Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informatiky a kvantitativních metod

Implementace webové aplikace pro plánování směn

Bakalářská práce

Autor: Jiří Pipek
Studijní obor: Aplikovaná informatika

Vedoucí práce: Ing. Michal Macinka

Hradec Králové duben 2021

Prohlášení:

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně a s použitím uvedené literatury.

vlastnoruční podpis

V Hradci Králové dne 24.4.2021 Jiří Pipek

Poděkování:

Děkuji vedoucímu bakalářské práce Ing. Michalu Macinkovi za metodické vedení práce.

Anotace

Cílem bakalářské práce je definovat webovou aplikaci, popsat technologie, které se pro její vývoj využívají, představit jazyk JavaScript s jeho frameworky a na základě těchto teoretických znalostí navrhnout a implementovat funkční webovou aplikaci pro plánování směn. Konkrétně práce představuje technologie HTML, CSS a DOM starající se o strukturu webové aplikace a technologie REST, SOAP a AJAX, které pracují s daty. Popisuje také rozdíl mezi jednostránkovými a vícestránkovými aplikacemi a novodobý trend progresivních webových aplikací. Následně je popsán programovací jazyk JavaScript, jeho frontendové technologie Vue.js, React, Angular a backendové technologie Express, Koa.js a Hapi. V závěru teoretické části se práce zaobírá i různými typy databází, a to MongoDB, PostgreSQL a Firebase. V praktické části práce je navržena aplikace dle daných požadavků a jsou vybrány technologie pro její vývoj. Poslední část již řeší problémy vzniklé při samotné implementaci.

Annotation

Title: Implementation of Shift Planning Web Application

The aim of the bachelor's thesis is to define the web application, describe the technologies used for its development, introduce JavaScript language with its frameworks and, based on theoretical knowledge, design and implement a functional web application for shift planning. In particular, thesis introduces technologies managing the structure of web applications, such as HTML, CSS, and DOM, as well as technologies working with data, namely REST, SOAP, and AJAX. It also describes the difference between single-page and multi-page applications and the modern trend of progressive web applications. Following chapters are describing JavaScript programming language, its frontend technologies like Vue.js, React, and Angular, followed by backend technologies including Express, Koa.js and Hapi. Theoretical part of the dissertation is concluded by various types of databases, namely MongoDB, PostgreSQL and Firebase. Subsequently is presented practical part of the thesis, where the web application itself is designed according to the given requirements, along with technologies being selected for app's development. The last part solves problems that occur during the implementation itself.

Obsah

Seznam obrázků

Seznam tabulek

Seznam zdrojových kódů

1 Úvod

Doby, kdy si zaměstnanci ve směnných a restauračních provozech psali směny na papír, do kalendáře, nebo si je nosili pouze v hlavě, jsou snad už ve většině podnicích pryč. Dnes, kdy existují informační systémy na spousty firemních procesů, je jasné, že existuje několik takových systémů i pro plánování směn.

Inspirace pro zvolení tohoto tématu bakalářské práce byla osobní zkušenost s takovou aplikací, která je používána v jednom z největších obchodních řetězců restaurací u nás. I přes to, že zmiňovaný řetězec zaujímá na trhu velkou část, tak reference na systém, jak už vlastní, tak zaměstnanců a managerů nejsou ideální. Z toho důvodu bylo velkou motivací pokusit se vytvořit systém, který by byl schopný konkurovat již existujícímu řešení.

Většina takových restauračních řetězců využívá pro práci především brigádníky z řad studentů, kterým vyhovuje flexibilní brigáda kvůli škole a ostatním zájmům. Zde vyvstává otázka, jak naplánovat a obsadit směny, pokud chce firma nabídnout výše zmíněnou flexibilitu.

Řešením je plánování směn dle požadavků, které mohou zaměstnanci vložit do aplikace. Ta by měla směny uložit, poskytnout managerovi ke schválení a případnému upravení. Zaměstnanci je poté zasláno oznámení, ve kterém si může prohlédnout následující směny.

Nástrojů a aplikací řešící plánování směn bylo vytvořeno už mnoho, bohužel však není snadné najít systém, který by přesně odpovídal navrhovaným požadavkům. Cílem této práce je proto vytvoření funkční webové aplikace, ve které bude správa směn komplexní, zároveň ale uživatelsky přívětivá a efektivní.

2 Cíl práce

Cílem bakalářské práce je vytvořit funkční webovou aplikaci pro plánování směn. Pro tvorbu aplikace bude použit primárně JavaScript a frameworky na něm založené.

V teoretické části bude rozebrána definice webových aplikací a technologií s nimi spojenými. V další kapitole pak budou prozkoumány vybrané technologie, jak pro frontend, tak i pro backend. V neposlední řadě budou řešeny i různé varianty databáze, a to jak relační, dokumentové, tak i cloudové. Na základě této rešerše bude poté v praktické části vybrána kombinace technologií, která bude použita pro implementaci výsledné aplikace.

V praktické části bude nejdříve provedena analýza funkčních a nefunkčních požadavků, v návaznosti na to proveden návrh systému a databáze za pomoci vhodných modelovacích nástrojů. Z tohoto návrhu nakonec vznikne výsledná aplikace a budou popsány problémy, které při implementaci vznikly.

3 Webové aplikace

Webovou aplikaci lze definovat jako aplikaci, kterou není nutno instalovat na uživatelovo zařízení, místo toho je možné ji spustit z jakéhokoliv zařízení ve webovém prohlížeče, jelikož je spuštěna na straně serveru. Webová aplikace se může na první pohled jevit jako webová stránka, ale obvykle se jedná o složitější aplikaci provádějící komplexnější úlohy využívající databázi [1].

Výhody webových aplikací

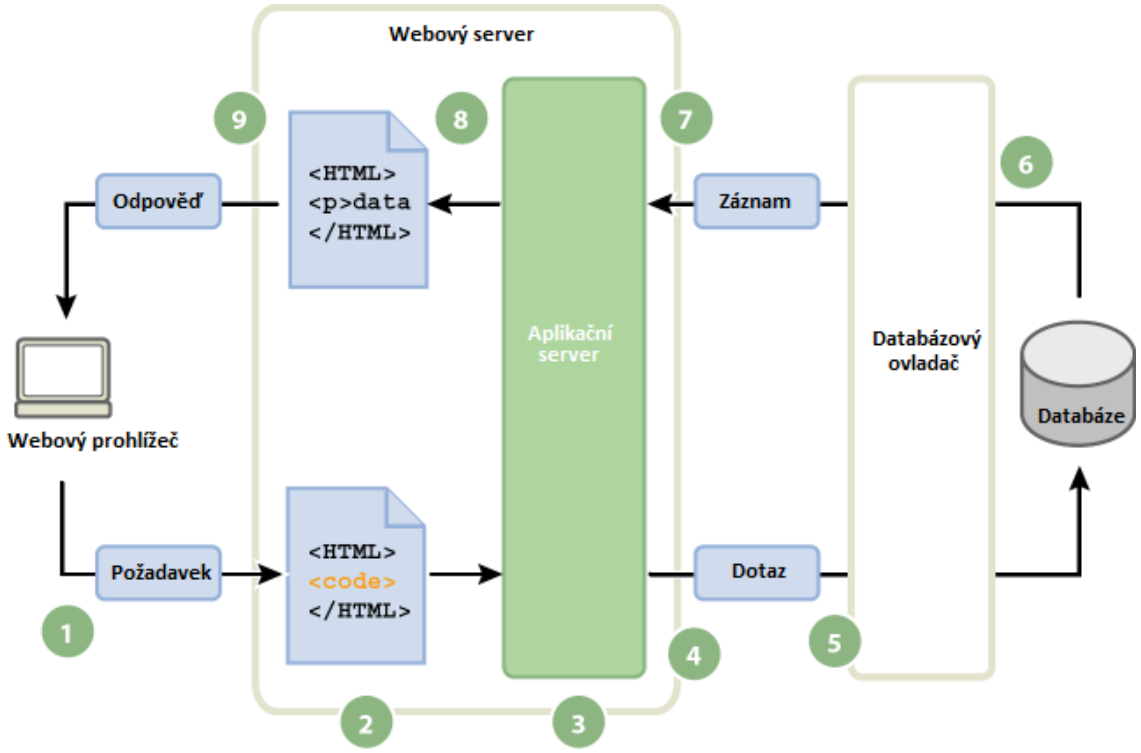
Za velké výhody je tak považováno především to, že není nutné nic instalovat, aktualizovat, stačí pouze webový prohlížeč a data uživatele jsou uchovány na serveru, takže k nim může přistupovat odkudkoliv [2]. Nezáleží ani na operačním systému klienta, a ani na zařízení, ze kterého k aplikaci přistupuje. Proto lze do jisté míry tvrdit, že je aplikace multiplatformní. Další velkou výhodou s tím spojenou, kdy spousta lidí vyřizuje většinu administrace, emailů a dalších věcí ze svých chytrých telefonů je to, že dnešní technologie dovolují přizpůsobit vzhled aplikace k daným rozlišením zařízeních a stává se tak responzivní. I z tohoto důvodu dnes stoupá návštěvnost webových stránek z jiných zařízeních než z osobních počítačů [3].

Nevýhody webových aplikací

Co se však jeví jako značná nevýhoda těchto aplikací je nutnost připojení k internetu, kdy při nedostatečné rychlosti může být práce s aplikací poměrně komplikovaná [2]. Mezi další nevýhodu patří také rizika spojená s přenosem dat po síti, která není dostatečně zabezpečená.

Princip fungování webových aplikací

Principem fungování takových aplikací je uživatelova interakce s webovou stránkou, kde má možnost například odeslat formulář nebo kliknout na tlačítko a tím spustit určitou událost, která odešle požadavek na server. Pokud se jedná o zobrazení nějakých dat z databáze, server pošle dál požadavek na data do databáze, ta mu je vrátí a on odešle zpátky jako odpověď statickou stránku s požadovaným obsahem webovému prohlížeči, tedy klientovi. Na následujícím obrázku je jednoduché schéma tohoto modelu.



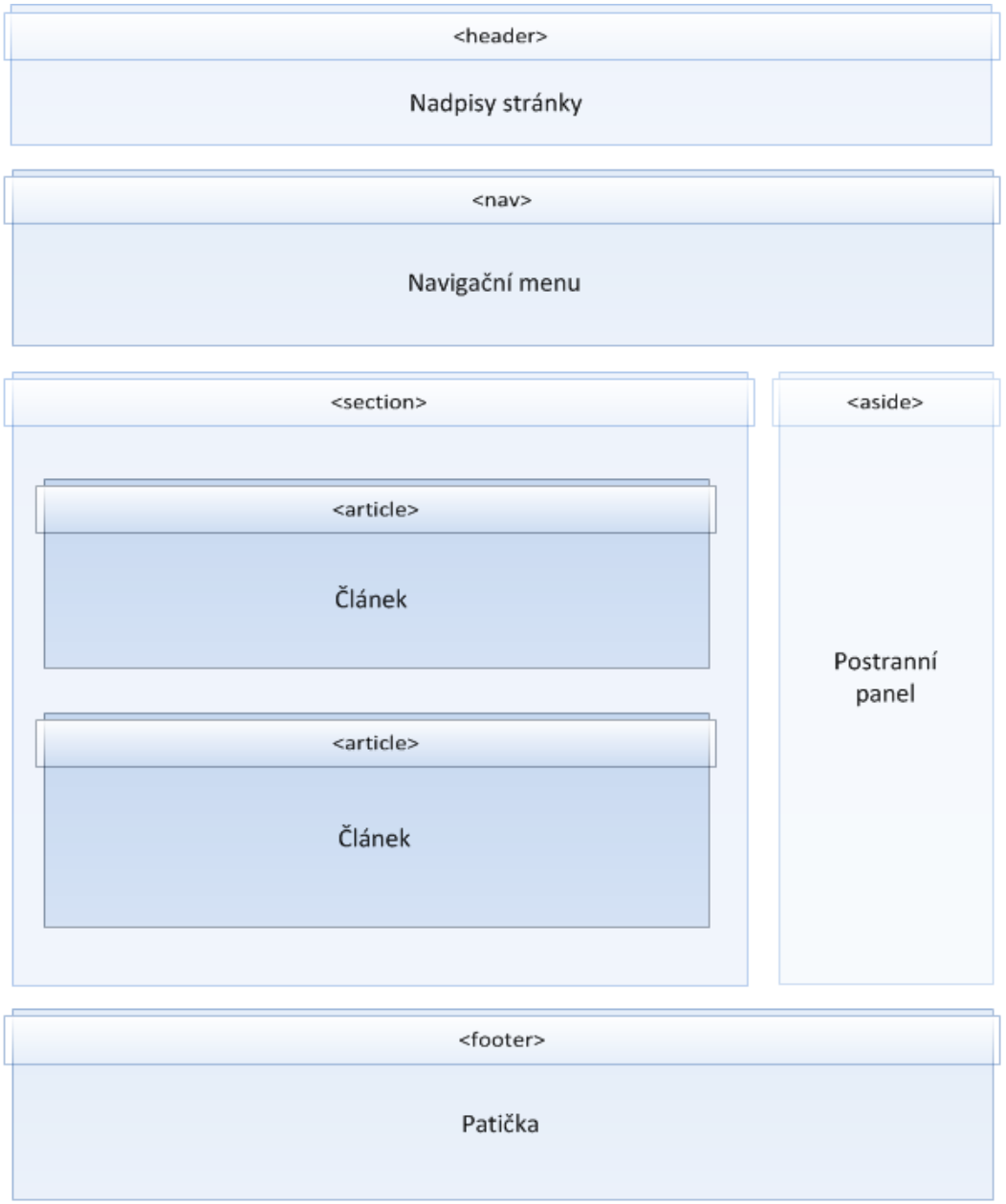
Obrázek 1 Princip fungování webové aplikace. Zdroj: [1]

3.1 HyperText Markup Language (HTML)

HTML je značkovací jazyk používaný pro vytváření webových stránek publikovaných na internetu. Je definován množinou značek (tzv. tagů) a jejich atributů. Tagy určují význam jednotlivých částí dokumentu a tvoří strukturu stránky. HTML má také specifikovanou základní strukturu, kterou je důležité v každém dokumentu dodržovat.

Jednotlivé tagy se uzavírají mezi špičaté závorky <>, a dále jsou děleny na párové a nepárové [4]. Párové značky mají svůj začátek, například `<html>` a pak jsou uzavřené pomocí lomítka ve značce, například `</html>`. Nepárové značky se pak mohou používat například pro vložení obrázku (`<img>`), nebo pro odsazení na nový řádek (`<br>`). Rozdíl mezi nimi je tedy takový, že mezi párové lze vkládat nějaký další obsah, do nepárových nikoliv. Mezi párové však lze vkládat kromě textového obsahu i další značky, které se postupně zanořují do sebe.

Atributy představují jejich jednotlivé vlastnosti, jako příklad lze použít `<body width=200px>`, kde atribut width určuje šířku obsahu dané značky.



Obrázek 2 Kostra HTML dokumentu. Zdroj: [5]

3.2 Cascading Style Sheets (CSS)

CSS je jazyk sloužící k popisu zobrazení a vzhledu stránek napsaných v HTML, XHTML nebo XML. S jeho pomocí je možno upravovat jednotlivým HTML blokům a tagům například barvu, velikost, odsazení, pozici a mnoho dalšího. Vznikl primárně z důvodu přehlednosti mezi vzhledem a strukturou stránky s obsahem. CSS je tak přelomem ve způsobu formátování webových stránek, jelikož neovlivňuje samotný obsah stránky a umožňuje tak vytvářet čisté a přísně strukturované HTML dokumenty [4]. To je výhodné hlavně pro zobrazení obsahu jinde než v běžných

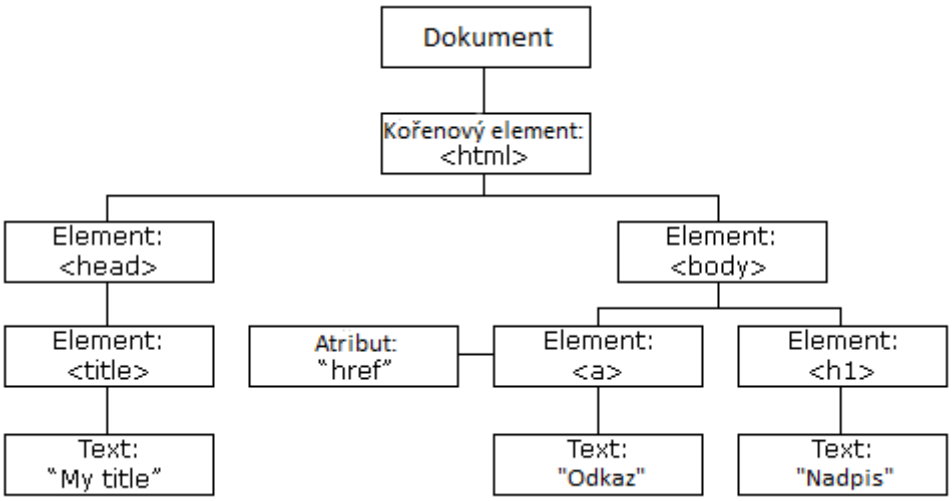
prohlížečích, včetně čtecích zařízeních pro nevidomé, nebo fulltextových vyhledávačů na webu.

Pro CSS jsou používány tři styly zápisu, a to řádkový, interní a externí. Řádkový stylopis se zapisuje přímo do atributu `style` dané HTML značky, interní je pak zapsán v hlavičce HTML dokumentu a externí zapisujeme do samostatného souboru s příponou `.css`, který můžeme používat mezi více dokumenty. Platí zde také pravidlo, že nejbližší stylopis k dané značce má přednost, a proto má nejvyšší váhu řádkový, poté interní a nakonec externí.

Obrázek 3 Deklarace CSS vlastností. Zdroj: [6]

3.3 Document Object Model (DOM)

DOM reprezentuje objekty, které tvoří strukturu a obsah webové stránky. Jedná se tak o rozhraní XML a HTML dokumentů, kde DOM reprezentuje dokument (webovou stránku) jako jednotlivé uzly (node) a objekty (objects), díky čemuž můžou jednotlivé skripty programovacích jazyků měnit strukturu, styl a obsah stránky. Díky DOM tedy můžeme propojit ostatní programovací jazyky s webovou stránkou a vytvořit jednoduchou webovou aplikaci.



Obrázek 4 DOM a reprezentace HTML stromu. Zdroj: [8]

Webová stránka je dokument, který může být buď zobrazen v internetovém prohlížeči, nebo jako HTML kód. V obou případech se však jedná stále o ten samý dokument. DOM také reprezentuje ten samý dokument, ale reprezentuje ho jako objektově orientovanou webovou stránku, která díky tomu může být upravována pomocí skriptovacích jazyků [7]. Jedním z jazyků, na kterém se často demonstruje DOM je JavaScript.

V následujícím příkladu JavaScript mění obsah elementu `p` a je zde ukázáno, jak může skriptovací jazyk zasahovat pomocí DOM do webové stránky.

```
<html>
<body>
<p id="ukazka"></p>
<script>
document.getElementById("ukazka").innerHTML = "Ahoj světe!";
</script>
</body>
</html>
```

Zdrojový kód 1 Ukázka JavaScriptu měnící DOM webové stránky. Zdroj: [autor]

3.4 Asynchronous JavaScript and XML (AJAX)

AJAX byl publikován Jamesem Garrettem v roce 2005 a je kombinací několika stávajících technologií, díky kterým poskytl uživatelům pokročilý přístup k procházení webu a je tak velmi využívaný v současných webových aplikacích [9]. S využitím technologií XHTML, CSS, JavaScriptu, XML spolu s XMLHttpRequest přiblížil webové aplikace ještě blíže těm desktopovým.

Principem AJAXu je schopnost odeslat data na server a následně je získat zpět bez nutnosti znovu načítání celé stránky za pomoci asynchronního zpracování [10]. Může mít tak mnohá využití při práci s formuláři, kde fungují našeptávače, nebo jako validátory dat do nich zadaných. V podkapitole Single-page aplikace je pak na obrázku (Obrázek 5) zobrazen princip graficky. AJAX díky tomu umožňuje vytvářet mnohem přívětivější a přístupnější weby a webové služby, a jelikož nevyužívá žádnou novou technologii, tak se vývojáři nemusí učit nic nového. Dá se také dobře integrovat s funkcionalitou danou webovými prohlížeči.

Nevýhodou v této technologii může být naopak načítání obsahu při vyšší internetové odezvě, kde není ošetřena signalizace načítání pomocí textu, nebo nějaké animace [9]. Uživatel si může myslet, že aplikace nereaguje, a tak spustí kliknutím událost několikrát a tím zatíží server víc, než kdyby se načítala celá stránka. Dále také nemusíme dostat odpověď na požadavek, a proto by měla stránka umět řešit i chybové stavy.

3.5 Simple Object Access Protocol (SOAP)

Podle [11], je nejnovější verze SOAP odlehčený protokol určený pro výměnu strukturovaných informací v distribuovaném decentralizovaném prostředí. Pro samotnou síťovou komunikaci nejčastěji používá HTTP protokol, může však využívat i protokoly pro e-mailové služby atd. Dále využívá technologie XML k definování rozšiřitelného rámce pro zasílání zpráv, který také poskytuje konstrukci zprávy zaměnitelnou v různých základních protokolech. Rámec byl navržen nezávisle na konkrétním programovacím modelu a různých specifických implementacích. SOAP tak pokrývá čtyři hlavní oblasti [12]:

- formát zprávy pro jednosměrnou komunikaci, který popisuje, jak může být zpráva zabalena do XML dokumentu,
- jakým způsobem má být zpráva poslána: přes HTTP protokol (webové služby), STMP protokol (e-mailové služby), atd.,
- sada pravidel, která je potřeba dodržovat při zpracování SOAP zprávy,
- sada konvencí, jak změnit RPC volání na SOAP zprávu a zpět.

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
<soap:Body>
  <m:GetPrice xmlns:m="https://www.w3schools.com/prices">
    <m:Item>Apples</m:Item>
  </m:GetPrice>
</soap:Body>
</soap:Envelope>
```

Zdrojový kód 2 Příklad typického SOAP požadavku. Zdroj: [13]

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
<soap:Body>
  <m:GetPriceResponse xmlns:m="https://www.w3schools.com/price
s">
    <m:Price>1.90</m:Price>
  </m:GetPriceResponse>
</soap:Body>
</soap:Envelope>
```

Zdrojový kód 3 Příklad typické SOAP odpovědi. Zdroj: [13]

3.6 Representational state transfer (REST)

REST je architektura, díky které je možno jednoduše přenášet data ze serveru pomocí jednoduchých HTTP volání. Je vzorem pro práci s daty, který se stal standardem pro návrh služeb v moderních webových aplikacích. Zatímco tradiční přístup k webovým službám založený na protokolu SOAP využívá plnohodnotné objekty se vzdáleným voláním procedur, REST se zabývá pouze datovými strukturami a přenosem jejich stavu. Díky jednoduchosti této architektury a přizpůsobení se HTTP protokolu se REST začal hojně využívat v aplikacích definovaných jako Web 2.0 [14]. Pro práci s daty jsou definovány operace Create, Read, Update, Delete, které jsou popsány v následující tabulce.

Tabulka 1 CRUD operace v architektuře REST. Zdroj: [14]

CRUD operace	HTTP metoda	Vstupní formát	Výstupní formát
Create (Vytvořit)	POST	Kódované data z formuláře	Status 201 - CREATED, nebo chybový status
Read (Číst)	GET	Žádný, nebo ID žádaného objektu	Celá kolekce, konkrétní objekt podle id, nebo chybový status
Update (Upravit)	PUT	Kódované data z formuláře	Status 200 OK, nebo chybový status
Delete (Smazat)	DELETE	ID daného objektu, který má být smazán	Status 200 OK, nebo chybový status

Pomocí těchto operací lze kompletně spravovat data uložená na serveru. REST vrací data primárně v JSON formátu, což je specifický textový řetězec, ve kterém může být zapsán libovolný objekt nebo pole v neomezené hierarchii. Takový formát by se mohl jevit jako nevýhodný při správě objektů v aplikaci, naštěstí dnes již většina programovacích jazyků umožňuje snadný převod mezi objekty a JSON formátem.

3.7 Multiple-page aplikace (MPA)

Vícestránková aplikace, dále jen MPA je tradičním druhem webové aplikace. To znamená, že pokaždé, když aplikace potřebuje zobrazit data nebo je odeslat zpět na server, tak by měla být načtena celá nová HTML stránka ze serveru do webového prohlížeče. Na tomto principu není nic špatného, ale pokud má stránka bohaté uživatelské rozhraní, může se stát její načítání se spoustou dat poměrně složité.
Dnes díky AJAXu, který mění obsah bez nutnosti aktualizace celé stránky se problému s načítáním spousty dat lze vyhnout. Bohužel AJAX přidává samotné stránce složitost, a proto je jednodušší vyvinout jednostránkovou aplikaci [15].

Výhody MPA:

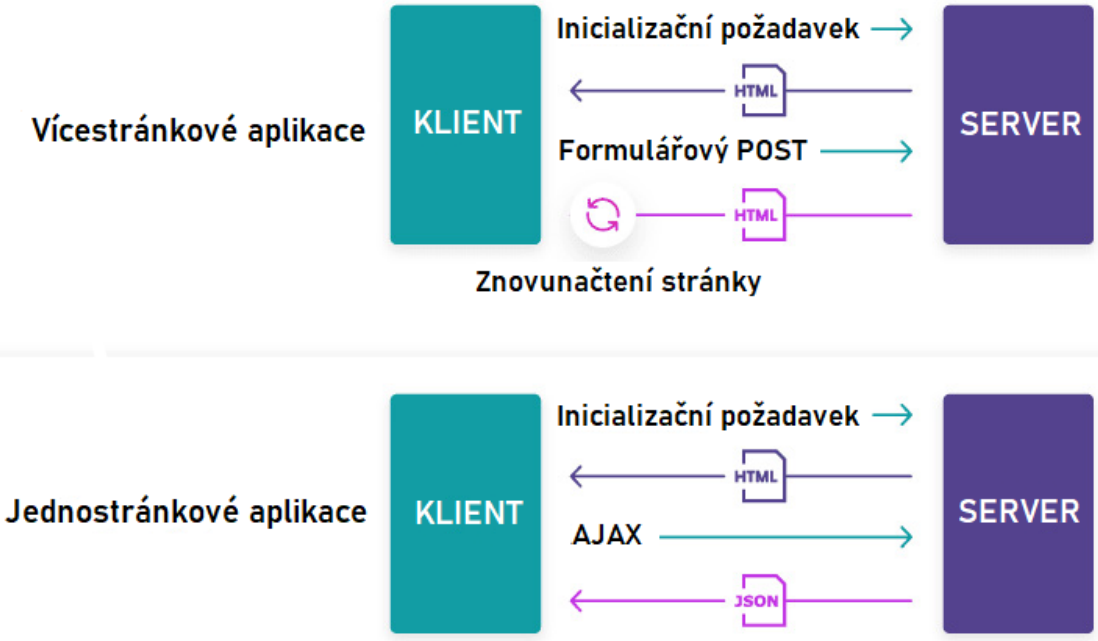
- nejlepší přístup pro komplexní aplikace,
- lepší SEO optimalizace než u jednostránkových aplikací,
- lepší zabezpečení a integrita dat,
- větší škálovatelnost.

Nevýhody MPA:

- vývoj prezentační a serverové části je úzce spjat,
- vývoj je poměrně složitý.

3.8 Single-page aplikace (SPA)

V případě jednostránkových aplikací, dále jen SPA, je celý web načtený jako jedna aplikace. To znamená, že se úplně oddělí prezentační vrstva (frontend) od té serverové (backend). SPA se pak stará právě o frontend část.
SPA si při prvotním načítání stránky stáhne potřebné nástroje na vykreslování stránek a vykreslí se první zobrazení, takzvaný pohled. Při potřebě generování nového pohledu je však vykreslený lokálně na straně klienta a dynamicky vložený do DOM stránky, zatímco původní pohled se z DOM odstraní [16]. Spolu s tím se pošle požadavek na server (pokud je potřebný), který obvykle vrátí data ve formátu JSON, nebo XML a ty jsou následně doplněné do pohledu. Výsledkem je přesměrování a vykreslení nové stránky bez nutnosti znovu načítání stránky.



Obrázek 5 Rozdíl mezi klient/server komunikací MPA a SPA. Zdroj: [17]

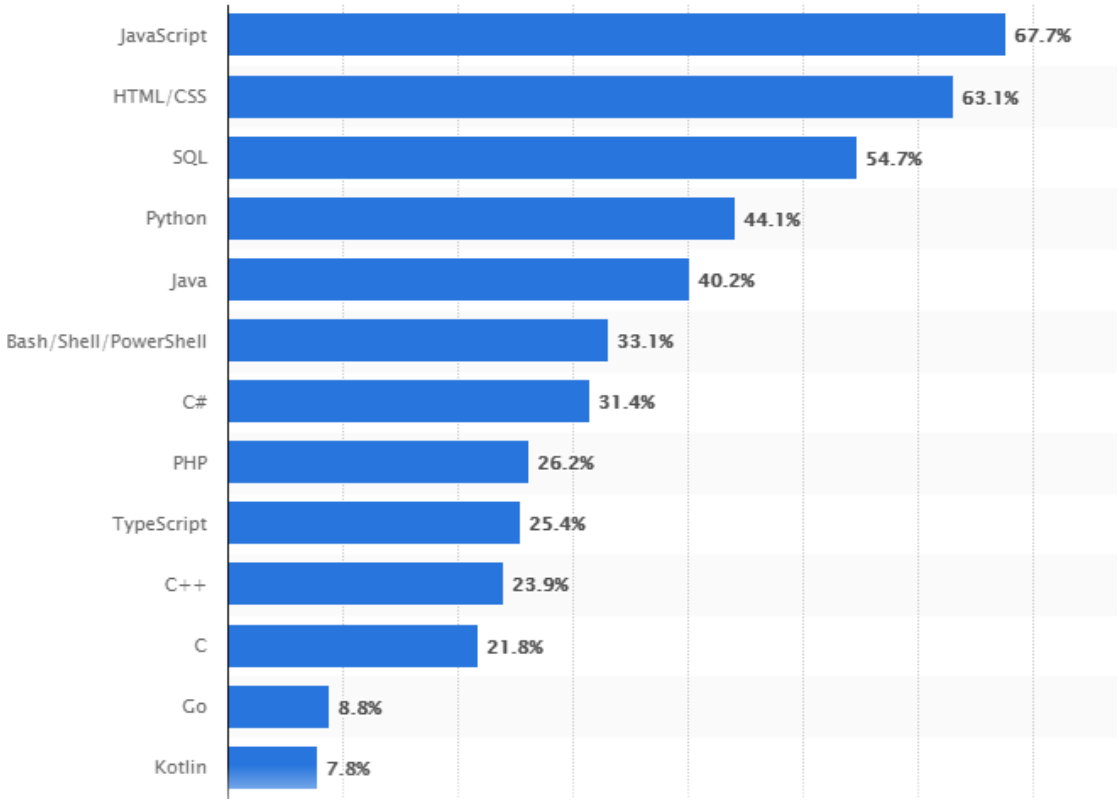
3.9 Progresivní webové aplikace (PWA)

Progresivní webová aplikace je typem webové aplikace, kterou lze používat v prohlížeči jako běžnou webovou stránku a zároveň jako mobilní aplikaci. PWA tak přebírá některé funkce z nativních aplikací, například že fungují offline, posílají push notifikace, nebo využívají hardware daných zařízení pro určení polohy atd. Aplikace je progresivní tehdy, když splňuje několik daných požadavků od Googlu, který je kmotrem této technologie.

Google definuje PWA tak, že jsou aplikace spolehlivé, poutavé a rychlé [18]. U spolehlivosti je kladen důraz na to, aby aplikace nebyla závislá na připojení k síti a dalo se s ní pracovat i offline. Rychlost je myšlena tak, že aplikace rychle a plynule reaguje na uživatele. Poslední z trojice, poutavost, tkví v tom, že její uživatelské rozhraní připomíná opravdu nativní aplikaci a uživatele dokáže vtáhnout do děje. Tato trojice se tak snaží upoutat a podmanit zákazníka, který na webovou stránku přišel, a podat mu obsah zajímavou a pro něj příjemnou formou.

4 Rešerše dostupných technologií

Tato kapitola je soustředěna primárně na jazyk JavaScript a s ním spojené knihovny a frameworky. Důvodem výběru je vysoká popularita jazyka a možnost pracovního uplatnění při znalosti daných technologií [19]. Nachází se zde několik variant, ze kterých lze vybrat vhodné řešení pro výslednou aplikaci, a proto jsou zde blíže představeny a popsány.



Obrázek 6 Nejpoužívanější programovací jazyky roku 2020. Zdroj: [20]

4.1 JavaScript

Jazyk JavaScript je základem, ze kterého vycházejí následující knihovny a frameworky. Byl představen v roce 1995 jako jedna z cest umožňující vkládat určitou dynamiku a inteligenci do statických webových stránek. Nejprve byl přidán do prohlížeče od společnosti Netscape, která jazyk vyvinula, nakonec byl však implementován i do všech dalších grafických webových prohlížečů. Jeho hlavní a neopomenutelnou vlastností je to, že pracuje na straně klienta, což v praxi znamená, že není potřeba při interakci s uživatelem znovu načítat celou stránku. Při prvním načtení se tak načte skript ke klientovi a pak už pracuje pouze

na jeho prohlížeči bez nutnosti odesílání dat na server. Tento princip je tak hojně využívaný například při ověřování formulářů, kde jsou nastavena určitá pravidla pro vyplnění, které se ověří ještě před odesláním správných dat na server.
Po rozšíření jazyka mezi další prohlížeče vznikl dokument, který popisuje standardy, jak by měl JavaScript fungovat. Tento standard se nazývá ECMAScript a je pojmenovaný podle mezinárodní neziskové organizace Ecma [21]. ECMAScript a JavaScript lze však považovat za ekvivalentní.
Tento jazyk je využívaný i jinde než webovými prohlížeči. MongoDB, nebo CouchDB databáze ho používají jako svůj skriptovací a dotazovací jazyk a další rozšířená technologie Node.js ho využívá například k psaní internetových aplikací, především tak webových serverů.

4.1.1 Node.js

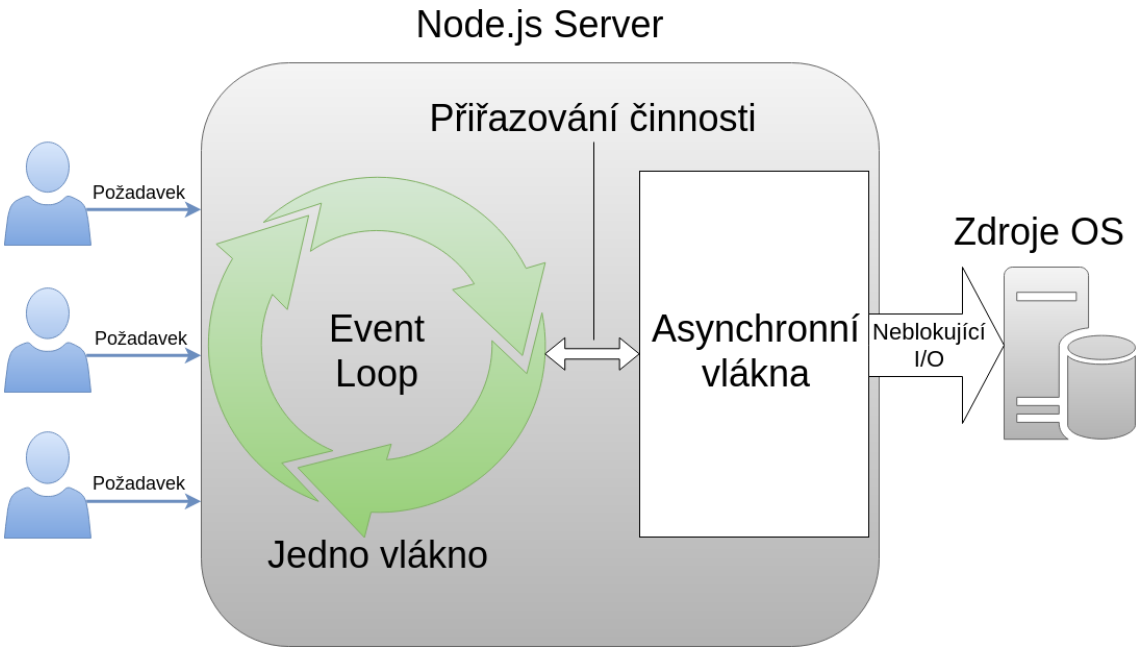
Node.js je prostředí, pomocí kterého lze spouštět JavaScriptový kód mimo webový prohlížeč. Je postaven na Chrome V8 JavaScript enginu, a proto základ tohoto prostředí je stejný s webovým prohlížečem Google Chrome [22]. Byl navržen primárně pro vytváření serverových částí webových aplikací (backend), ale lze v něm napsat i jednoduché aplikace.

Instalace

Pro vývoj Node.js aplikací je třeba mít jeho prostředí nainstalované na svém stroji. Instalační soubor lze získat jednoduše na oficiálních stránkách Node.js. V samotné instalaci je také obsažen balíčkový systém `npm`, který se stará o správu závislostí a knihoven.

Funkcionalita prostředí

Princip Node.js aplikace je takový, že běží jako jeden proces, který nevytváří vlákna pro každý požadavek. Poskytuje však ve své knihovně sadu asynchronních I/O primitiv, které zabraňují blokování kódu. Příklad takového fungování lze vidět níže.



Obrázek 7 Node.js architektura. Zdroj: [23]

4.2 Frontend řešení

Frontend, nebo také prezentační vrstva aplikace je primárně soustředěna na uživatelské rozhraní aplikace, a proto je velmi spjata s již zmiňovanými technologiemi, jako jsou DOM, HTML, CSS, AJAX a Single-page aplikace. Jedná se o tu část aplikace, která řídí interakci mezi uživatelem a danou aplikací. Vytvoření kvalitní frontend části tak tkví především ve funkčnosti, intuitivnosti a jednoduché ovladatelnosti vizuálního prostředí dané aplikace.

4.2.1 Vue.js

Vue je progresivní JavaScriptový framework používaný pro tvorbu uživatelského rozhraní ve webových aplikacích. Řadí se ke skupině frameworků a knihoven, jako jsou například React, Angular, Ember a mnoho dalších. Na rozdíl od jiných je ale Vue od základu navržen tak, aby byl postupně adaptovatelný, což přináší výhodu hlavně při implementaci jeho funkcionalit do již fungujících projektů. Na druhou stranu se Vue prezentuje také jako framework, ve kterém je možné vytvářet sofistikované single-page aplikace [24].

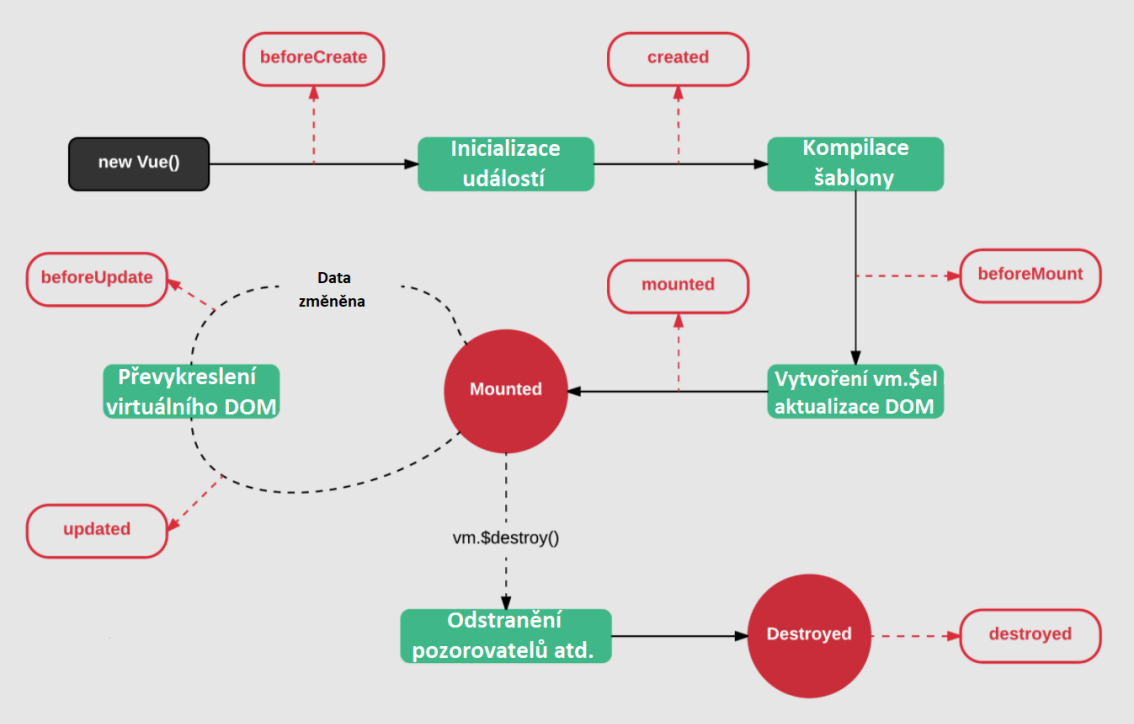
Instalace

Vue nabízí několik variant, jak ho použít. První variantou je vložení knihoven pomocí tzv. CDN (Content delivery network) do tagu `<script>`. To umožní HTML

dokumentu práci s přidanými knihovnami a vytvoření Vue aplikace. Tento způsob se využívá spíše při menších projektech, nebo pokud je třeba ve stávajícím projektu přidat nějakou jeho funkcionalitu. Dokazuje se zde však výše zmíněná adaptabilita s různými projekty. Doporučovaným způsobem je ale instalace pomocí balíčkového systému `npm`, který je vhodný již pro složitější projekty. V neposlední řadě Vue nabízí také oficiální CLI (příkazový řádek), který umožňuje vytvořit základní kostru aplikace.

Funkcionality frameworku

Vue aplikace se skládá z komponent, využívá šablony založené na HTML, je reaktivní, poskytuje mnoho způsobů přechodů a animací a podporuje směrování. Do komponent se vkládají základní HTML prvky a zajišťují tak znovu použitelnost kódu [26]. Každá komponenta je v podstatě instance Vue aplikace s předdefinovanými možnostmi. Šablony zas umožňují vazbu vykresleného DOMU s daty se zmiňovanou instancí. Díky reaktivitě a jejím závislostem systém přesně ví, kdy má které komponenty znovu vykreslit [25].



Obrázek 8 Instance Vue aplikace. Zdroj: [28]

```
Vue.component('button-counter', {
  data: function () {
    return {
      count: 0
    }
  },
  template: '<button v-
on:click="count++">You clicked me {{ count }} times.</button>
'
})
```

Zdrojový kód 4 Příklad Vue komponenty. Zdroj: [26]

4.2.2 React

React je flexibilní JavaScriptová knihovna pro tvorbu uživatelského rozhraní, které je skládáno z několika izolovaných částí kódu, nazývajících se komponenty [29]. Je vyvíjen společností Facebook a používá se především pro tvorbu single-page aplikací, případně při použití frameworku React Native, který je prakticky stejný jako React, také pro tvorbu mobilních aplikací. V Model-View-Controller architektuře tvoří právě view, pohled, který je prezentován uživateli. Obvykle jsou s použitím Reactu používány i další knihovny, jelikož základní knihovna se stará pouze o vykreslování dat do modelu DOM. Mezi nejužívanější patří například knihovna Redux starající se o správu stavů, knihovna React Router starající se o směrování, nebo knihovna Create-React-App, která pomáhá se základní konfigurací a inicializací aplikace [30]. Knihoven, které lze v aplikaci postavené na Reactu použít existuje několik, což značně usnadňuje práci při vývoji.

Instalace

Instalace Reactu je podobná, jako u zmiňovaného Vue.js v kapitole 4.2.1. Pro jednoduché použití lze knihovny načíst pomocí CDN, ale pro produkční prostředí je lepší instalovat pomocí prostředí Node.js s využitím jeho balíčkových systémů.

Funkcionality frameworku

React aplikace využívá JSX, což je rozšíření syntaxe jazyka JavaScript podobná syntaxi HTML, která umožňuje popis vzhledu uživatelského rozhraní [31]. Jak bylo výše zmíněno, tak se aplikace skládá z komponent. Existují dva druhy komponent, a to funkční a třídní [32]. Komponenta se stará o vykreslení daného React elementu

vytvořeného pomocí JSX. Dále také využívá metody životního cyklu, které jsou důležité k uvolnění zdrojů komponent, když jsou zničeny.

```
const name = 'Josh Perez';
const element = <h1>Hello, {name}</h1>;

ReactDOM.render(
  element,
  document.getElementById('root')
);
```

Zdrojový kód 5 Vkládání výrazu do JSX v Reactu. Zdroj: [31]

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}

class Welcome extends React.Component {
  render() {
    return <h1>Hello, {this.props.name}</h1>;
  }
```

Zdrojový kód 6 Rozdíl mezi funkční a třídní komponentou v Reactu. Zdroj: [32]

4.2.3 Angular

Angular je framework a vývojová platforma sloužící k tvorbě sofistikovaných single-page webových aplikací. Využívá komponentové architektury se službami a namísto čistého JavaScriptu využívá jazyk TypeScript [33]. Existuje také jeho předchůdce, AngularJS, který však není s tímto frameworkem navzájem kompatibilní a staví na jiných principech a na odlišném jazyku. Proto se také tento framework dříve označoval jako Angular 2 [34].

Instalace

Instalace Angularu je prováděna také pomocí prostředí Node.js a jeho balíčkového systému `npm`. Pomocí něho je možné nainstalovat Angular CLI, který umožňuje vytvářet a generovat projekty a knihovny s nimi spojené.

Funkcionality frameworku

Stávající Angular je založen na komponentách a službách. Skládá se z TypeScript tříd, které jsou označené jako `decorators`. Pomocí těchto označení je specifikováno, jak má Angular třídy používat. Jednotlivé komponenty jsou třídy definující data a logiku a skládají se z nich pohledy (views) [35]. Třída komponenty je spojena se šablonou a definuje tak samotný pohled. Služby umožňují přístup dalších dat a logiky pro jednu až několik komponent. Využívají k tomu `dependency injection`. Dependency injection je technika, která vkládá závislosti mezi jednotlivé komponenty tak, aby jedna komponenta mohla používat druhou. Angular také využívá své vlastní moduly (NgModul), které slouží k členění kódu.

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
  })
export class AppComponent {
  title = 'Hello World';
  }
```

Zdrojový kód 7 Definice komponenty v Angularu. Zdroj: [34]

4.3 Backend řešení

Backend, nebo také serverová vrstva aplikace slouží k logickým operacím s daty, zabezpečením samotných dat před neoprávněným přístupem a také ke komunikaci s databází a frontendem aplikace. Dotýká se také již zmiňovaných technologií, jako je architektura REST a protokol SOAP.

4.3.1 Express

Express je minimalistický a flexibilní Node.js framework, který poskytuje sadu funkcí pro webové a mobilní aplikace. Díky několika metodám obsluhujících HTTP protokol je možné vytvářet API snadno a rychle [36]. Proto se také Express využívá pro tvorbu REST API, které poskytuje data pro klientskou část aplikace a tím se stávají obě části aplikace nezávislými.

Instalace

Jako ostatní zmiňované frontendové technologie, i Express využívá prostředí Node.js a jeho balíčkový systém `npm`, pomocí kterého může být nainstalován.

Funkcionality frameworku

Mezi jeho hlavní funkcionalitu patří to, že umožňuje nastavit middleware[1] pro odpovědi na HTTP požadavky, dokáže definovat směrovací tabulku která určuje, jaká akce bude provedena při volání dané URL adresy a dokáže dynamicky vykreslovat HTML stránky na základě předávaných argumentů do šablon [37].

```
const express = require('express')
const app = express()
const port = 3000

app.get('/', (req, res) => {
  res.send('Hello World!')
})

app.listen(port, () => {
  console.log
(`Example app listening at http://localhost:${port}`)
})
```

Zdrojový kód 8 Ukázka webové cesty v Express aplikace. Zdroj: [36]

4.3.2 Koa.js

Koa je nový webový framework vyvíjený stejným týmem, kterým je vyvíjen Express. Naproti Expressu se snaží být menším, expresivnějším a robustnějším základem pro webové aplikace a API [38]. Využívá asynchronních funkcí, které dovolují zrušit zpětná volání a výrazně tak zvyšují možnost zpracování chyb. Ve svém jádru Koa nesdružuje žádný middleware a poskytuje sadu několika elegantních metod pro rychlé a snadné psaní serverů.

Instalace

Koa pro svou funkčnost potřebuje Node ve verzi 7.6.0, nebo vyšší, který podporuje ES2015 (ECMAScript 2015) a asynchronní funkce. Instalace probíhá pak stejně

[1] software, který poskytuje aplikacím služby nad rámec služeb poskytovaných operačním systémem

jako u předchozích frameworků a knihoven, a to pomocí nějakého správce verzí a balíčků [38].

Funkcionality frameworku

Koa aplikace je objekt obsahující několik middleware funkcí, které se na vyžádání komponují a s využitím principu zásobníku jsou pak prováděny za sebou. Mezi podobné middleware systémy, se kterými se lze setkat patří například Ruby's Rack, nebo Connect. Tyto systémy byly navrženy tak, aby byla zajištěna vysoká hladina tzv. „cukru" v jinak nízkoúrovňových vrstvách middlewaru, díky čemuž je zlepšena robustnost a interoperabilita [38]. I přes to, že si Koa zakládá na malé velikosti, obsahuje několik metod pro různé úkoly, jako je správa mezipaměti, podpora proxy serveru a mnoho dalších.

Koa, stejně jako Express umožňuje vytvářet REST API, které poté poskytují data mobilním, webovým a dalším aplikacím.

```
const Koa = require('koa');
const router = require('koa-router')();
const app = new Koa();

const car = {
  type: 'Ford'
  year: '1999'
  model: 'Focus'
};

router.get('/car', async (ctx, next) => {
  ctx.body = car;
  await next();
});
app.use(router.routes());
module.exports = app;
    app.use(router.routes());
    module.exports = app;
```

Zdrojový kód 9 Ukázka webové cesty v Koa aplikaci. Zdroj: [39]

4.3.3 Hapi

Hapi (zkratka pro HTTP API) je bohatý framework pro tvorbu aplikací a služeb. Původně byl navržen jen pro rychlý vývoj REST API služeb využívající JavaScript, ale postupem času vyrostl a stal se z něj plnohodnotný webový framework.

Obsahuje mnoho funkcí pro šablonování, ověřování vstupů, autentizaci, podporu pro aplikace v reálném čase a mnoho dalšího [40].

Instalace

Instalace Hapi probíhá stejně, jako u předchozích, a to pomocí nainstalovaného balíčkového systému `npm` a příkazů `npm init` a `npm install @hapi/hapi`.

Funkcionality frameworku

Hapi, jak již bylo zmíněno, umožňuje vytvořit server obsahující cesty, a tak vytvořit jednoduše REST API. Poskytuje ověřování založené na konceptu schémat a strategií, usnadňuje konfiguraci ukládání do mezipaměti na stranách klienta i severu, dokáže zabezpečit cookies soubory, stará se o směrování, validaci dat, testování a mnoho dalšího. Poskytuje tak podobnou škálu možností jako Express.

```
'use strict';
const Hapi = require('@hapi/hapi');
const init = async () => {
  const server = Hapi.server({
    port: 3000,
    host: 'localhost'
  });
server.route({
  method: 'GET',
  path: '/',
  handler: (request, h) => {
  return 'Hello World!';
  }
});
await server.start();
  console.log('Server running on %s', server.info.uri);
};
process.on('unhandledRejection', (err) => {
  console.log(err);
  rocess.exit(1);
});
init();
```

Zdrojový kód 10 Ukázka webové cesty v Hapi aplikaci. Zdroj: [41]

4.4 Databáze

Databáze je softwarový systém sloužící k ukládání dat a operacím nad nimi. Existuje několik struktur, pomocí nichž se data do databáze ukládají. Pro tuto práci, v návaznosti na předchozí technologie, je důležité si definovat struktury dvě, a to relační a dokumentovou. Z toho důvodu byly vybráni zástupci těchto struktur a jsou zde blíže popsány.

4.4.1 MongoDB

MongoDB je dokumentová databáze, která data ukládá flexibilně do dokumentů, mající podobný formát jako JSON soubory. Hlavní benefit této architektury je to, že lze kdykoliv změnit datovou strukturu daného dokumentu a nepotřebuje tak pevné schéma, jako SQL databáze. Dokumentový model také dokáže snadno namapovat data z databáze do objektů aplikace, která s daty pracuje. Dále poskytuje například ad-hoc dotazy, vyvažování zátěže, replikace, indexování a agregace [42]. Rovněž je databáze podporována několika velmi populárními programovacími jazyky, jako je právě JavaScript a jeho frameworky, ale také Java, Python, C++ nebo C#.

Instalace

MongoDB nabízí dvě možnosti, jak ho používat. A to buď jako lokální server, který má dvě edice (Community a Enterprise) a lze je stáhnout z oficiálních stránek databáze, nebo jako cloud řešení, tzv. MongoDB Atlas, kde je poskytnuta základní verze databáze zdarma. Cloud řešení je výhodnější v tom, že není třeba nic instalovat a databáze je dostupná na síti.

4.4.2 PostgreSQL

PostgreSQL je objektově-relační databázový systém s otevřeným zdrojovým kódem. Využívá a také rozšiřuje jazyk SQL, disponuje mnoho funkcemi, které bezpečně nakládají s nejkomplikovanějšími datovými úlohami. Znám je především díky své osvědčené architektuře, spolehlivosti, robustní sadě funkcí, integritě dat a rozšiřitelnosti [43]. Tyto vlastnosti získal díky třicetiletému aktivnímu vývoji, jelikož jeho začátky sahají až do roku 1986. Jednou z dalších výhod je podpora všech hlavních operačních systémů.

Snaží se vycházet ze standardu SQL, kdy v tuto chvíli splňuje 170 ze 179 povinných funkcí pro shodu s SQL [43], a proto lze k datům v databázi přistupovat pomocí SQL dotazů.

Na rozdíl od dokumentových databází, tak relační jsou založené na předem definovaných tabulkách, ve kterých jsou uložená data jednoho typu. Proto ve většině případů před vznikem relační databáze musí vzniknout tzv. relační model, který udává, jak budou data v tabulkách uložena a jaké mezi nimi budou vazby.

Instalace

PostgreSQL je možné nainstalovat jako jiné aplikace, instalační balíček se nachází přímo na oficiálních webových stránkách, které disponují různými verzemi pro nejznámější operační systémy.

4.4.3 Firebase

Firebase je platforma pro vývojáře mobilních a webových aplikací obsahující řadu knihoven pro ně. Je poskytována společností Google a mezi její hlavní funkcionality se řadí sběr analytických dat, odesílání push notifikací, nebo vzdálená konfigurace aplikací.

Realtime databáze

Realtime databáze je cloudovým řešením, které ukládá data v JSON formátu a umožňuje je přenášet v reálném čase (realtime) mezi klienty. Cloudové databáze jsou v principu databáze, které běží na nějaké cloudové platformě a přístup k nim je poskytnut po síti jako služba, a z toho důvodu se značí jako „Database-as-a-service“, česky „Databáze jako služba“. Firebase realtime databáze funguje na tom samém principu, z čehož plyne, že některé služby pro jejich výpočetní náročnost a paměť mohou být bohužel zpoplatněny.

Naproti tomu hlavní benefit této databáze je především zmiňovaný přenos dat v reálném čase, kdy aplikace komunikují pomocí HTTP požadavků s databází a zajišťují tak okamžitou synchronizaci dat. Mezi další výhody patří velká podpora pro různé jazyky a frameworky, posílání oznámení o změnách dat, korespondující datový model s JavaScript objekty a mnoho dalšího [44]. Díky těmto vlastnostem využívají databázi například chatovací aplikace, kde se okamžitý přenos dat a oznámení perfektně hodí.

Firestore

Firestore je novější typ databáze určený primárně pro vývoj rozsáhlejších mobilních, webových a serverových aplikací dědící funkcionality realtime databáze. Nabízí však určitá zlepšení, díky kterým by měl být vývoj přívětivější a snazší. Poskytuje nový datový model a podporuje složitější a rychlejší dotazy [45]. Je tedy vhodnou volbou, pokud zákazník požaduje provádět rozsáhlejší dotazy, filtrování dat, ukládání dat do kolekcí, uchovávání většího množství dat a práci s offline lokálními daty.

Srovnání databází

Pokud by mě měla sloužit databáze pouze k synchronizaci dat, jednoduchému dotazování se na ně a uchovávala by menší množství dat s jednoduchou strukturou, tak podle [45] je vhodnou volbou právě Realtime databáze.

Pokud by však databáze měla podporovat offline podporu pro klienty všech technologií, jak mobilních, tak webových, měla by mít skoro 100% dostupnost a složitější strukturu dat, tak je doporučeno zvolit právě druhou variantu, a to Firestore.

Obrázek 9 Princip aktualizace Firebase databáze v reálném čase. Zdroj: [46]

Instalace

Jelikož jsou databáze cloudové, tak není potřeba nic instalovat. Ale i přesto je nutné splnit určité kroky k tomu, aby bylo možné databázi používat.

Nejprve je třeba si založit účet na oficiální webové stránce, poté vytvořit Firebase projekt, vybrat jednu ze zmiňovaných variant a založit databázi, která je od té doby dostupná na dané URL adrese. Následně je třeba naimportovat do vytvářené aplikace pomocnou knihovnu pro daný programovací jazyk, který bude přistupovat k databázi.

5 Analýza a návrh webové aplikace

Analýza a návrh aplikace je velice důležitou částí při vývoji aplikace. Jsou zde vytyčeny hlavní požadavky na systém, z kterých poté vzniká návrh samotné aplikace. V návrhu je třeba definovat strukturu dat, jednotlivé komponenty aplikace a v neposlední řade technologie pro její implementaci.

5.1 Popis problému

Jak již bylo v úvodu zmíněno, aplikace řeší správu směn jednotlivých zaměstnanců. Klíčovou vlastností je možnost volby směn v jednotlivých dnech a hodinách. Zaměstnanec si volí směny na týden dopředu dle jeho potřeb, v jednotlivých dnech může volit, kolik hodin chce v práci strávit. Po tomto výběru odesílá směny manažerovi, který zkontroluje stav jeho zaměstnanců na dané dny a směny buď přijme tak, jak si zaměstnanec zažádal, upraví podle jeho potřeb nebo je zamítne. Následně jsou zaměstnanci poslány platné modifikované směny, podle kterých chodí do práce.

5.2 Požadavky

Požadavky na aplikaci vycházejí primárně z předcházejícího popisu problému plánování směn. Základní požadavky jsou odrazem existují aplikace, ale je zde i několik vlastních požadavků, které byly konzultovány s nezávislými zaměstnanci využívajících takovou aplikaci. Požadavky jsou rozděleny na funkční a nefunkční, kde funkční jsou dále děleny podle priorit. K takovému dělení je použita metoda MoSCoW[2].

5.2.1 Funkční požadavky

Must Have (Aplikace musí mít)

- Aplikace umožní registrovat nové zaměstnance.

[2] MoSCow je prioritizační metoda, která definuje požadavky podle priorit: Must Have (musí mít), Should Have (mělo by mít), Could Have (bylo by dobré, kdyby mělo), Won't Have (zatím mít nebude)

- Zaměstnanci a manageři musí mít možnost se do systému přihlásit pod vlastním uživatelským účtem.
- Aplikace umožní správu zaměstnanců.
- K aplikaci lze přistupovat pod různými rolemi.
- Aplikace umožní zaměstnancům přidávat požadavky na směny.
- Aplikace umožní manažerům směny editovat, schvalovat a zamítat.
- Aplikace zobrazí zaměstnanci jeho směny.

Should Have (Aplikace by měla by mít)

- Aplikace uchovává zaměstnancovu pracovní pozici a mzdu.
- Aplikace zobrazí počet odpracovaných hodin zaměstnance za aktuální měsíc.

Could Have (Bylo by dobré, kdyby aplikace měla)

- Aplikace dokáže automaticky obsadit směny na jednotlivé dny bez práce managera.

Won't Have (Aplikace zatím mít nebude)

- Aplikace umožní komunikaci mezi zaměstnanci a managery.

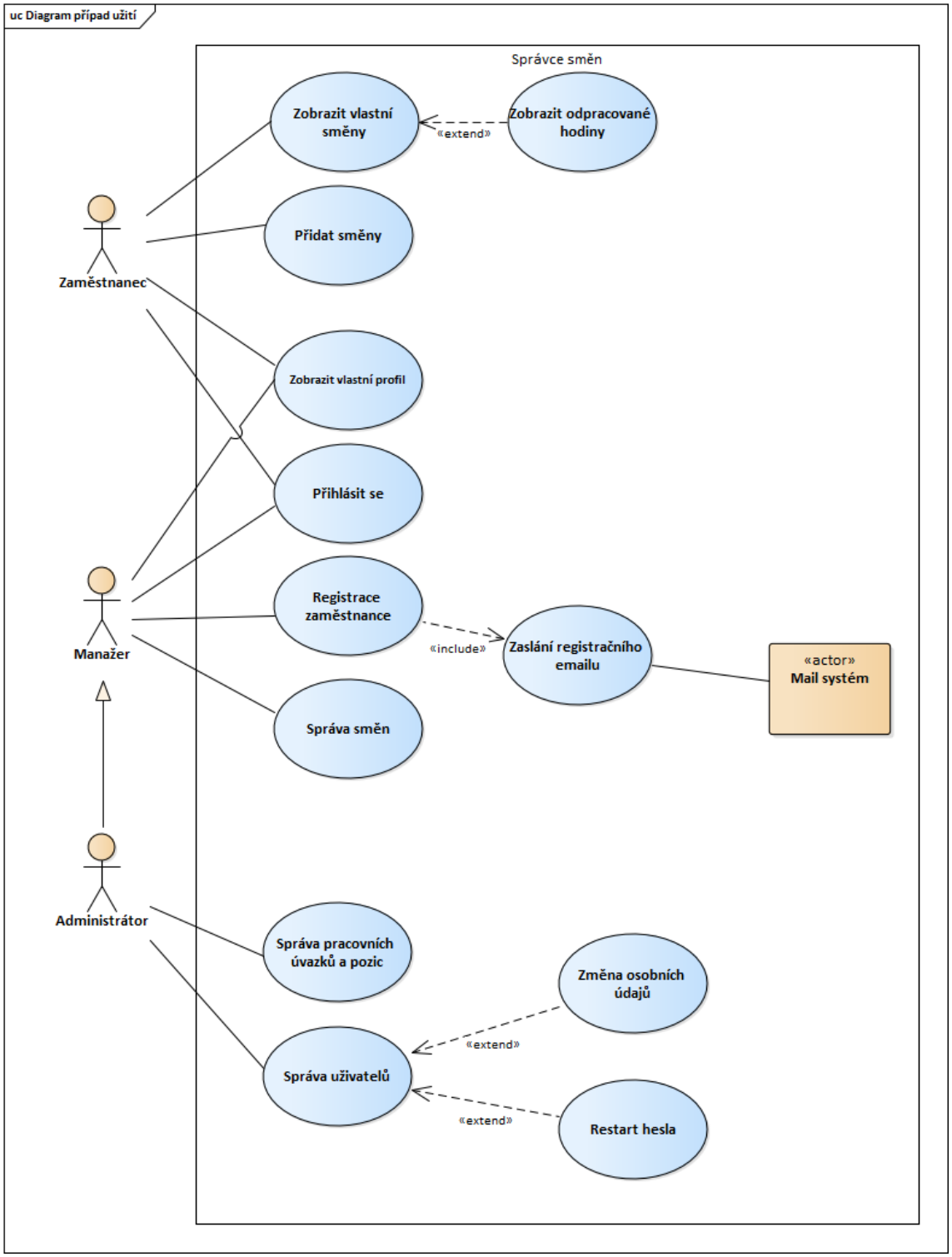
5.2.2 Nefunkční požadavky

- Aplikace bude implementována pomocí JavaScript frameworků.
- Aplikace bude využívat dokumentovou databázi.
- Aplikace bude responzivní.
- Aplikace bude dostupná na internetu.

5.3 Diagram případu užití

Diagram případu užití specifikuje funkční požadavky a prezentuje chování budoucí aplikace vůči uživatelům. Smyslem diagramu je popis, k čemu a kým bude aplikace používána. Jednotlivé případy užití jsou části celkového systému s předem definovanými scénáři, které popisují způsob interakce uživatele s aplikací při plnění

konkrétního cíle. Uživatelé zastávají vzhledem k systému určité role, které jsou v diagramu modelovány jako aktéři.



Obrázek 10 Diagram případů užití vytvářené aplikace. Zdroj: [autor]

5.4 Vybrané technologie

Po provedení rešerše vzniklo několik možných variant, pomocí kterých lze samotnou aplikaci zkonstruovat. V této podkapitole je vybrána vždy jedna pro danou část aplikace.

5.4.1 Frontend

Rešerše obsahovala tři frontend frameworky, které si drží první příčky v žebříčcích popularity. React i Angular jsou velmi silnými variantami, avšak pro tuto aplikaci byl zvolen framework Vue.js, který se jeví jako vhodný pro začínající programátory v JavaScript frameworcích. Podle [47] je Vue.js nejméně náročnou variantou pro naučení, což při neznalosti daných technologií hraje důležitou roli vzhledem k času, který je na zhotovení aplikace dán.

5.4.2 Backend

Pro backend byl vybrán Express, jelikož je nejpopulárnější mezi frameworky zde v práci probíraných [48][49]. Popularita nemusí být samozřejmě při výběru zásadní, ale ve většině případů má populárnější řešení více návodů a větší komunitu, což je pro tento případ důležité. Je to opět z důvodu časových možností, a jelikož má Express silnou základnu vývojářů, není takový problém v krátkém časovém intervalu najít jednotlivé funkcionality, které se následně implementují do vytvářené aplikace.

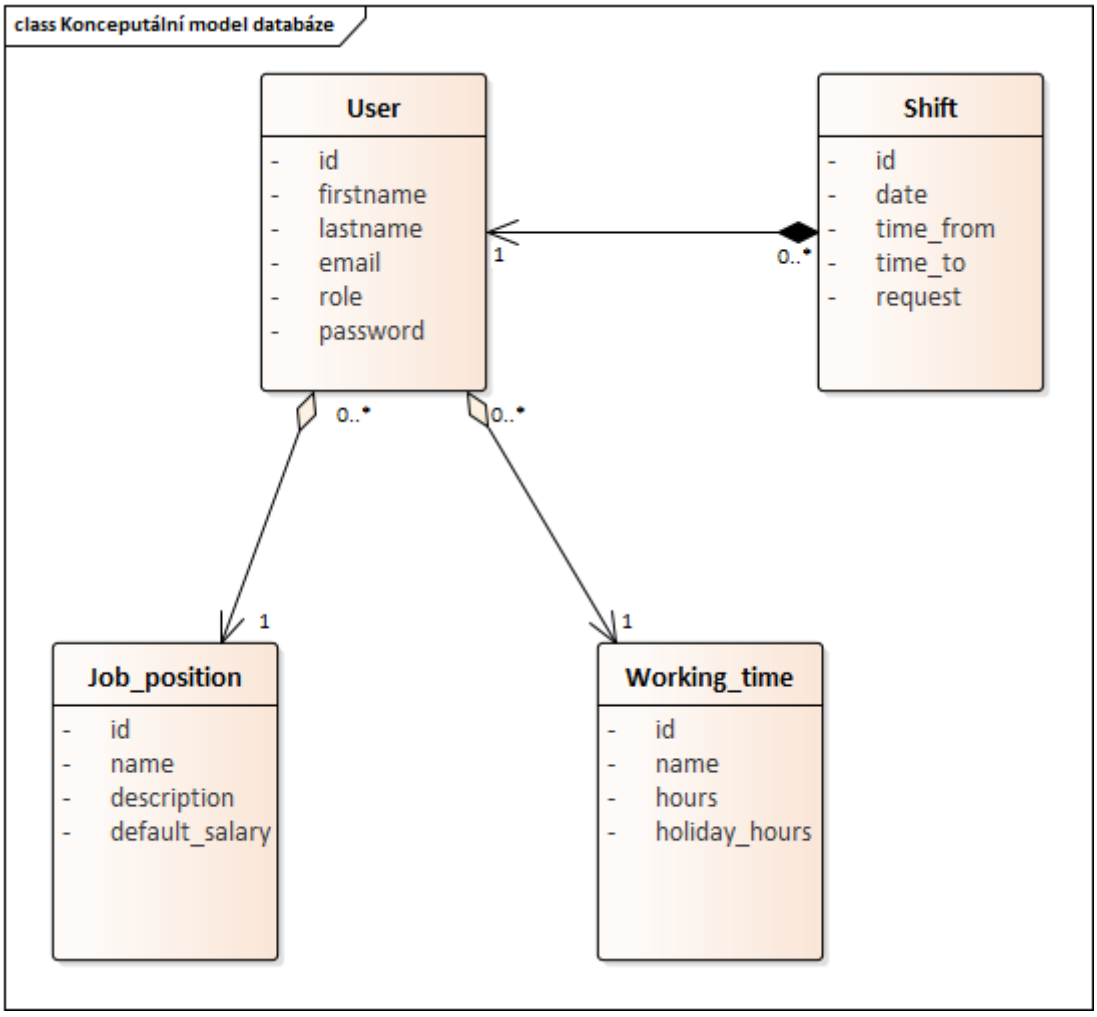
5.4.3 Databáze

Pro uložení dat byla zvolena dokumentová databáze MongoDB. Hlavním důvodem byla volnost v návrhu databáze, jelikož dokumentová databáze nemusí mít předem dané schéma. Pro vytvářenou aplikaci, která má mnoho možných budoucích vylepšení se tak tato databáze dokonale hodí. Dalším důvodem bylo také snadné a funkční propojení s backendem aplikace pomocí knihovny `mongoose`.

5.5 Návrh databáze

I přes to, že zvolená dokumentová databáze nemusí mít dané schéma a nemusejí být definovány relace mezi jednotlivými entitami, je důležité si předem vytvořit alespoň konceptuální model databáze, aby nevznikala poté přílišná nadbytečnost dat a bylo snazší se v nich orientovat. Z tohoto modelu budou následně vycházet kolekce[3], které budou vytvořeny v MongoDB.

Konceptuální model zobrazuje a popisuje databázi a její jednotlivé entity a vztahy mezi nimi. Jedná se o databázové schéma, které není nijak implementačně závislé, z toho důvodu ho lze použít i pro modelování dokumentové databáze.



Obrázek 11 Konceptuální model databáze. Zdroj: [autor]

[3] Kolekce jsou jednotlivé části databáze, které sjednocují data stejného druhu (lze je přirovnat k tabulkám v relačních databázích).

5.6 Komponenty aplikace

Je zvykem, že moderní aplikace se skládají z několika oddělených komponent s různým druhem funkcionalit, které na sobě nejsou přímo závislé. Při návrhu aplikace je tak třeba rozdělit funkcionality do odlišných komponent, které poté budou podle priorit požadavků implementovány.

5.6.1 Uživatelé

Přihlášení uživatele

Tato komponenta má na starost ověřování uživatelů pomocí přihlašovacích údajů. Jelikož je systém interní, tak bez této komponenty aplikace nemůže fungovat. Vyžaduje totiž přihlášení před jakoukoliv další aktivitou uživatele, a proto při spuštění webové aplikace bude zobrazována jako první. Po uživatelovi bude žádáno zadání emailu a hesla, pomocí nichž aplikace ověří přístup a při správnosti údajů vpustí uživatele do systému.

Profil uživatele

Profil uživatele bude obsahovat osobní údaje o uživateli, jeho jméno, příjmení, email, plat, pracovní pozici a úvazek. Slouží tedy spíše informativně a poskytuje informaci přihlášenému uživateli o jeho účtu.

Registrace nového uživatele

Uživatel se do systému nemůže zaregistrovat sám, ale musí ho registrovat administrátor, případně manažer. Je to z toho důvodu, že systém je interní a je třeba mít pod kontrolou správu uživatelů (v tomto případě hlavně zaměstnanců), kteří v podniku aplikaci využívají. Při registraci jsou zadány uživatelovy osobní údaje a informace o jeho pracovním úvazku a pozici. Heslo uživatelovi generuje aplikace sama a posílá mu jej spolu s přihlašovacími údaji a potvrzením o registraci na email.

Změna hesla uživatele

Díky této komponentě bude uživatel schopný změnit své heslo. Zadá své stávající a nové heslo a změní jej. Tato část je zde hlavně kvůli tomu, že uživatel své základní heslo dostává přiděleno při registraci, a proto mu je pro jeho bezpečí umožněno ho změnit.

Správa uživatelů

Správu uživatelů může provádět pouze administrátor. Komponenta obsahuje seznam uživatelů a možnost upravovat jejich osobní údaje, případně je smazat z databáze.

5.6.2 Směny

Požadavky na směny

Tuto komponentu budou využívat pouze zaměstnanci, kterým bude sloužit pro zadání jejich požadavků na směny. Budou zde zadávat datum a čas, kdy v následujícím období chtějí pracovat.

Správa směn

Komponenta bude přístupná pouze managerům a administrátorům, kteří zde budou schvalovat požadavky od zaměstnanců. Budou zde zobrazeny vždy všechny požadavky směn na dané období dopředu s možností jejich úprav.

Potvrzené směny zaměstnance

V této sekci bude mít uživatel přístup ke svým potvrzeným směnám, které byly schváleny. Je zde také zobrazen počet hodin, které zaměstnanec za daný měsíc již odpracoval. Komponenta je spíše informativního charakteru a má za úkol sloužit zaměstnanci pro jeho přehled.

5.6.3 Pracovní úvazky a pozice

Díky těmto dvěma komponentám bude administrátor schopný přidat jednotlivé pracovní úvazky a pozice daného podniku. Měli by být nastaveny a použity hned při zavádění aplikace, ale při růstu podniku a zavádění nových pracovních pozic budou moci býti použity i za provozu aplikace a rozšířit tak stávající.

6 Realizace výsledné aplikace

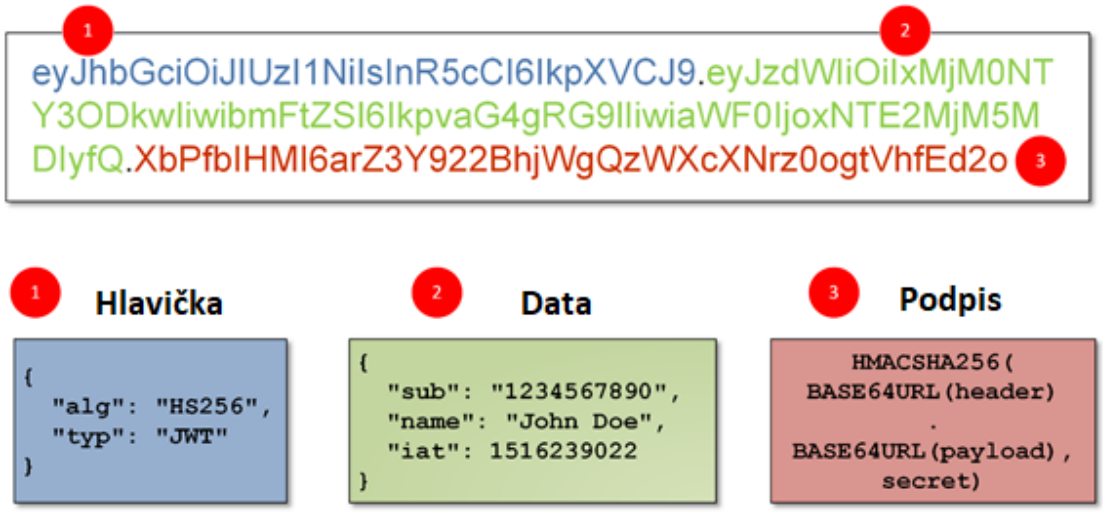
Po definování požadavků, databázové struktury a jednotlivých komponent byly vybrány technologie pro implementaci, pomocí nichž byla aplikace vytvořena. Při zhotovování byly řešeny dílčí problémy, které jsou v této kapitole blíže přiblíženy.

6.1 Zabezpečení

Důležitou součástí při implementaci aplikace je bezpochyby zabezpečení, které se stará o integritu dat uživatelů. Je třeba zajistit, aby měl k datům přístup pouze ověřený uživatel, který se prokáže svými přihlašovacími údaji, a aby jeho údaje nemohly být zneužity.

6.1.1 JSON Web Token (JWT)

JWT poskytuje možnost zabezpečené výměny informací mezi dvěma stranami. Jedná se o JSON objekt skládající se z hlavičky (header), dat (payload) a podpisu (signature) podle specifikace RFC 7519. JWT se tak stará o ověření dat a o to, jestli nebyla cestou změněna.



Obrázek 12 Ukázka JSON Web Tokenu. Zdroj: [50]

Aplikace pomocí tohoto tokenu uloženého v hlavičce HTTP požadavku ověřuje uživatele, kontroluje roli, kterou má uživatel přidělenou a díky tomu mu poskytuje ty data, na které má oprávnění. Následující ukázka ukládá data do zmiňovaného `payloadu` a generuje token, který je odesílán jako odpověď klientovi (frontendu).

```
const payload = {
      _id: user._id,
      name: user.name,
      lastname: user.lastname,
      email: user.email,
      role: user.role
}
jwt.sign(payload, key, {
}, (err, token) => {
      res.status(200).json({
            token: `Bearer ${token}`,
      });
});
```

Zdrojový kód 11 Vytvoření JSON Web Tokenu. Zdroj: [autor]

Klient následně ukládá token do HTTP hlavičky a `localStorage`, což je vidět níže v ukázce. Díky této části kódu je uživatel ověřen a je mu umožněn přístup k zabezpečeným datům a privátním cestám aplikace.

```
store.subscribe((mutation) =>{
  switch(mutation.type){
      case 'SET_TOKEN':
          if(mutation.payload){
             axios.defaults.headers.common['Authorization'] =
mutation.payload;
             localStorage.setItem('token', mutation.payload)
          }else{
             axios.defaults.headers.common['Authorization'] =
null;
                localStorage.removeItem('token')
          }
          break;
  }
})
```

Zdrojový kód 12 Uložení tokenu do HTTP hlavičky a localStorage. Zdroj: [autor]

6.1.2 Ochrana hesel

O ochranu hesel uživatelů se stará knihovna `bcryptjs`, která umožňuje vytvořit hash hesla. Navíc umožňuje i tzv. `salted hashing`, které k danému heslu přidá ještě několik náhodných znaků (sůl), a pak heslo zahashuje. V následující ukázce je reprezentováno ukládání hesla při registraci nového uživatele v aplikaci.

```
bcrypt.genSalt(10, (err, salt) => {
    bcrypt.hash(newUser.password, salt, (err, hash) => {
        if (err) throw err;
        newUser.password = hash;
        newUser.save().then(user => {
            return res.status(201).json({
                    success: true,
                    msg: "The user was registered!!"
                    });
        });
    });
});
```

Zdrojový kód 13 Ochrana hesla pomocí knihovny bcryptjs. Zdroj: [autor]

6.1.3 Routování

O zabezpečení cest se stará oficiální router pro Vue, který podporuje zabezpečení jednotlivých cest podle volitelných podmínek. Jelikož v aplikaci vystupují uživatelé ve více rolích, tak jedním z důležitých úkolů bylo povolit přístup k jednotlivým pohledům podle role. Ukázka níže předkládá, jak je tento problém v aplikaci zpracován.

```
{
  path: '/mngrequests',
  name: 'Manage requests',
  meta:{title: 'Manage shift requests', requiresManager: true}
,
  component: () => import('../views/ShiftRequestsMng.vue'),
},

if(to.matched.some(record => record.meta.requiresManager)){
    if(store.getters.isLoggedIn &&
(store.getters.isAdmin || store.getters.isManager)){
      next();
    }else{
    next('/login')
  }else{
    next()
  }
```

Zdrojový kód 14 Zabezpečení cest pomocí Vue Routeru. Zdroj: [autor]

6.2 Emailová komunikace

K odesílání emailů byla použita knihovna `@sendgrid/mail`, která umožňuje odesílat emaily ze zvolené emailové adresy. Pro použití této knihovny je však nutné vytvořit si účet na webu `sendgrid.com`, který je třeba k získání API klíče. Pomocí klíče je následně umožněno odesílání emailů z aplikace. Následující kód ukazuje už praktické použití knihovny ve vytvářené aplikaci. Pro účely práce byl zvolen Gmail, ale při produkčním použití aplikace lze nastavit odesílání z emailového serveru s vlastní doménou.

```
const sgMail = require('@sendgrid/mail')
sgMail.setApiKey(process.env.SENDGRID_API_KEY)

async function sendConfirmMail(
name, lastname, email, password) {
    sgMail.send({
        from: 'Shift Manager <mailershiftmanager@gmail.com>',
        to: email,
        subject: "Registration in the Shift Manager",
        html:
      `You have been successfully registered to the
      <b>Shift Manager</b>
       <br>Name: ${name}
       <br>Lastname: ${lastname}
       <br>Your email: ${email}
       <br>Your PIN: ${password}`,
    }).then(() => {
        console.log("E-mail successfully sent!")
    }).catch((error) => {
        console.log(error);
    })
}
```

Zdrojový kód 15 Odesílání potvrzovacích emailů. Zdroj: [autor]

6.3 Uživatelské rozhraní

Nedílnou součástí při tvorbě aplikace není pouze její logická a datová část, ale i část prezentační, která slouží k interakci s uživatelem. Důležitým faktorem při vývoji uživatelského rozhraní je tedy vytvoření intuitivního prostředí, které umožní uživateli jednoduchou správu aplikace.

6.3.1 BootstrapVue

Bootstrap je HTML, CSS a JavaScript framework pro vývoj responzivních webových stránek. Obsahuje šablony založené na HTML a CSS, které slouží k úpravě formulářů, tabulek, navigačních panelů a mnoho dalších komponent. Velkou výhodou tohoto frameworku je zmiňovaná responzivita daných komponent, jelikož ulehčuje, a hlavně urychluje vývoj uživatelského rozhraní.

BootstrapVue je přímou nadstavbou pro vývoj Vue.js aplikací, poskytuje tak kompatibilní komponenty s jejich rozhraním.

Vyvíjená aplikace využívá tento framework ve většině případů, a to z toho důvodu, že komponenty není potřeba složitě stylovat, jelikož už jsou předpřipravené. Následující ukázka zobrazuje část navigačního panelu a ukazuje jednoduchost vývoje s frameworkem, jelikož není nutnost dále nijak stylovat.

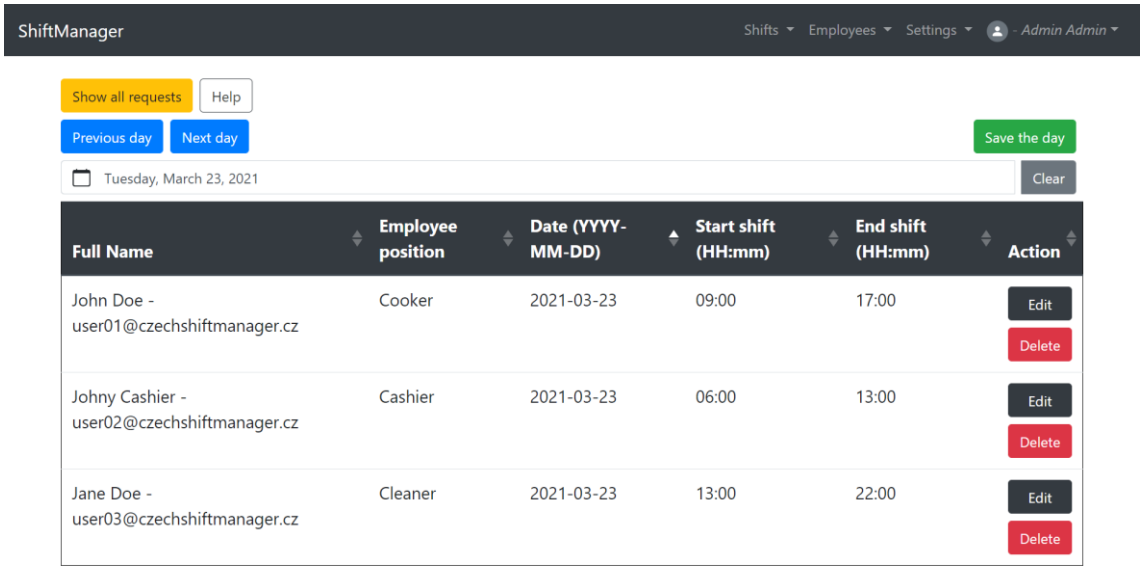
```
<b-navbar toggleable="lg" type="dark" variant="dark">
      <b-navbar-brand to="/">ShiftManager</b-navbar-brand>
      <b-navbar-toggle target="nav-collapse" v-
if="isLoggedIn"></b-navbar-toggle>

      <b-collapse id="nav-collapse" is-nav v-if="isLoggedIn">
       <b-navbar-nav class="ml-auto">
         <b-nav-item-dropdown right v-
if="isLoggedIn && (isAdmin || isManager)">
           <template v-slot:button-content>
             Employees
           </template>
           <b-dropdown-item to="/employees" exact exact-
active-class="active">All employees</b-dropdown-item>
           <b-dropdown-item to="/register" exact exact-active-
class="active">Register emploeyee</b-dropdown-item>
         </b-nav-item-dropdown>
       </b-navbar-nav>
      </b-collapse>
</b-navbar>
```

Zdrojový kód 16 Navigační panel pomocí BootstrapVue. Zdroj: [autor]

6.3.2 Vzhled aplikace

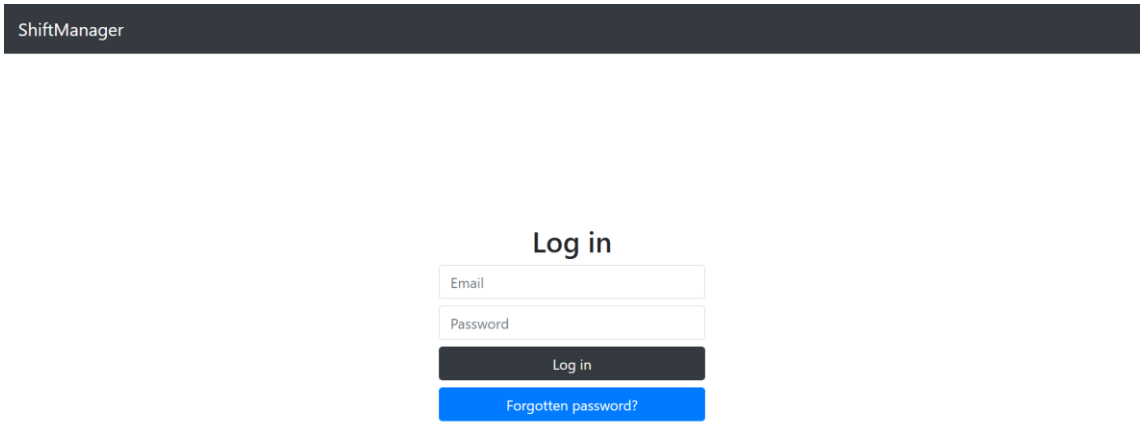
Pro aplikaci bych zvolen jednotný design mající za cíl být intuitivní a jednoduchý. Využívá komponent ze zmiňovaného BootstrapVue a základem většiny pohledů je tabulka, do které jsou vkládány data z jednotlivých částí.

Následující obrázek zobrazuje rozhraní pro správu požadavků od zaměstnanců. U každého zaměstnance je možnost upravit čas směny, nebo ji smazat. Po editaci je možné jednotlivý den uložit. Pohled také obsahuje tlačítka pro přesun mezi jednotlivými dny.



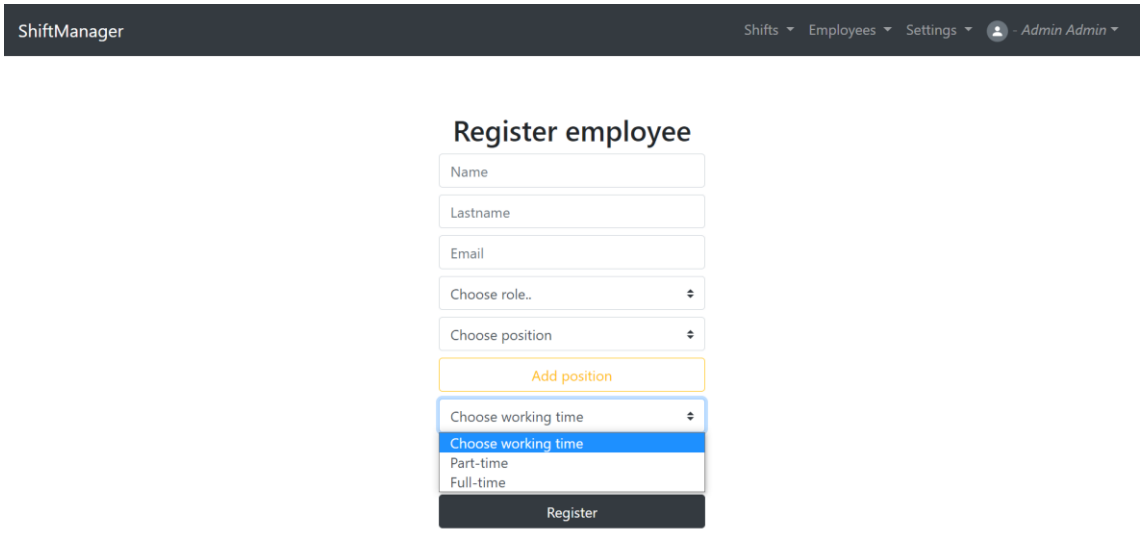
Obrázek 13 Vzhled rozhraní pro správu směn. Zdroj: [autor]

Z hlediska uživatelského rozhraní sloužícímu k přihlašování uživatele do aplikace byl vytvořen jednoduchý formulář, který vyžaduje uživatelův email a heslo. Zároveň je to také jediný formulář a pohled, ke kterému má přístup neověřený uživatel. Obrázek níže zobrazuje rozhraní, které bylo pro autentizaci vytvořeno.



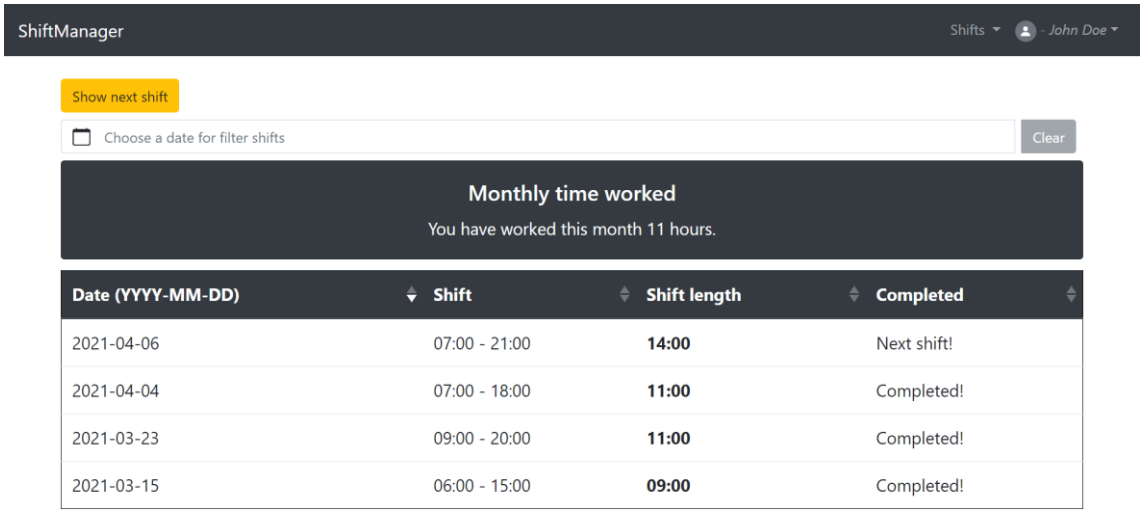
Obrázek 14 Vzhled přihlašovacího formuláře aplikace. Zdroj: [autor]

Pro registraci uživatelů byl vytvořen již komplexnější formulář, do kterého jsou poskytovány uložená data o pracovních pozicích a úvazcích z databáze. Ty jsou poté zobrazeny ve výběrových polích a slouží ke specifikaci zaměstnance a jeho pracovního postavení v daném podniku. Na obrázku níže je zobrazeno právě uživatelské rozhraní, které aplikace pro registraci používá.



Obrázek 15 Vzhled registračního formuláře aplikace. Zdroj: [autor]

Nedílnou součástí je také samotné zobrazování směn konkrétním zaměstnancům, kteří k aplikaci přistupují. Jak již bylo v návrhu zmíněno, tak je tato komponenta převážně informativního charakteru, avšak je nesmírně důležitá, jelikož je na ní postaven základní koncept aplikace, a to předat informace o směnách zaměstnanci. Na obrázku níže je zobrazeno uživatelské rozhraní, které je poskytováno samotným zaměstnancům pro jejich přehled o směnách.



Obrázek 16 Vzhled uživatelského rozhraní směn zaměstnance. Zdroj: [autor]

6.4 Nasazení aplikace

Po dokončení aplikace bylo třeba vyřešit nasazení aplikace do produkčního prostředí, ze kterého bude dostupná pro ostatní uživatele. Jelikož je aplikace interního charakteru, tak by měla být poskytována jednotlivým zákazníkům s určitým nastavením pro jejich podnik. Pro účely práce byla však aplikace nasazena pouze s testovacími daty na neplacenou cloudovou platformu Heroku a databáze na cloudovou službu MongoDB Atlas.

6.4.1 Heroku

Heroku je cloudová platforma sloužící k nasazení aplikací do produkčního prostředí. Podporuje řadu programovacích jazyků, jako jsou Java, Ruby, Node.js, PHP, Python či Scala [51].

V první řadě bylo zapotřebí nastavit aplikaci pro produkční prostředí, což znamená nastavit koncové body aplikace, jelikož ve vývojovém prostředí byly nastaveny na `localhost`. Poté byl proveden `build` frontendu aplikace, která byla následně nahrána do repozitáře na GitHub. Následně bylo třeba vytvořit uživatelský účet na Heroku a propojit ho s daným repozitářem v GitHubu. Po nahrání zdrojových kódu z GitHubu na server Heroku byla aplikace automaticky zkompilována a spuštěna na poskytnuté url adrese.

6.4.2 MongoDB Atlas

MongoDB Atlas je databázová služba, která poskytuje cloudovou verzi databáze Mongo s možností zvolit si vlastního poskytovatele cloudových služeb (AWS, Azure, nebo Google Cloud) [52].

Před nasazením databáze bylo třeba opět vytvořit uživatelský účet, zvolit zmiňovaného poskytovatele a server, který bude databázi hostovat. Po vytvoření tzv. `clusteru`, bylo třeba vytvořit databázi a získat `uri`, které bylo vloženo do kódu aplikace, čímž se k ní připojila.

7 Závěr

Hlavním cílem této bakalářské práce bylo vytvořit webovou aplikaci pro plánování směn dle zadaných požadavků od zaměstnanců. Před samotným vývojem bylo třeba si definovat principy webových aplikací, novodobé technologie a JavaScript frameworky, které byly použity při vývoji.

V první části teoretické práce byly stručně popsány principy webové aplikace a technologie, které se při vývoji využívají a díky nimž lze aplikaci definovat jako webovou. Dále byly představeny technologie HTML, CSS a DOM, které se starají o uspořádání a strukturu dokumentu, byl popsán protokol SOAP společně s REST architekturou, které slouží pro přenos dat pomocí základních HTTP požadavků. Práce také vysvětluje rozdíl mezi jednostránkovou a vícestránkovou aplikací a popisuje novodobý trend takzvaných progresivních webových aplikací.

V druhé části je představen programovací jazyk JavaScript, rozhraní Node.js a frameworky pro backend a frontend aplikace. Práce následně popisuje i samotné možnosti pro uchovávání dat a uvádí do problému různého typu databází.

V praktické části byla na základě požadavků navržena webová aplikace, která byla následně implementována ve vybraných technologiích. První verze této aplikace je poskytována se základním nastavením pro produkční prostředí. To znamená, že je vytvořen základní uživatelský účet administrátora, pomocí něhož je umožněno definovat základní nastavení pro jednotlivý podnik, a to přidat pracovní pozice, na kterých mohou zaměstnanci pracovat, vytvořit pracovní úvazky a v neposlední řadě zaregistrovat do systému zaměstnance se zmiňovanými specifikacemi. Po tomto kroku je umožněn zaměstnancům vstup do systému, kde můžou libovolně žádat o směny na následující období, své požadavky libovolně upravovat, zobrazit odpracované a nadcházející směny zároveň s odpracovanými hodinami v daný měsíc. Na druhé straně manageři a administrátoři mají možnost tyto požadavky spravovat a kontrolovat.

Pro praktické využití by aplikace potřebovala projít ještě dalším vývojem a testováním, které by zajistilo maximální spolehlivost aplikace. Mezi možná vylepšení patří automatické generování směn, větší zabezpečení, komunikace mezi zaměstnancem a managerem, či vylepšené a přehlednější uživatelské rozhraní, které by poskytlo snazší správu zaměstnanců a směn.

8 Seznam použité literatury

[1] Dreamweaver User Guide. 301 Moved Permanently [online]. Copyright © 2020 [cit. 12.10.2020]. Dostupné z: https://helpx.adobe.com/cz/dreamweaver/user-guide.html/cz/dreamweaver/using/web-applications.ug.html

[2] Webová aplikace (Web Application) - ManagementMania.com. [online]. Copyright © 2011 [cit. 15.10.2020]. Dostupné z: https://managementmania.com/cs/webova-aplikace-web-application

[3] What Percentage of Internet Traffic Is Mobile? [Feb 2021]. Oberlo | Where Self Made is Made [online]. Copyright © 2015 [cit. 14.03.2021]. Dostupné z: https://www.oberlo.com/statistics/mobile-internet-traffic

[4] PROCHÁZKA, David. CSS a XHTML: tvorba dokonalých WWW stránek krok za krokem. 2., aktualiz. vyd. Praha: Grada, 2011. Průvodce (Grada). ISBN 9788024738970.

[5] Layout (rozložení stránky) - Český HTML 5 manuál. itnetwork.cz - Ajťácká sociální síť a materiálová základna pro C#, Java, PHP, HTML, CSS, JavaScript a další. [online]. Copyright © 2021 [cit. 24.01.2021]. Dostupné z: https://www.itnetwork.cz/html-css/html5/html-manual/html-css-html-manual-rozlozeni/html-layout-rozlozeni-stranky-cesky-manual/

[6] CSS basics - Learn web development | MDN. [online]. Copyright © 2021 [cit. 24.01.2021]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/CSS_basics

[7] Introduction to the DOM - Web APIs | MDN. [online]. Copyright © 2005 [cit. 19.10.2020]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction

[8] JavaScript HTML DOM. W3Schools Online Web Tutorials [online]. Copyright © 2020 [cit. 15.10.2020]. Dostupné z: https://www.w3schools.com/js/js_htmldom.asp

[9] HOLDENER, Anthony T. Ajax: the definitive guide. Sebastopol: O'Reilly, 2008. ISBN 978-0-596-52838-6.

[10] AJAX Introduction. W3Schools Online Web Tutorials [online]. Copyright © 2021 [cit. 09.04.2021]. Dostupné z: https://www.w3schools.com/js/js_ajax_intro.asp

[11] SOAP Version 1.2 Part 1: Messaging Framework (Second Edition). World Wide Web Consortium (W3C) [online]. Copyright © 2007 [cit. 30.10.2020]. Dostupné z: https://www.w3.org/TR/soap12/

[12] Lesson 5: Simple Object Access Protocol - SOAP - YouTube. YouTube [online]. Copyright © 2020 [cit. 30.10.2020]. Dostupné z: https://www.youtube.com/watch?v=l2QrCqzRhWo

[13] XML Soap. W3Schools Online Web Tutorials [online]. Copyright © 2020 [cit. 30.10.2020]. Dostupné z: https://www.w3schools.com/xml/xml_soap.asp

[14] BATTLE, Robert a Edward BENSON. Bridging the semantic Web and Web 2.0 with Representational State Transfer (REST). Journal of Web Semantics. 2008, 6(1), 61-69. ISSN 15708268. Dostupné z: doi:10.1016/j.websem.2007.11.002

[15] Single Page Application vs Multiple Page Application - [Ultimate Guide] | Intelegain. Custom Software & App Development Company US & India | Intelegain [online]. Copyright © 2019 [cit. 15.10.2020]. Dostupné z: https://www.intelegain.com/single-page-applications-vs-multiple-page-applications/

[16] SCOTT, Emmit A. SPA design and architecture: understanding single-page web applications. Shelter Island, NY: Manning, 2016. ISBN 978-1617292439

[17] Single-page App vs. Multi-page App: Pros, Cons, and Which is Better?. Lvivity – Custom Software Development Company [online]. Copyright © 2013 [cit. 15.10.2020]. Dostupné z: https://lvivity.com/single-page-app-vs-multi-page-app

[18] Introduction to Progressive Web Apps | The PWA Book. Divante - eCommerce technology company [online]. [cit. 9.11.2020]. Dostupné z: https://divante.com/pwabook/chapter/01-introduction-to-pwa-technology.html#what-are-progressive-web-apps

[19] Employers Most In-Demand Programming Languages 2020 | by Women Who Code | Medium. Medium – Where good ideas find you. [online]. [cit. 22.11.2020]. Dostupné z: https://medium.com/@WomenWhoCode/employers-most-in-demand-programming-languages-2020-ca13705a73fc

[20] Most used languages among software developers globally 2020 | Statista. Statista - The Statistics Portal for Market Data, Market Research and Market Studies [online]. Copyright © 2020 [cit. 05.12.2020]. Dostupné z: https://www.statista.com/statistics/793628/worldwide-developer-survey-most-usedlanguages/

[21] HAVERBEKE, Marijn. Eloquent JavaScript: a modern introduction to programming. Third edition. San Francisco: No Starch Press, [2019]. ISBN 978-159-3279-509

[22] Node.js. [online]. [cit. 22.11.2020]. Dostupné z: https://nodejs.org/en/

[23] Lekce 1 - Úvod do Node.js. itnetwork.cz - Ajťácká sociální síť a materiálová základna pro C#, Java, PHP, HTML, CSS, JavaScript a další. [online]. Copyright © 2020 itnetwork.cz. Veškerý obsah webu [cit. 22.11.2020]. Dostupné z: https://www.itnetwork.cz/javascript/nodejs/uvod-do-nodejs

[24] Introduction — Vue.js. Vue.js [online]. [cit. 24.11.2020] Dostupné z: https://vuejs.org/v2/guide/

[25] Reactivity in Depth — Vue.js. Vue.js [online]. [cit. 24.11.2020] Dostupné z: https://vuejs.org/v2/guide/reactivity.html

[26] Components Basics — Vue.js. Vue.js [online]. [cit. 24.11.2020] Dostupné z: https://vuejs.org/v2/guide/components.html

[27] The Vue Instance — Vue.js. Vue.js [online]. [cit. 05.12.2020]. Dostupné z: https://vuejs.org/v2/guide/instance.html

[28] Vue Instance Lifecycle & Hooks [online]. [cit. 14.03.2021]. Dostupné z: https://codingexplained.com/coding/front-end/vue-js/vue-instance-lifecycle-hooks

[29] Tutorial: Intro to React – React. React – A JavaScript library for building user interfaces [online]. Copyright © 2020 Facebook Inc. [cit. 05.12.2020]. Dostupné z: https://reactjs.org/tutorial/tutorial.html

[30] 17 Powerful React Libraries to Try in 2020. Software Development Company in USA | Simform [online]. Copyright © 2020 Simform. All Rights Reserved. [cit. 05.12.2020]. Dostupné z: https://www.simform.com/react-libraries/

[31] Introducing JSX – React. React – A JavaScript library for building user interfaces [online]. Copyright © 2020 Facebook Inc. [cit. 05.12.2020]. Dostupné z: https://reactjs.org/docs/introducing-jsx.html

[32] Components and Props – React. React – A JavaScript library for building user interfaces [online]. Copyright © 2020 Facebook Inc. [cit. 05.12.2020]. Dostupné z: https://reactjs.org/docs/components-and-props.html

[33] Angular. Angular [online]. [cit. 05.12.2020]. Dostupné z: https://angular.io/docs

[34] Lekce 1 - Úvod do Angular frameworku. itnetwork.cz - Ajťácká sociální síť a materiálová základna pro C#, Java, PHP, HTML, CSS, JavaScript a další. [online]. Copyright © 2020 itnetwork.cz. Veškerý obsah webu [cit. 05.12.2020]. Dostupné z: https://www.itnetwork.cz/javascript/angular/zaklady/uvod-do-angular-frameworku

[35] Angular. Webdesign, webové aplikace, grafika [online]. [cit. 05.12.2020]. Dostupné z: http://visionplus.cz/tutorialy/angular.php

[36] Express - Node.js web application framework. Express - Node.js web application framework [online]. Copyright © 2017 StrongLoop, IBM, and other expressjs.com contributors. [cit. 05.12.2020]. Dostupné z: https://expressjs.com/

[37] Node.js - Express Framework - Tutorialspoint. Apache HTTP Server Test Page powered by CentOS [online]. Copyright © Copyright 2020. All Rights Reserved. [cit. 05.12.2020]. Dostupné z: https://www.tutorialspoint.com/nodejs/nodejs_express_framework.htm

[38] Koa - next generation web framework for node.js. Koa - next generation web framework for node.js [online]. Dostupné z: https://koajs.com/

[39] Simple REST Server using KOA 2.0. Koa is a web framework for Node.js… | by Tejas Rao | Medium. Medium – Where good ideas find you. [online]. [cit. 22.01.2020]. Dostupné z: https://medium.com/@tejasrr19/simple-rest-server-using-koa-2-0-fd59cf81bd05

[40] BRETT, John. Getting Started with Hapi.js. Birmingham, England: Packt Publishing Limited, 2016. ISBN 978-1-78588-818-2

[41] Getting Started - hapi.dev. hapi.dev - The simple, secure framework developers trust [online]. Copyright © 2021 [cit. 22.01.2021]. Dostupné z: https://hapi.dev/tutorials/gettingstarted/?lang=en_US

[42] What Is MongoDB? | MongoDB. The most popular database for modern apps | MongoDB [online]. Copyright © 2021 MongoDB, Inc. [cit. 22.01.2021]. Dostupné z: https://www.mongodb.com/what-is-mongodb

[43] PostgreSQL: About. PostgreSQL: The world's most advanced open source database [online]. Copyright © 2021 [cit. 22.01.2021]. Dostupné z: https://www.postgresql.org/about/

[44] Firebase: krátké seznámení - Zdroják. Zdroják - o tvorbě webových stránek a aplikací [online]. Copyright © 2021 [cit. 23.01.2021]. Dostupné z: https://www.zdrojak.cz/clanky/firebase-kratke-seznameni/

[45] Choose a Database: Cloud Firestore or Realtime Database | Firebase. Firebase [online]. Copyright © 2021 [cit. 14.03.2021]. Dostupné z: https://firebase.google.com/docs/database/rtdb-vs-firestore

[46] Add backend logic to real-time data with Firebase and Google App Engine [online]. [cit. 14.03.2021]. Dostupné z: https://developers.googleblog.com/2015/11/add-backend-logic-to-real-time-data.html

[47] Angular vs React vs Vue: Which Framework to Choose in 2021. CodeinWP - A Hub for WordPress Freelancers, Bloggers & Creators [online]. Copyright © 2021 [cit. 26.01.2021]. Dostupné z: https://www.codeinwp.com/blog/angular-vs-vue-vs-react/

[48] Choosing the right Node.js Framework: Express, Koa, or Hapi?. For mission-critical Node.js applications - NodeSource [online]. Copyright © 2021 [cit. 27.01.2021]. Dostupné z: https://nodesource.com/blog/Express-Koa-Hapi

[49] Hapi vs Koa vs Express | Section. Edge as a Service (EaaS) - Flexible Edge Compute Platform | Section [online]. Copyright © 2020 Section [cit. 27.01.2021]. Dostupné z: https://www.section.io/engineering-education/hapi-vs-koa-vs-express/

[50] JWT (JSON Web Token) (in)security [online]. Copyright © 2021 [cit. 15.03.2021]. Dostupné z: https://research.securitum.com/jwt-json-web-token-security/

[51] Heroku | VIP Trust s.r.o.. Úvodní strana | VIP Trust s.r.o. [online]. Copyright © 2021 VIP Trust s.r.o. [cit. 15.03.2021]. Dostupné z: https://viptrust.com/technologie/ostatni/heroku

[52] MongoDB Atlas — MongoDB Atlas. MongoDB Atlas — MongoDB Atlas [online]. Copyright © MongoDB, Inc 2008 [cit. 15.03.2021]. Dostupné z: https://docs.atlas.mongodb.com/

Univerzita Hradec Králové
Fakulta informatiky a managementu

Zadání bakalářské práce

Autor: **Jiří Pipek**

Studium: I1800217

Studijní program: B1802 Aplikovaná informatika

Studijní obor: Aplikovaná informatika

Název bakalářské práce: **Implementace webové aplikace pro plánování směn**

Název bakalářské práce AJ: Implementation of Shift Planning Web Application

Cíl, metody, literatura, předpoklady:

Cílem bakalářské práce je představení webových aplikací a technologií pro jejich vývoj, ze kterých bude vycházet funkční webová aplikace pro plánování směn.

Osnova:

- Úvod
- Webové aplikace
- Rešerše dostupných technologií
- Návrh a analýza webové aplikace
- Realizace výsledné aplikace
- Závěr

Garantující pracoviště: Katedra informatiky a kvantitativních metod, Fakulta informatiky a managementu

Vedoucí práce: Ing. Michal Macinka

Datum zadání závěrečné práce: 14.1.2018