievementy, v českém znění úspěchy, by byly uděleny zaměstnancům na základě jejich výsledků v jejich firmě. Příkladem by mohl být úspěch, který by vyjadřoval, že zaměstnanec prošel první směnou.

Literatura

[1] *Amazon AWS: What is PostgreSQL?* [online]. [cit. 2021-04-14]. Dostupné z: https://aws.amazon.com/rds/postgresql/what-is-postgresql/.

[2] *Best-hosting: CO JE TO DATABÁZE MYSQL?* [online]. [cit. 2021-04-14]. Dostupné z: https://best-hosting.cz/cs/napoveda/co-je-to-databaze-mysql.

[3] *Chart.js: Simple yet flexible JavaScript charting for designers & developers* [online]. [cit. 2021-04-14]. Dostupné z: https://www.chartjs.org/.

[4] *Composer: Command-line interface* [online]. [cit. 2021-04-14]. Dostupné z: https://getcomposer.org/doc/03-cli.md.

[5] *Composer: Getting Started* [online]. [cit. 2021-04-14]. Dostupné z: https://getcomposer.org/doc/00-intro.md.

[6] *CoreIT: Co je to PHP a k čemu mi bude dobré?* [online]. [cit. 2021-04-14]. Dostupné z: https://www.coreit.cz/php/.

[7] *Free Java Guide: History of Java programming language* [online]. [cit. 2021-04-14]. Dostupné z: https://www.freejavaguide.com/history.html.

[8] *Google API Console: APIs and Services* [online]. [cit. 2021-04-14]. Dostupné z: https://console.developers.google.com/.

[9] *Google Drive API v3: Introduction to Google Drive API* [online]. [cit. 2021-04-14]. Dostupné z: https://developers.google.com/drive/api/v3/about-sdk.

[10] *Google Drive API v3: Roles* [online]. [cit. 2021-04-14]. Dostupné z: https://developers.google.com/drive/api/v3/ref-roles.

[11] *Google Drive Help: Buy more Google storage* [online]. [cit. 2021-04-14]. Dostupné z: https://support.google.com/drive/answer/2375123?co=GENIE.Platform%3DDesktop&hl=en.

[12] *Guru99: What is Backend Developer? Skills to become a Web Developer* [online]. [cit. 2021-04-14]. Dostupné z: https://www.guru99.com/what-is-backend-developer.html.

[13] *JavaTpoint: History of Java* [online]. [cit. 2021-04-14]. Dostupné z: https://www.javatpoint.com/history-of-java.

[14] *Laravel docs: Artisan Console* [online]. [cit. 2021-04-14]. Dostupné z: https://laravel.com/docs/8.x/artisan.

[15] *Laravel docs: Authentication* [online]. [cit. 2021-04-14]. Dostupné z: https://laravel.com/docs/7.x/authentication.

[16] *Laravel Docs: Blade Templates* [online]. [cit. 2021-04-14]. Dostupné z: https://laravel.com/docs/8.x/blade.

[17] *Laravel docs: Directory Structure* [online]. [cit. 2021-04-14]. Dostupné z: https://laravel.com/docs/8.x/structure.

[18] *Lucidchart: What is an Entity Relationship Diagram (ERD)?* [online]. [cit. 2021-04-14]. Dostupné z: https://www.lucidchart.com/pages/er-diagrams.

[19] *ManagementMania: DBMS (Database Management System) - systém řízení báze dat* [online]. [cit. 2021-04-14]. Dostupné z: https://managementmania.com/cs/dbms-database-management-system-system-rizeni-baze-dat.

[20] *MongoDB: How much easier are documents to work with than tables?* [online]. [cit. 2021-04-14]. Dostupné z: https://www.mongodb.com/document-databases.

[21] *PHP Manual: Supported Versions* [online]. [cit. 2021-04-14]. Dostupné z: https://www.php.net/releases/index.php.

[22] *PHP Manual: What can PHP do?* [online]. [cit. 2021-04-14]. Dostupné z: https://www.php.net/manual/en/intro-whatcando.php.

[23] *PostgreSQL: About PostgreSQL* [online]. [cit. 2021-04-14]. Dostupné z: https://www.postgresql.org/about/.

[24] *Python: Python Releases* [online]. [cit. 2021-04-14]. Dostupné z: https://www.python.org/downloads/.

[25] *Python: What is Python? Executive Summary* [online]. [cit. 2021-04-14]. Dostupné z: https://www.python.org/doc/essays/blurb/.

[26] *Sites google: Základní pojmy databáze* [online]. [cit. 2021-04-14]. Dostupné z: https://sites.google.com/site/xvinformatika/databaze/zakladni-pojmy-databaze.

[27] *Smartdraw: Entity Relationship Diagram* [online]. [cit. 2021-04-14]. Dostupné z: https://www.smartdraw.com/entity-relationship-diagram/.

[28] *Stitchdata: ETL Process* [online]. [cit. 2021-04-14]. Dostupné z: https://www.stitchdata.com/etldatabase/etl-process/.

[29] *That Company: What is CSS?* [online]. [cit. 2021-04-14]. Dostupné z: https://www.thatcompany.com/what-is-css.

[30] *W3schools: HTML - The Head Element* [online]. [cit. 2021-04-14]. Dostupné z: https://www.w3schools.com/html/html_head.asp.

[31] *W3Schools: jQuery Introduction* [online]. [cit. 2021-04-14]. Dostupné z: https://www.w3schools.com/jquery/jquery_intro.asp.

[32] *W3Schools: PHP - AJAX Introduction* [online]. [cit. 2021-04-14]. Dostupné z: https://www.w3schools.com/php/php_ajax_intro.asp.

[33] *W3schools: Python Introduction* [online]. [cit. 2020-04-14]. Dostupné z: https://www.w3schools.com/python/python_intro.asp.

[34] *ManagementMania: Třívrstvá architektura (Three-tier architecture)* [online]. 2015 [cit. 2021-04-14]. Dostupné z: https://managementmania.com/cs/trivrstva-architektura-three-tier-architecture.

[35] *ManagementMania: Architektura klient-server (Client–server model)* [online]. 2016 [cit. 2021-04-14]. Dostupné z: https://managementmania.com/cs/architektura-klient-server.

[36] *Hack Reactor: What is JavaScript Used For?* [online]. 2018 [cit. 2021-04-14]. Dostupné z: https://www.hackreactor.com/blog/what-is-javascript-used-for.

[37] *FreeCodeCamp: The Advantages and Disadvantages of JavaScript* [online]. 2019 [cit. 2021-04-14]. Dostupné z: https://www.freecodecamp.org/news/the-advantages-and-disadvantages-of-javascript/.

[38] *ScaleGrid: 2019 Database Trends – SQL vs. NoSQL, Top Databases, Single vs. Multiple Database Use* [online]. 2019 [cit. 2021-04-14]. Dostupné z: https://scalegrid.io/blog/2019-database-trends-sql-vs-nosql-top-databases-single-vs-multiple-database-use/.

[39] *Xicom: 9 BEST REASONS TO CHOOSE JAVA FOR WEB DEVELOPMENT* [online]. 2019 [cit. 2021-04-14]. Dostupné z: https://www.xicom.biz/blog/9-best-reasons-to-choose-java-for-web-development/.

[40] *IBM Cloud Education: OLTP* [online]. 2020 [cit. 2021-04-14]. Dostupné z: https://www.ibm.com/cloud/learn/oltp.

[41] *IBM: OLAP analytics* [online]. 2020 [cit. 2021-04-14]. Dostupné z: https://www.ibm.com/cloud/learn/olap.

[42] *MongoDB: Support Policy* [online]. 2021 [cit. 2021-04-14]. Dostupné z: https://www.mongodb.com/support-policy.

[43] ARISTOTE. *ACID* [online]. 2014 [cit. 2021-04-14]. Dostupné z: https://blog.root.cz/aristote/acid/.

[44] ARSENAULT, C. *The Pros and Cons of 8 Popular Databases* [online]. 2017 [cit. 2021-04-14]. Dostupné z: https://www.keycdn.com/blog/popular-databases.

[45] BURGET, R. *Pojem informačního systému, data, informace.* 2020 [cit. 2021-04-14]. Dostupné z: https://wis.fit.vutbr.cz/FIT/st/cfs.php?file=%2Fcourse%2FIIS-IT%2Flectures%2Fp01_Informacni_systemy.pdf&cid=13985.

[46] CARY, I. *Python: Pros and Cons* [online]. 2020 [cit. 2021-04-14]. Dostupné z: https://scholarlyoa.com/python-pros-and-cons/.

[47] DANEL, R. *OLAP* [online]. [cit. 2021-04-14]. Dostupné z: http://homel.vsb.cz/~dan11/dzdb/Danel%20-%20IS%20-%20OLAP.pdf.

[48] Domantas, G. *What is HTML? The Basics of Hypertext Markup Language Explained* [online]. 2019 [cit. 2021-04-14]. Dostupné z: https://www.hostinger.com/tutorials/what-is-html.

[49] Dwika, H. *What is AJAX and How Does It Work?* [online]. 2021 [cit. 2021-04-14]. Dostupné z: https://www.hostinger.com/tutorials/what-is-ajax.

[50] Foral, J. *Webová aplikace* [online]. [cit. 2021-04-14]. Dostupné z: https://wiki.knihovna.cz/index.php/Webov%c3%a1_aplikace#V.C3.BDhody_webov.C3.BDch_aplikac.C3.AD.

[51] Goel, A. *Difference Between CSS, CSS2 And CSS3* [online]. 2021 [cit. 2021-04-14]. Dostupné z: https://hackr.io/blog/difference-between-css-css2-and-css3.

[52] Goyal, S. *Top 11 Python Frameworks for Web Development In 2020* [online]. 2020 [cit. 2021-04-14]. Dostupné z: https://www.netsolutions.com/insights/top-10-python-frameworks-for-web-development-in-2019/.

[53] Herout, T. *Co to jsou statické webové stránky* [online]. 2012 [cit. 2021-04-14]. Dostupné z: https://www.helpmark.cz/slovnikpojmu/46-staticke-webove-stranky.

[54] Herout, T. *Co to jsou statické webové stránky* [online]. 2012 [cit. 2021-04-14]. Dostupné z: https://www.helpmark.cz/slovnikpojmu/32-dynamicke-webove-stranky.

[55] Heuvel, B. *DOMPDF Wrapper for Laravel* [online]. 2020 [cit. 2021-04-14]. Dostupné z: https://github.com/barryvdh/laravel-dompdf.

[56] Hiwarale, U. *How does JavaScript and JavaScript engine work in the browser and node?* [online]. 2018 [cit. 2021-04-14]. Dostupné z: https://medium.com/jspoint/how-javascript-works-in-browser-and-node-ab7d0d09ac2f.

[57] Hoffmann, J. *A Look Back at the History of CSS* [online]. 2017 [cit. 2021-04-14]. Dostupné z: https://css-tricks.com/look-back-history-css/.

[58] Hordějčuk, V. *Architektura Klient-Server* [online]. [cit. 2021-04-14]. Dostupné z: http://voho.eu/wiki/klient-server/.

[59] Jamsheer, K. *ALL ABOUT MONGODB: THE NOSQL DATABASE* [online]. 2019 [cit. 2021-04-14]. Dostupné z: https://acodez.in/mongodb-nosql-database/#Disadvantages_of_MongoDB.

[60] Kutáč, P. *Google Calendar API - události z vlastního kalendáře* [online]. [cit. 2021-04-14]. Dostupné z: https://www.kutac.cz/weby-a-vse-okolo/google-calendar-api-udalosti-z-vlastniho-kalendare.

[61] Lutz, G. *Working with OLAP Cubes*. 2018 [cit. 2021-04-14]. Dostupné z: https://www.grapecity.com/blogs/working-with-olap-cubes.

[62] MDN contributors. *What is JavaScript?* [online]. 2021 [cit. 2021-04-14]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript.

[63] Mutimutema, T. *10 of the Most Popular Java Frameworks of 2020* [online]. 2021 [cit. 2020-04-14]. Dostupné z: https://stackify.com/10-of-the-most-popular-java-frameworks-of-2020/.

[64] Njenga, A. *10 Popular PHP frameworks in 2020* [online]. 2020 [cit. 2021-04-14]. Dostupné z: https://raygun.com/blog/top-php-frameworks/.

[65] O'Brien, J. *A Brief History of Laravel* [online]. 2016 [cit. 2021-04-14]. Dostupné z: https://medium.com/vehikl-news/a-brief-history-of-laravel-5d55970885bc.

[66] Ouellette, A. *What is Bootstrap: A Beginners Guide* [online]. [cit. 2021-04-14]. Dostupné z: https://careerfoundry.com/en/blog/web-development/what-is-bootstrap-a-beginners-guide/.

[67] Rieuf, E. *History of MySQL* [online]. 2016 [cit. 2021-04-14]. Dostupné z: https://www.datasciencecentral.com/profiles/blogs/history-of-mysql.

[68] Rouse, M., Ferguson, K., Leake, A. a Hughes, A. *Database (DB)* [online]. [cit. 2021-04-14]. Dostupné z: https://searchsqlserver.techtarget.com/definition/database.

[69] Saha, D. *What is MongoDB and Why to use it?* [online]. 2018 [cit. 2021-04-14]. Dostupné z: https://www.dotnettricks.com/learn/mongodb/what-is-mongodb-and-why-to-use-it.

[70] Shweta, D. *What is PHP & Reasons You Should Learn PHP* [online]. 2019 [cit. 2021-04-14]. Dostupné z: https://medium.com/javarevisited/what-is-php-reasons-you-should-learn-php-38f898a8e287.

[71] Smallcombe, M. *PostgreSQL vs MySQL: The Critical Differences* [online]. 2020 [cit. 2021-04-14]. Dostupné z: https://www.xplenty.com/blog/postgresql-vs-mysql-which-one-is-better-for-your-use-case/#whichprogramminglanguagesdotheysupport.

[72] Sojka, P., Růžička, M., Lupták, D. a Novotný, V. *JavaScript* [online]. [cit. 2021-04-14]. Dostupné z: https://www.fi.muni.cz/lemma/PB029/practices/css-a-javascript/.

[73] Szecsei, S. *ADVANTAGES AND DISADVANTAGES OF THE LARAVEL FRAMEWORK* [online]. 2020 [cit. 2021-04-14]. Dostupné z: https://www.popwebdesign.net/popart_blog/en/2020/03/advantages-and-disadvantages-of-the-laravel-framework/.

[74] TechTarget, C. *Web application (Web app)* [online]. [cit. 2021-04-14]. Dostupné z: https://searchsoftwarequality.techtarget.com/definition/Web-application-Web-app.

[75] Wales, M. *3 Web Dev Careers Decoded: Front-End vs Back-End vs Full Stack* [online]. 2014 [cit. 2021-04-14]. Dostupné z: https://blog.udacity.com/2014/12/front-end-vs-back-end-vs-full-stack-web-developers.html.

[76] Watts, S. *ACID Explained: Atomic, Consistent, Isolated & Durable* [online]. 2020 [cit. 2021-04-14]. Dostupné z: https://www.bmc.com/blogs/acid-atomic-consistent-isolated-durable/.

[77] Zendulka, J. a Bartík, V. *Konceptuální modelování* [online]. 2020 [cit. 2021-04-14]. Dostupné z: https://wis.fit.vutbr.cz/FIT/st/cfs.php?file=%2Fcourse%2FIDS-IT%2Flectures%2Fcz%2F2_kmod.pdf&cid=13296.

[78] Zendulka, J. a Bartík, V. *Relační model dat* [online]. 2020 [cit. 2021-04-14]. Dostupné z: https://wis.fit.vutbr.cz/FIT/st/cfs.php?file=%2Fcourse%2FIDS-IT%2Flectures%2Fcz%2F3_relmod.pdf&cid=13296.

[79] Zendulka, J., Bartík, V., Lukáš, R. a Rudolfová, I. *Získávání znalostí z databází, Studijní opora.* 2009 [cit. 2021-04-14].

[80] Zezula, P. *P002 Úvod do databázových systémů* [online]. [cit. 2021-04-14]. Dostupné z: https://www.fi.muni.cz/~xdohnal/lectures/PB154/czech/zezula13.pdf.

[81] Čermák, M. *Vícevrstvá architektura: tenký, tlustý a chytrý klient* [online]. 2010 [cit. 2021-04-14]. Dostupné z: https://www.cleverandsmart.cz/vicevrstva-architektura-tenky-tlusty-a-chytry-klient/.

[82] Čápka, D. *Lekce 1 - Úvod do HTML a váš první web* [online]. [cit. 2021-04-14]. Dostupné z: https://www.itnetwork.cz/html-css/webove-stranky/jak-psat-moderni-web-html-tutorial-uvod-do-html.

[83] Čápka, D. *Lekce 2 - UML - Use Case Diagram* [online]. [cit. 2021-04-14]. Dostupné z: https://www.itnetwork.cz/navrh/uml/uml-use-case-diagram.

[84] Čápka, D. *MVC architektura* [online]. [cit. 2021-04-14]. Dostupné z: https://www.itnetwork.cz/navrh/mvc-architektura-navrhovy-vzor.

[85] Šmíd, V. *Pojem informačního systému* [online]. [cit. 2021-04-14]. Dostupné z: https://www.fi.muni.cz/~smid/mis-infsys.htm.

[86] Štráfelda, J. *HTML element* [online]. [cit. 2021-04-14]. Dostupné z: https://www.strafelda.cz/html-element.

Příloha A

Obsah SD karty

- **/xsklen12/tozondo/** - zdrojové soubory informačního systému.
- **/xsklen12/sablona/** - zdrojové soubory pro tvorbu technické zprávy ve formátu PDF.
- **/xsklen12/zprava/technicka_zprava.pdf** - samotná technická zpráva.
- **/xsklen12/instalace/** - pokyny pro instalaci.