VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA PODNIKATELSKÁ
FACULTY OF BUSINESS AND MANAGEMENT

ÚSTAV INFORMATIKY
INSTITUTE OF INFORMATICS

VÝVOJ WEBOVÉHO FAKTURAČNÍHO NÁSTROJE VE FIREMNÍM PROSTŘEDÍ

DEVELOPMENT OF A WEB INVOICING TOOL IN A CORPORATE ENVIRONMENT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Tomáš Bohdálek**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Petr Dydowicz, Ph.D.**
SUPERVISOR

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA PODNIKATELSKÁ

Zadání bakalářské práce

Ústav:	Ústav informatiky
Student:	**Tomáš Bohdálek**
Studijní program:	Systémové inženýrství a informatika
Studijní obor:	Manažerská informatika
Vedoucí práce:	**Ing. Petr Dydowicz, Ph.D.**
Akademický rok:	2020/21

Ředitel ústavu Vám v souladu se zákonem č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů a se Studijním a zkušebním řádem VUT v Brně zadává bakalářskou práci s názvem:

Vývoj webového fakturačního nástroje ve firemním prostředí

Charakteristika problematiky úkolu:

Úvod
Vymezení problému a cíle práce
Teoretická východiska práce
Analýza problému a současné situace
Vlastní návrh řešení, přínos práce
Závěr
Seznam použité literatury

Cíle, kterých má být dosaženo:

Cílem této bakalářské práce je vytvořit pro společnost Proof & Reason, s.r.o. frontend webové aplikace, která bude sloužit jako firemní nástroj pro fakturaci práce klientům a interních lidí. Důležitou součástí aplikace bude prezentace dat pro monitoring těchto procesů a napojení aplikace na již připravené REST API.

Základní literární prameny:

BASL, J. a R. BLAŽÍČEK. Podnikové informační systémy. Podnik v informační společnosti. Praha: Grada, 2008. 283 s. ISBN 978-80-247-2279-5.

MOLNÁR, Z. Automatizované informační systémy. Praha: Strojní fakulta ČVUT, 2000. 126 s. ISBN 80-01-02269-2.

MOLNÁR, Z. Efektivnost informačních systémů. Praha: Grada Publishing, 2000. 142 s. ISBN 80-716--410-X.

PECINOVSKÝ, R. Myslíme objektově v jazyku Java: kompletní učebnice pro začátečníky. Praha: Grada, 2009. 570 s. ISBN 978-80-247-2653-3.

SODOMKA, P. a H. KLČOVÁ. Informační systémy v podnikové praxi. Brno: Computer Press, 2010. 501 s. ISBN 978-80-251-2878-7.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2020/21

V Brně dne 28.2.2021

L. S.

Mgr. Veronika Novotná, Ph.D. ředitel	doc. Ing. Vojtěch Bartoš, Ph.D. děkan

Abstrakt

Bakalářská práce se zaměřuje na návrh a vývoj webové aplikace pro podporu finančního řízení ve společnosti Proof & Reason, s.r.o., která poskytuje služby v oblasti tvorby a rozvoje digitálních produktů. Proces návrhu a vývoje aplikace vychází z požadavků společnosti a provedených analýz současného řešení. Frontend webové aplikace je implementován s využitím frameworku Next.js a knihoven React.js a Material UI. Aplikace také komunikuje s již připraveným REST API, které vytvořila společnost.

Klíčová slova

webová aplikace, frontend, SPA, JavaScript, TypeScript, Next.js, React.js, Material UI, REST API

Abstract

The bachelor thesis focuses on the design and development of a web application for financial management in the company Proof & Reason, s.r.o. The company provides services in the field of digital products. The process of design and development of the application is based on the requirements of the company and analysis of the current solution, which will be replaced by this application. The frontend of the web application is implemented using the Next.js framework and the React.js and Material UI libraries. The application also communicates with the REST API which was already created by the company.

Key words

web application, front-end, SPA, JavaScript, TypeScript, Next.js, React.js, Material UI, REST API

Bibliografická citace

BOHDÁLEK, Tomáš. *Vývoj webového fakturačního nástroje ve firemním prostředí.* Brno, 2021. Dostupné také z: https://www.vutbr.cz/studenti/zav-prace/detail/135307. Bakalářská práce. Vysoké učení technické v Brně, Fakulta podnikatelská, Ústav informatiky. Vedoucí práce Petr Dydowicz.

Čestné prohlášení

Prohlašuji, že předložená bakalářská práce je původní a zpracoval jsem ji samostatně. Prohlašuji, že citace použitých pramenů je úplná, že jsem ve své práci neporušil autorská práva (ve smyslu Zákona č. 121/2000 Sb., o právu autorském a o právech souvisejících s právem autorským).

V Brně dne 16. května 2021

……..

podpis autora

Poděkování

Rád bych poděkoval panu Ing. Petru Dydowiczovi, Ph.D. za odborné vedení a cenné rady, které mi byly poskytnuty, při zpracování této bakalářské práce. Dále bych rád poděkoval společnosti Proof & Reason, s.r.o. za poskytnutí konzultací a veškerých potřebných materiálů pro vypracování této bakalářské práce.

Obsah

ÚVOD

Finanční řízení je nedílnou součástí řízení společnosti. Aby bylo co nejvíce efektivní, je potřeba pro něj mít přesné informace, na základě kterých se pak vedení společnosti rozhoduje. V dnešní době používají společnosti nejrůznější informační systémy a software, které poskytují nástroje pro podporu finančního řízení.

Tato práce se zabývá návrhem a vývojem frontendové části aplikace, která bude sloužit jako nástroj pro podporu finančního řízení společnosti Proof & Reason, s.r.o. a vykazování odvedené práce. Téma práce bylo zvoleno z důvodu potřeby nahradit současné řešení za nové, které bude používat moderní technologie a bude nabízet požadované funkcionality.

V úvodní části práce jsou představena teoretická východiska práce, která se především zaměřují na popis využitých webových technologií a nástrojů. Další část se zabývá analýzou problému a současné situace a také je zde představena společnost Proof & Reason, s.r.o. Hlavní část práce se věnuje samotnému návrhu a vývoji aplikace na základě provedených analýz. Nejdříve je představena architektura aplikace a její zabezpečení, poté následuje návrh a implementace uživatelského rozhraní a popis hlavních modulů aplikace. Dále jsou zde uvedeny způsoby testování aplikace a představeny návrhy pro její budoucí vývoj. Na závěr je provedeno ekonomické zhodnocení nákladů na celou aplikaci včetně přínosů práce.

VYMEZENÍ PROBLÉMU A CÍLE PRÁCE

Problémem společnosti Proof & Reason, s.r.o. je nedostatečná funkcionalita vlastní webové aplikace pro podporu finančního řízení společnosti a vykazování odvedené práce na projektech. Tato webová aplikace byla před několika lety vytvořena v programovacím jazyce Ruby on Rails, ten se však ve společnosti již nepoužívá a další rozvoj aplikace by byl nejen finančně náročný, ale i technicky omezující. Hlavním problémem aplikace je nemožnost zobrazení některých metrik pro řízení projektů, vedení týmů a finanční řízení společnosti. Dále aplikace neumožňuje úpravu vytvářených výkazů přímo v aplikaci. Výkazy je tedy nutné upravovat až po jejich vyexportování z aplikace, kvůli tomu nedochází k přesné aktualizaci dat o odpracovaných a vyfakturovaných hodinách a obecně o stavu projektu v aplikaci. Společnost požaduje vytvoření nové aplikace, která bude jednoduchá, přehledná a dostupná na mobilních zařízeních.

Hlavním cílem bakalářské práce je návrh a vývoj frontendové části webové aplikace pro společnost Proof & Reason, s.r.o., která bude prezentovat data pro řízení projektů, vedení týmů a finanční řízení společnosti. Důležitou součástí aplikace bude možnost přípravy a editace výkazů práce přímo v aplikaci. Aplikace bude napojena na již připravené REST API vytvořené společností.

K dosažení tohoto cíle budou nejdříve vymezena teoretická východiska, ze kterých bude vycházeno při zpracování analýzy problému současné situace a vlastního návrhu řešení. V úvodu analýzy problému a současné situace bude představena společnost a její organizační struktura. Následně bude provedena analýza současného řešení a analýza uživatelů a infrastruktury tohoto řešení. Dále budou zjištěny požadavky společnosti na novou aplikaci a bude provedena analýza konkurenčních aplikací pro zjištění možných alternativ k vlastnímu řešení. Na závěr analytické části bude provedeno shrnutí zjištěných skutečností v podobě SWOT analýzy. Následně bude na základě výše zmíněných analýz navrhnuta a implementována aplikace za využití programovacího jazyka JavaScript, frameworku Next.js a knihoven React.js a Material UI.

1. TEORETICKÁ VÝCHODISKA PRÁCE

V této kapitole budou popsány základní pojmy, se kterými bude pracováno v dalších částech práce. Jedná se především o pojmy související s webovými aplikacemi, technologiemi a nástroji používanými pro tvorbu moderních webových aplikací.

1.1. Webová aplikace

Webová aplikace je počítačový program, který umožňuje provádět konkrétní úkoly přes internet pomocí webového prohlížeče a webových technologií. Webovou aplikaci není nutné instalovat na klientské zařízení a k aplikaci je možné v případě připojení k internetu a kompatibilního prohlížeče přistupovat odkudkoliv z libovolného zařízení [1].

1.1.1. Frontend

V oblasti tvorby webových stránek a aplikací se jako frontend označuje ta část, kterou vidí návštěvník ve svém prohlížeči, když přijde na adresu konkrétní stránky. Je to také vše, s čím návštěvník stránky interaguje. Frontend zahrnuje vzhled, strukturu stránky a rozložení jednotlivých prvků jako jsou například texty, obrázky nebo tlačítka. U frontendu by měl být kladen důraz především na uživatelskou přívětivost, přístupnost, rychlou odezvu rozhraní a správné zobrazení na různých zařízeních. Opakem frontendu je backend, ten může v pozadí zajišťovat například zpracování dat a jejich ukládání do databáze.

1.1.2. Single Page Application

Single Page Application (SPA) je webová aplikace, která při interakci s uživatelem nepotřebuje načítat zcela nové stránky ze serveru, ale pouze dynamicky přepisuje aktuální stránku. Při úvodním načtení aplikace jsou staženy všechny potřebné zdroje pro její funkčnost a aplikace se chová podobně jako nativní mobilní nebo desktopová aplikace. Díky tomuto principu může být rozhraní aplikace velmi rychlé a může nabízet dobrý uživatelský prožitek.

1.2. Webové technologie

V této kapitole budou popsány pojmy týkající se webových technologií, kterým je potřeba porozumět pro pochopení nástrojů pro tvorbu webových aplikací, které budou v této práci použity a které budou popsány v následující kapitole.

1.2.1. Protokol HTTP/HTTPS

Protokol HTTP (Hypertext Transfer Protocol) je protokolem aplikační vrstvy, který používají webové prohlížeče a webové servery k vzájemnému přenosu informací. Protokol HTTP používá mechanismus požadavků a odpovědí, kdy klient odešle požadavek z určitého odchozího portu (typicky 80) na cílový server v Internetu, který naslouchá na určitém příchozím portu a odpovídá poskytnutím odpovídajícího prostředku v odpovídajícím formátu [2, s. 587-588].

HTTP požadavek

Požadavek se skládá ze záhlaví, řádku s požadavkem, prázdného řádku a volitelného těla. Řádek s požadavkem obsahuje metodu, na kterou je daná akce provedena. V protokolu HTTP/1.1 existuje osm základní metod požadavků: OPTIONS, GET, HEAD, POST, PUT, DELETE, TRACE a CONNECT [2, s. 591].

Metoda OPTIONS slouží k získání informací o možnostech komunikace s uvedenou URL adresou. Tato metoda umožňuje klientovi určit možnosti a omezení spojené se zdrojem nebo schopnostmi serveru. Nejčastěji používaná metoda je GET, která se používá pro načtení informace identifikované pomocí URL adresy. Metoda HEAD je identická s metodou GET, jediný rozdíl je ten, že server neposílá tělo odpovědi [3].

Metoda POST se používá pro odeslání zdrojů na cílový server. Její využití nalezneme například při odesílání formuláře. Metoda PUT požaduje uložení zdrojů na server, které mohou být následně dostupné například pro metodu GET. Poslední metodou pracující se zdroji je metoda DELETE, která slouží ke smazaní zdroje [3].

HTTP odpověď

Webový server při odpovědi na požadavek vrací klientovi jednořádkový stavový kód, společně s vysvětlujícím textem, který interpretuje odpověď. Podle prvního čísla stavového kódu můžeme stavové kódy dělit na třídy [2, s. 591].

První třídu představují kódy *1xx*, které jsou označovány jako informační a dočasné. Další třída s kódy *2xx,* slouží k signalizaci úspěšně přijatého požadavku, který byl pochopen a akceptován serverem. Třetí třídu představují kódy *3xx* vyjadřující přesměrování, kdy pro splnění požadavku musí další akci provést klient. Třídu s kódy *4xx* využívá server k oznámení chyby klienta. Jedná například o situaci, kdy požadavek nemohl být zpracován kvůli chybné syntaxi. Poslední třídu tvoří kódy *5xx*, které vyjadřují chybu na straně serveru [2, s. 591-594].

Bezstavovost protokolu

Důležitou vlastností protokolu HTTP je bezstavovost. Všechny informace, které jsou potřebné k reakci na požadavek jsou obsaženy v samotném požadavku. Díky tomu nemusí klient ani server uchovávat žádné uživatelské informace [2, s. 588].

Pokud server potřebuje spravovat uživatelská data například pro přizpůsobení uživatelského rozhraní, může k tomu využít technologii zvanou Cookies [2, s. 588]. Na straně klienta se do Cookies uloží jednoznačný identifikátor, který následně klient používá v požadavku na server a ten si mezi více požadavky vytvoří souvislost.

Zabezpečení protokolu

Komunikace probíhající přes protokol HTTP není šifrovaná, proto vznikla jeho zabezpečená verze HTTPS (HTTP Secure), která používá TLS (Transport Layer Security) nebo SSL (Secure Sockets Layer) protokol pro šifrování a autentizaci [4].

Webové stránky a aplikace, které nepoužívají HTTPS, podléhají bezpečnostním varováním, chybám prohlížeče a komunikace může být odposlouchávána, proto by používání protokolu HTTPS mělo být v dnešní době samozřejmostí [4].

1.2.2. API

API (Application Programming Interface) je sada definovaných pravidel, která vysvětlují, jak počítače nebo aplikace vzájemně komunikují. Je to prostřední vrstva mezi aplikací a webovým serverem, která zpracovává přenos dat mezi systémy. API umožňuje společnostem zpřístupňovat data a funkce svých aplikací externím vývojářům třetích stran nebo interním oddělením v rámci společnosti. Vývojáři mají zpravidla dostupné zdokumentované rozhraní, které ve své aplikaci využívají a nepotřebují vědět, jakým způsobem je API implementováno [5].

REST API

REST (Representational State Transfer) je sada principů, které definují, jak by se měly používat webové standardy při vývoji webových služeb. REST API je typ aplikačního rozhraní, které typicky využívá HTTP protokol za účelem provádění standardních funkcí, jako je vytváření, čtení, aktualizace a mazání záznamů v rámci určitého prostředku. Tyto funkce jsou někdy označovány jako CRUD (Create, Read, Update, Delete) a v podstatě představují HTTP metody GET, POST, PUT a DELETE, které se používají ke zpracování požadavků na zdroje. REST API není závislé na technologii, a tak může být implementováno v různých programovacích jazycích, jako je například PHP, Node.js nebo Java [6].

1.2.3. Značkovací jazyk HTML

HTML (HyperText Markup Language) je základní technologie používaná k definování významu a struktury obsahu webové stránky. HTML se používá k určení, zda má být obsah webové stránky rozpoznán jako odstavec, seznam, nadpis, odkaz, formulář nebo jeden z mnoha dalších dostupných prvků. [7].

HTML není programovací jazyk, ale jedná se o značkovací jazyk. Skládá se z množiny prvků, které ohraničují různé části obsahu webové stránky tak, aby se zobrazovaly nebo působily určitým způsobem. Ohraničující značky mohou například z obsahu vytvořit odkaz na jinou stránku nebo vytvořit seznam položek [7].

```
<a href="#">Odkaz</a>
<ul>
    <li>První položka seznamu</li>
    <li>Druhá položka seznamu</li>
</ul>
```

Obrázek č. 1: HTML značky
(Zdroj: Vlastní zpracování)

1.2.4. Document Object Model (DOM)

Document Object Model (DOM) je rozhraní, které reprezentuje logickou strukturu HTML a XML dokumentu a standardizuje přístup a manipulaci s dokumentem napříč jednotlivými webovými prohlížeči. Dokument je v DOM reprezentován jako strom, přes který lze například pomocí JavaScriptu přistupovat k jednotlivým částem dokumentu a dále s nimi pracovat. Webový standard DOM v současné době udržuje a vyvíjí komunita WHATWG (Web Hypertext Application Technology Working Group) [8].

1.2.5. Kaskádové styly CSS

CSS (Cascading Style Sheets) neboli také kaskádové styly je deklarativní jazyk, pomocí kterého můžeme definovat vzhled HTML prvků v dokumentu, ať už se jedná o velikost písma, barvu pozadí, animace nebo rozložení jednotlivých prvků [9]. CSS také umožňuje vytvářet responzivní zobrazení, kdy pomocí tzv. Media Queries můžeme jednotlivým prvkům definovat vlastnosti na základě viditelného výřezu stránky v okně prohlížeče.

Deklarace stylů se nejčastěji provádí v souboru s příponou css nebo přímo v dokumentu, a to pomocí selektorů, do kterých se zapisují vlastnosti a k nim příslušné hodnoty. Všechny vlastnosti a jejich hodnoty, které je možné v deklaraci použít, předem definuje CSS specifikace [10], kterou spravuje CSS Working Group [9].

```
.nazev-selektoru {
    font-size: 16px; /* nastavení velikosti písma */
    background-color: white; /* nastavení barvy pozadí */
}
```

Obrázek č. 2: CSS syntaxe
(Zdroj: Vlastní zpracování)

1.2.6. Skriptovací jazyk JavaScript

JavaScript (zkráceně JS) je objektově orientovaný skriptovací jazyk, který je nejčastěji využíván na webových stránkách. Jeho primární využití je při tvorbě interaktivního uživatelského rozhraní, kdy umožňuje například manipulovat s obsahem stránky pomocí DOM nebo pracovat s nejrůznějšími API. JavaScript není omezen pouze na prostředí webového prohlížeče, ale lze jej využívat také v jiných prostředích jako je Node.js [11].

JavaScript vytvořil Brendan Eich, který byl následně zaměstnán někdejší společností Netscape. JavaScript byl poprvé vydán v roce 1995 s prohlížečem Netscape Navigator 2. V roce 1996 začala společnost Netscape spolupracovat s organizací ECMA International a vznikla standardizovaná verze JavaScriptu nazývaná ECMAScript (zkráceně ES) [11]. V roce 2015 zveřejnila ECMA International šestou hlavní verzi ES6, která přidala například syntaxe tříd a modulů. Od té doby jsou vydávány nové standardizace v ročních cyklech a v názvu je vždy uveden rok vydání [12, s. 2].

JavaScript podporuje nejen objektově orientované programování s objektovými prototypy ale také funkcionální programování, kdy je možné do proměnných ukládat funkce [11].

1.2.7. TypeScript

TypeScript (zkráceně TS) je open-source jazyk založený na JavaScriptu, který vyvíjí společnost Microsoft. TypeScript primárně přidává statické typování, pomocí kterého mohou vývojáři definovat datové typy proměnných při jejich definici. U JavaScriptu toto není možné, jelikož je dynamicky typovaný a datový typ proměnné je znám až při spuštění skriptu, a to po přiřazení hodnoty k proměnné. Statické typování poskytuje lepší dokumentaci kódu a umožňuje předcházet chybám v kódu [13].

Skripty napsané v TypeScriptu není možné nativně spouštět ve webových prohlížečích, proto musí být kompilovány do JavaScriptu pomocí TypeScript nebo Babel kompilátoru [13].

```typescript
type UserRole = 'ROLE_USER' | 'ROLE_ADMIN';

interface User {
    id: number;
    first_name: string;
    last_name: string;
    roles: UserRole[];
}
```

Obrázek č. 3: TypeScript syntaxe
(Zdroj: Vlastní zpracování)

1.3. Nástroje pro vývoj webových aplikací

V této kapitole budou představeny hlavní technologie a nástroje, které budou použity při vytváření vlastního návrhu řešení a jeho implementaci.

1.3.1. Knihovna React

React je open-source javascriptová knihovna umožňující vytvářet interaktivní uživatelská rozhraní pro webové a mobilní aplikace. Knihovna vychází z principu SPA a je založena na deklarativním přístupu, kdy vývojář popisuje, jak má aplikace vypadat v daném okamžiku a React se poté stará o efektivní aktualizaci zobrazení aplikace. První verzi knihovny představila v roce 2013 společnost Facebook, která se spolu s nezávislými vývojáři podílí na kontinuálním rozvoji [14].

Základním stavebním kamenem aplikace jsou komponenty (*Components*), které představují znovupoužitelné nezávislé HTML bloky s vlastními funkcemi. Každá komponenta může mít své vlastnosti (*Props*) a svůj vnitřní stav (*State*), na základě kterých může měnit své chování a způsob zobrazení. Komponenty se mohou díky vlastnostem navzájem ovlivňovat a mohou si mezi sebou předávat různé datové struktury. Komponenty společně tvoří hierarchii, která je lehce upravitelná a rozšiřitelná [14].

State

State je objekt, který slouží k uchovávání vnitřního stavu komponenty. Na základě stavu komponenty je možné měnit způsob jejího vykreslení a při každé změně stavového objektu je komponenta znovu vykreslena. Vnitřní stav komponenty se může změnit například po interakci uživatele s komponentou [15].

JSX a TSX

JSX je rozšíření syntaxe k JavaScriptu, která se používá pro popis struktury uživatelského rozhraní. Pomocí JSX je možné zapisovat HTML strukturu do JavaScript souborů, to umožňuje uchovávat logiku komponenty společně s její strukturou ve stejném souboru. Používání JSX v Reactu není povinné, avšak díky JSX je kód přehlednější a snadnější na ladění [15].

TSX je typescriptová verze JSX, která obsahuje všechny vlastnosti JSX a přidává podporu statického typování, které je zmíněno v kapitole 1.2.7.

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

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

Obrázek č. 4: JSX syntaxe
(Zdroj: Vlastní zpracování dle [14])

Virtual DOM

Virtual DOM (VDOM) je odlehčená javascriptová reprezentace skutečného DOM, kterou React udržuje v paměti a synchronizuje se skutečným DOM. Aktualizace VDOM je poměrně rychlejší než u DOM, jelikož se na obrazovku uživatele nemusí nic vykreslovat. V případě, kdy se v Reactu změní vnitřní stav komponenty, VDOM se aktualizuje a následně probíhá porovnání s jeho předchozím stavem. Po vyhodnocení se v DOM aktualizují pouze ty části, které se změnily [15].

1.3.2. Framework Next.js

Problémem React aplikací může být způsob generování HTML stránky, které probíhá až v prohlížeči na straně klienta. To může přinášet vyšší nároky na výkon zařízení, ale také problémy s viditelností obsahu stránek pro vyhledávací roboty. Nejen tyto problémy řeší javascriptový framework Next.js, který je postavený na knihovně React.

Next.js nabízí dva způsoby generování stránky – Server-side Rendering (SSR) a Static Site Generation (SSG). U SSR dochází ke generování HTML na straně serveru při každém požadavku, zatímco u SSG dochází k vygenerování v době kompilace aplikace. Ke každé vygenerované stránce je přidružen minimální JavaScript kód, který je nezbytný pro fungování dané stránky. Po načtení stránky v prohlížeči je tento kód spuštěn a stránka se následně stává plně interaktivní. Tento proces se nazývá hydratace [16].

Next.js nabízí také další výhody jako je podpora TypeScriptu, možnost směrování aplikace na základě souborového systému, vytváření vlastních API koncových bodů, automatické rozdělování kódu na základě stránek nebo rychlé obnovení stránky po změně zdrojových souborů aplikace [16].

1.3.3. Knihovna Material UI

Material UI je knihovna React komponent vycházející z Google Material Designu umožňující rychle vytvářet interaktivní uživatelské rozhraní aplikace. Knihovna má zpracovanou rozsáhlou dokumentaci komponent a obsahuje příklady jejich použití. Velkou výhodou knihovny je možnost přizpůsobení vzhledu komponent, a to za použití různých způsobů stylování. Komponenty používají řešení CSS-in-JS, díky kterému je možné popsat vzhled komponent přímo v JavaScriptu a oproti běžnému způsobu vkládání CSS do stránky – odkaz na externí CSS soubor, stránka obsahuje pouze ty definice stylů, které jsou na dané stránce potřeba [17].

1.3.4. Apiary

Apiary je webová služba pro navrhování, vývoj a dokumentaci API. Jeho hlavní výhodou je možnost navrhnout přesnou podobu API bez nutnosti jeho programování a také možnost spolupráce týmu při jeho vytváření. Apiary díky generované dokumentaci pomáhá vývojářům při následném vývoji API a jeho implementaci v aplikaci. Navrhované API je také možné testovat pomocí Apiary Mock serveru, který umí reagovat na požadavky a vracet testovací data. Apiary používá pro popis API open-source formát API Blueprint [18].

1.4. Funkční modelování

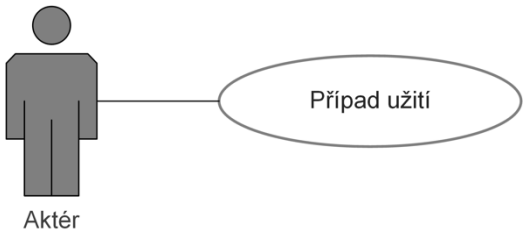
Funkční modelování se zabývá především zkoumáním a algoritmizací činností, procesů, které v informačním systému probíhají. Funkční model je možné popsat různými analytickými metodami, které se na zkoumané činnosti a procesy dívají z různých úhlů pohledu [19].

1.4.1. Slovní popis funkčního modelu

Metoda slovního popisu funkčního modelu se používá při řešení úloh menšího rozsahu. Vzhledem k její menší přehlednosti se však nepoužívá v dokumentaci informačních systémů, ale pro popis funkčního modelu webové aplikace je tato metoda dostačující [19].

1.4.2. Diagram případů užití

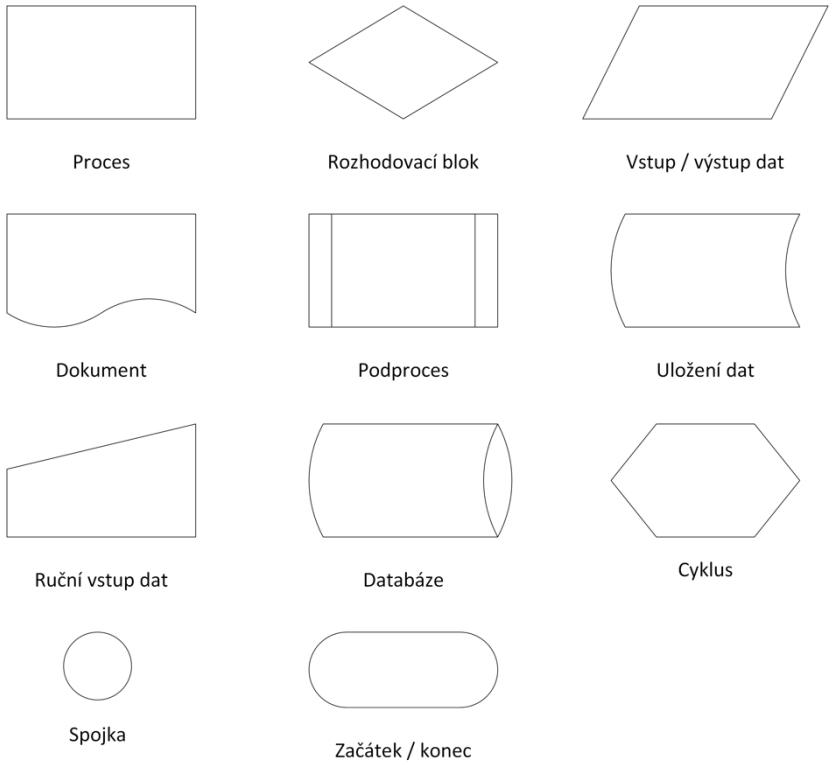
Diagram případů užití zobrazuje funkčnost systému pomocí aktérů a případů užití. V této práci je za systém považována samotná webová aplikace a aktéři představují uživatelské role. Případy užití definují jednotlivé funkcionality, které má systém umět a se kterými mají aktéři interagovat. Diagram však dále nedefinuje, jakým konkrétním způsobem má systém fungovat [20].



Obrázek č. 5: Základní prvky diagramu případů užití
(Zdroj: Vlastní zpracování)

1.4.3. Vývojový diagram

Vývojový diagram patří k nejčastěji používaným typům diagramů a slouží především ke grafickému znázornění jednotlivých kroků algoritmu, procesu nebo pracovního postupu. Mezi jeho hlavní výhody patří možnost zachytit větvení procesů na základě splnění či nesplnění požadovaných vstupních podmínek. Pro jeho znázornění jsou využívány obrazce různých tvarů, které se navzájem propojují pomocí šipek [19].



Obrázek č. 6: Základní značky vývojového diagramu
(Zdroj: Vlastní zpracování dle [19])

1.5. SWOT analýza

SWOT analýza je technika používaná k zhodnocení vnitřních a vnějších faktorů, které ovlivňují společnost nebo konkrétní projekt. Zkratka SWOT představuje počáteční písmena anglických slov strengths, weaknesses, opportunities a threats, která jsou překládána do češtiny jako silné a slabé stránky, příležitosti a hrozby [21].

Silné stránky vyjadřují, v čem daný subjekt vyniká a co jej odděluje od konkurence. Slabé stránky popisují negativní vlastnosti, které přispívají ke konkurenční nevýhodě subjektu. Příležitosti se týkají příznivých vnějších faktorů, které by mohly subjektu poskytnout konkurenční výhodu. Hrozby představují vnější faktory, které mohou subjekt poškodit [22].

Analýzu je možné vizuálně reprezentovat pomocí čtyř kvadrantů, kdy každý kvadrant vyjadřuje jeden prvek. Tato reprezentace umožňuje zobrazit výstupy analýzy ve snadno čitelném formátu [21].

2. ANALÝZA PROBLÉMU A SOUČASNÉ SITUACE

2.1. Základní údaje o společnosti

Název společnosti:	**Proof & Reason, s.r.o.**
Právní forma:	Společnost s ručením omezeným
Sídlo:	Kollárova 703, 508 01 Hořice
Identifikační číslo:	616 79 992
Daňové identifikační číslo:	CZ 616 79 992

PROOF & REASON

Obrázek č. 7: Logo společnosti
(Zdroj: [23])

2.2. Představení společnosti

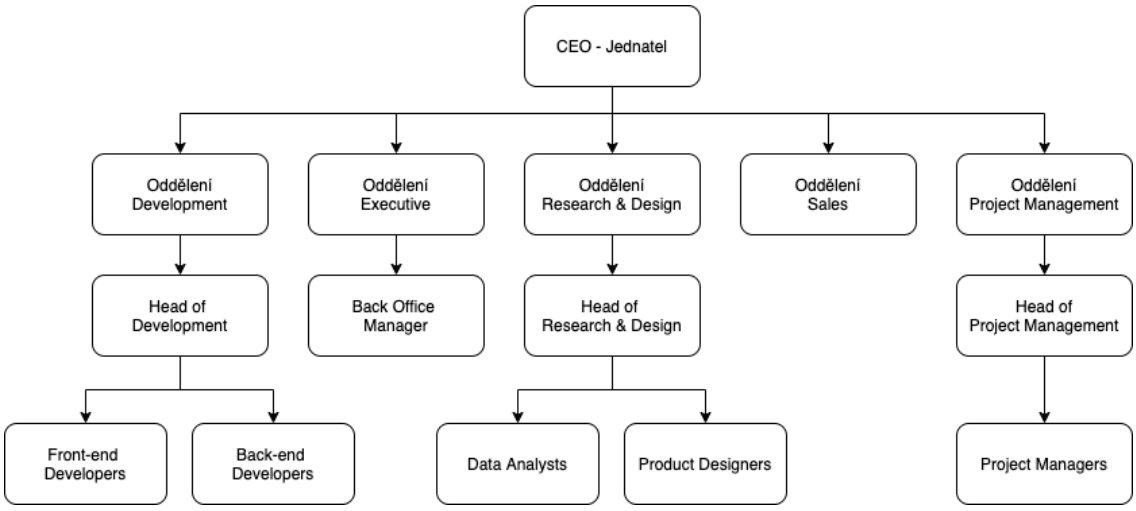
Společnost Proof & Reason, s.r.o. působí na trhu již od roku 1999 a poskytuje služby v oblasti tvorby digitálních produktů. Specializuje se na tvorbu digitálních produktů, od strategie a výzkumu přes UX, grafický design, vývoj až po 24/7 podporu. Svým zákazníkům poskytuje vlastní know-how a bohaté zkušenosti nasbírané z více než 20leté činnosti na trhu.

2.3. Poskytované služby

Společnost nabízí služby spojené s digitálními produkty, jsou to především:

- Webové portály a prezentace
- Mobilní aplikace
- Datová a webová analytika
- Uživatelské výzkumy
- Strategie, analýzy a přípravy zadávacích dokumentací
- Zakázkový vývoj
- Optimalizace konverzního poměru a evaluace digitálních produktů

2.4. Organizační struktura



Obrázek č. 8: Organizační struktura společnosti
(Zdroj: Vlastní zpracování)

Ve společnosti pracuje přibližně 40členný tým složený ze zaměstnanců a kontraktorů, kteří regionálně působí ve dvou pobočkách – v Brně a Praze. Na vrcholu společnosti stojí jednatel a následně je společnost rozdělena do třech základních oddělení (Development, Project Management a Research & Design) a dvou podpůrných (Executive a Sales). Základní oddělení mají svého manažera, tzv. Heada, který je zodpovědný za chod daného oddělení a jeho členové s ním řeší veškeré potřebné záležitosti.

Oddělení Development se zabývá vývojem produktů a součástí tohoto oddělení jsou Frontend (HTML, CSS, JS, React) a Backend (PHP) vývojáři. Ti úzce spolupracují s oddělením Research & Design, které má na starosti výzkumy, tvorbu strategií, uživatelské testování a tvorbu grafických návrhů. V tomto oddělení pracují datoví analytici a produktoví designéři. Plánování a organizovaní práce mají na starosti projektoví manažeři z oddělení Project Management. Dalším nezbytným oddělením je také Sales, které je zodpovědné za propagaci a prodej služeb, péči o klienty a sledování stavu všech projektů. Úlohou posledního oddělení – Executive a sem spadajícího back office manažera je podpora chodu společnosti, to zahrnuje především finanční a pracovněprávní záležitosti, nábor nových zaměstnanců, chod kanceláří, vedení porad a organizaci akcí.

2.5. Analýza současného stavu

Společnost v současné době používá pro řízení projektů volně dostupný tiketovací systém Redmine, kde jsou vedeny všechny projekty a v rámci nich jednotlivé úkoly. U každého úkolu se eviduje strávený čas nad úkolem, který se na straně pracovníka zaznamenává pomocí aplikace Toggl [24]. Zaznamenávání času se provádí vytvořením nového logu v aplikaci Toggl, kdy každý takto vytvořený log obsahuje strávený čas, identifikační číslo úkolu z Redmine, název projektu a popis vykonávané práce. Tento log se pak pomocí Toggl API automaticky importuje do interního webového fakturačního nástroje Timey a z tohoto nástroje se následně získává čas strávený nad úkolem, který se zobrazuje u úkolu v Redmine.

Timey je firemní webová aplikace, která slouží k zaznamenávání finančních položek, nastavování hodinových sazeb pro jednotlivé členy realizačních týmů a projekty, tvorbu výkazů práce a jejich následný export. Výkaz práce vždy obsahuje přehled odpracovaných hodin na určitém projektu za určité období. Exportovaný výkaz je možné nahrát do interního nástroje pro generování faktur – Proof Invoice Generator (PIG). Ten spolupracuje se službami iDoklad a Fakturoid, které pomocí REST API umožňují na základě podkladů vygenerovat fakturu.

V současné době není možné přímo v Timey upravovat vytvářené výkazy práce například po dodatečné dohodě s klientem, a proto je nutné výkazy upravovat v MS Excelu až po jejich exportování z Timey. Výsledkem tohoto nedostatku jsou nepřesné informace v Timey, jelikož nedochází k přesné aktualizaci dat o stavu projektu a odpracovaných a vyfakturovaných hodinách.

Dalším nedostatkem nástroje je nemožnost zobrazení a sledování některých důležitých metrik pro řízení projektů, vedení jednotlivých oddělení a finanční řízení společnosti. Také není možné si v Timey zobrazit uzavřené výkazy jednotlivých lidí za předchozí měsíce. Dále aplikaci chybí podpora responzivního zobrazení, tím se aplikace stává špatně použitelnou na mobilních zařízeních.

Z technického hlediska je u současného řešení problémem použitá technologie, jelikož je aplikace napsaná v jazyce Ruby on Rails a pro tuto technologii již společnost nemá specialisty. To způsobuje problémy s údržbou aplikace a jejím dalším rozvojem.

Vzhledem k tomu, že aplikace používá neaktualizované knihovny a přistupuje ke staré verzi Toggl API, může se stát, že aplikace přestane náhle zcela fungovat.

2.6. Analýza uživatelů Timey

Nástroj Timey v současné době používají především členové realizačních týmů, projektoví manažeři, back office manažer a jednatel společnosti. Těmto uživatelům pomáhá Timey řešit různé problémy, které budou uvedeny v následujících podkapitolách.

Členové realizačních týmů

Všichni členové realizačních týmů si v průběhu měsíce měří čas, který strávili nad úkoly z tiketovacího systému Redmine. Následně pak na začátku následujícího měsíce odesílají pomocí Timey výkaz práce ke schválení. Výkaz se generuje automaticky agregací všech logů importovaných z Toggl v minulém měsíci. Žádost o schválení výkazu se následně posílá e-mailem zodpovědné osobě za schvalování výkazů. Po schválení výkazu obdrží pracovník e-mail s podklady na jejichž základě mu je vyplácena mzda nebo tento podklad slouží pro potřeby vystavení faktury kontraktorem. Pokud se jedná o kontraktora, tak ten si následně může fakturu vytvořit ve svém nástroji, který pro tyto účely používá, nebo využít nástroj PIG, do kterého nahraje obdržené podklady k fakturaci. Následně fakturu odesílá společnosti k proplacení, to může udělat buďto manuálně pomocí e-mailu nebo odesláním vytvořené faktury rovnou z nástroje PIG.

Projektový manažer

Každý projektový manažer má na starost několik projektů a ke každému projektu zajišťuje většinou jednou za měsíc podklady k fakturaci, aby mohl připravit výkaz vykonané práce na daném projektu.

V rámci zajištění podkladů k fakturaci projektový manažer prochází v Timey práci, kterou chce klientovi fakturovat, a kontroluje odpracovaný čas, aby zjistil, jestli odpracované hodiny nepřekročily odhadovanou dobu trvání práce. Na základě podkladů vytváří výkaz práce, který pak exportuje z Timey a v případě individuálních požadavků klienta manuálně upravuje vzhled výkazu. Dále vytváří tzv. balík logů, který umožňuje sloučit více výkazů práce k různým projektům jednoho klienta s fakturou. Balík logů vzniká v Timey automaticky po označení vykázané práce jako vyfakturované. Následně

výkaz posílá klientovi ke schválení a po jeho schválení žádá o vystavení faktury na základě vytvořeného výkazu. Po vystavení faktury ji pracovník, který vystavuje faktury, posílá zpět projektovému manažerovi a ten ji v Timey manuálně propojuje s balíkem faktur, kdy k balíku zadává číslo a částku faktury. Následně fakturu posílá e-mailem klientovi k proplacení.

Projektový manažer také v průběhu měsíce sleduje v Timey poměr odpracovaných hodin k vyfakturovaným hodinám, a případně další finanční ukazatele.

Back office manažer

Pro pravidelné porady oddělení Project Management připravuje back office manažer přehled o finanční bilanci projektů. Jedná se o přehled ve vizualizačním nástroji Google Data Studio, který kombinuje vykázanou práci s reálně vyfakturovanou prací. Pro vytvoření přehledu musí vždy manuálně získávat podklady z Timey, konkrétně se jedná o vystavené faktury a přehled vykázané práce za jednotlivé projekty.

Jednatel

Jednatel průběžně sleduje stav a průběh projektů během jejich života. Zajímá se především o finanční ukazatele na projektech. Sleduje například poměr odvedené práce k výsledné vyfakturované částce a pro mzdové náklady projektu sleduje jaké byly náklady na odvedenou práci.

Jednatel také každý měsíc schvaluje výkazy práce členům realizačních týmů. U případných kontraktorů je výkaz schvalován před samotným vystavení faktury. Při schvalování se jednatel také zajímá o pracovní vytížení pracovníka – kolik hodin práce za předchozí měsíc odpracoval a jestli není dlouhodobě přetížený.

2.7. Analýza infrastruktury aplikace

Současná aplikace běží na vlastním serveru společnosti, který je pravidelně zálohován. Jak již bylo zmíněno v kapitole 2.5 popisující současný stav, aplikace je napsaná v jazyce Ruby on Rails. Do databáze aplikace se importují časové logy pomocí Toggl API a také jednotlivé aktivity, úkoly, cílové verze a projekty pomocí Redmine API. Import probíhá v pravidelných intervalech pomocí CRON úloh.

2.8. Požadavky společnosti na aplikaci

Vedení společnosti si na základě provedených rozhovorů s uživateli fakturačního nástroje stanovilo konkrétní požadavky, které by měla nová aplikace splňovat.

Hlavním požadavkem na aplikaci je zobrazování metrik (souhrny, grafy) pro řízení projektů, vedení týmů a finanční řízení firmy. Manažeři oddělení a jednatel budou mít možnost si zobrazit informace o jednotlivých členech oddělení, kolik odpracovali hodin na interních a externích projektech za různá období, jejich výkazy práce a odepsané a vyfakturované hodiny za určitá období. Projektoví manažeři si budou moci zobrazit přehled projektů, jejich finanční ukazatele za různá období a jednotlivé vytvořené výkazy práce.

Dalším důležitým požadavkem na aplikaci je zjednodušení přípravy výkazu práce. Aplikace by měla umožnit editaci výkazů, kdy bude možné do výkazu přidat práci, která ještě nebyla fakturována. Také bude možné u jednotlivých logů upravovat popis prováděné práce a nastavovat stav logu, který bude vyjadřovat, co se s ním má stát (fakturovat, odložit fakturaci nebo odepsat). V případě, kdy budou ve výkazu logy, které nemají přiřazený úkol, bude je možné přiřadit pomocí vyhledávání v seznamu úkolů ve vykazovaných projektech.

Mezi nové funkcionality patří také zasílání upozornění v aplikaci na důležité události jako je například neúspěšný import logů. Dále by v aplikaci měla být možnost spravovat týmy a přiřazovat uživatele do týmů.

Aplikace musí poskytovat přehledné a dobře použitelné rozhraní, které bude podporovat responzivní zobrazení, tak aby byla zajištěna snadná použitelnost aplikace na mobilních zařízeních.

Dalším požadavkem je zachování současných funkcionalit Timey jako je například export klientských výkazu do CSV a XLSX, nastavení hodinových sazeb členům týmů a různým typům činností na projektech, možnost manuálně vytvořit záznam odpracovaného času nebo manuálně vyvolat import z Toggl do Timey přímo v aplikaci.

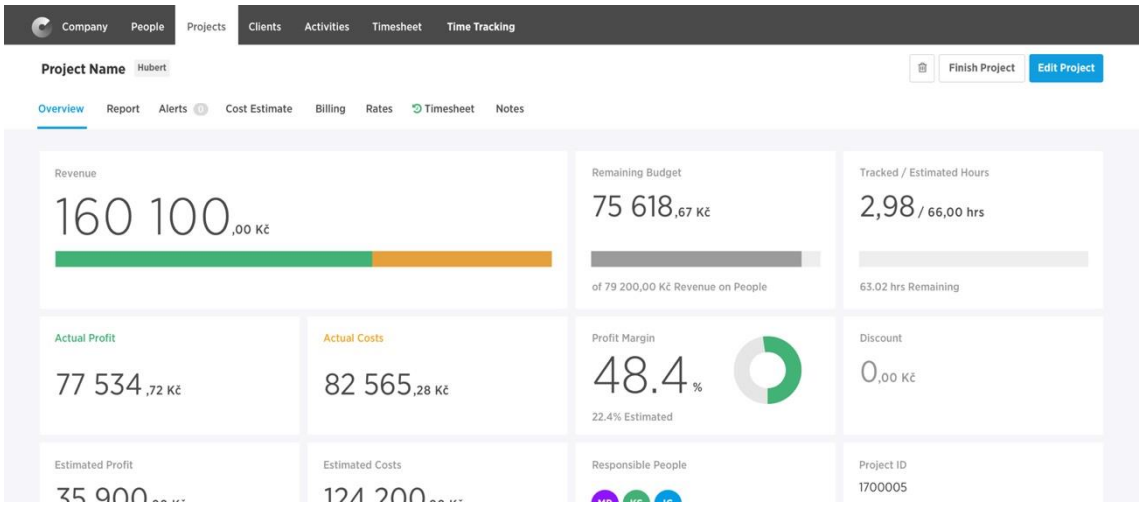
2.9. Analýza konkurenčních aplikací

Na základě požadavků společnosti na novou aplikaci byly vybrány dvě konkurenční aplikace, které by v případě splnění požadavků mohly být alternativou k vlastnímu řešení. V následujících podkapitolách budou tyto dvě aplikace popsány.

2.9.1. Costlocker

Costlocker je mobilní, desktopová a webová aplikace pro sledování času a finanční výkonnosti projektů, klientů a zaměstnanců, kterou vytvořila v roce 2013 česká firma 2FRESH [25].

Costlocker u projektů porovnává odhadované hodiny s odpracovanými a sleduje náklady na odpracovaný čas, které porovnává na základě nastaveného rozpočtu projektu. Aplikace také nabízí přehledy o tom, jaký klient je nejziskovější, zdali si zaměstnanci na sebe vydělají nebo kolik peněz stojí firmu péče o jednotlivé klienty. Costlocker má také dostupné vlastní API, pomocí kterého lze například importovat týdenní přehledy z aplikace Toggl nebo provádět import klientů, projektů a lidí z jiných aplikací. Dále také nabízí možnost napojení aplikace na fakturační služby iDoklad nebo Fakturoid [25].



Obrázek č. 9: Ukázka z webové aplikace Costlocker
(Zdroj: [25])

Costlocker nabízí 30denní zkušební verzi ve které je možné aplikaci využívat zdarma. Po zkušební době je zde limit do 4 uživatelů nebo 25 projektů, do kterého je možné aplikaci nadále používat bez poplatku. Při překročení limitu je již aplikace placená a stojí 20

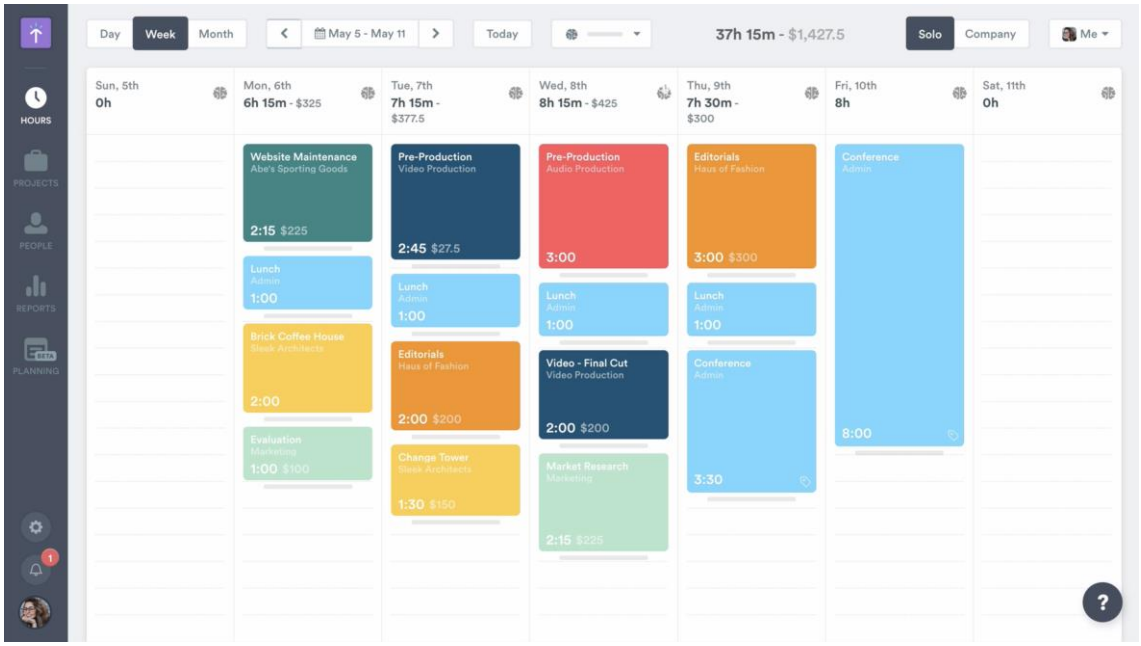
dolarů (přibližně 430 Kč) měsíčně za jednoho uživatele. V případě půlročního předplatného stojí jeden uživatel 16 dolarů (přibližně 344 Kč) měsíčně a u ročního předplatného 12 dolarů (přibližně 258 Kč) měsíčně [25].

Chybějící vlastnosti

- Z aplikace Toggl není možné do aplikace importovat jednotlivé logy, lze importovat pouze týdenní přehledy.
- Nemožnost exportovat výkazy do CSV souborů.
- Do aplikace není možné importovat jednotlivé úkoly z Redmine.
- V aplikaci není možné pracovníky přiřazovat do týmů.
- Nepříliš přívětivé uživatelské rozhraní.

2.9.2. Timely

Timely je aplikace zaměřená na řízení projektů, týmový management a sledování činností strávených na projektech, kterou vyvinula společnost Vikings z Norska. Nabízí jak webovou aplikaci, tak i mobilní a desktopovou. Timely si zakládá především na ochraně osobních údajů a uživatelské přívětivosti rozhraní aplikace [26].



Obrázek č. 10: Ukázka z webové aplikace Timely
(Zdroj: [26])

Timely nabízí tři druhy předplatného a to základní, rozšířené a neomezené. Pro požadované funkcionality by bylo potřeba neomezené předplatné, které stojí 20 dolarů (přibližně 430 Kč) na měsíc za jednoho uživatele [26].

Chybějící vlastnosti

- Do aplikace nelze na pravidelné bázi importovat data z aplikace Toggl a pro měření stráveného času na úkolech je nutné využívat Timely.
- Aplikace neumožňuje přímý export výkazů práce do služeb iDoklad nebo Fakturoid.

2.10. SWOT analýza současného řešení

Jedná se o modifikovanou SWOT analýzu, která je zaměřena na současné řešení – aplikaci Timey. Zpracovává se stejným způsobem jako běžná SWOT analýza. Nejdříve jsou nalezeny a definovány silné a slabé stránky, které vyjadřují vlivy vnitřního prostředí. Následně jsou určeny vlivy vnějšího prostředí, tedy příležitosti a hrozby.

Silné stránky

Jako hlavní silnou stránku současného řešení můžeme vidět synchronizaci dat s aplikací Toggl a tiketovacím systémem Redmine, díky které jsou v Timey neustále aktuální data o projektech, úkolech a odpracovaném času na těchto úkolech. Následně je možné vytvářet výkazy práce jak u projektů, tak i členů realizačních týmů a dále je možné tyto výkazy exportovat do CSV nebo XLSX. Další silnou stránkou je agregace veškerého času stráveného na projektech, díky tomu se nemůže stát, že by se nějaký čas strávený na projektech nepromítl do klientských výkazů. Současně lze za silné stránky považovat skutečnosti, že Timey si vyvinula společnost sama, běží na vlastním serveru společnosti a její data jsou pravidelně zálohována.

Slabé stránky

Hlavní slabou stránkou současného řešení je absence zobrazení některých metrik pro finanční řízení společnosti a projektů. Dále není možné upravovat položky ve vytvářeném výkazu práce a úpravy je možné provádět až po exportu výkazu do XLSX nebo CSV souboru. Současně aplikace není snadno rozšiřitelná o nové funkcionality z důvodu

použité technologie. Další slabou stránkou aplikace je chybějící responzivní zobrazení, které způsobuje špatnou použitelnost aplikace na mobilních zařízeních.

Příležitosti

Příležitost lze spatřit především v návrhu a implementaci nové aplikace za použití moderních technologií, která by poskytovala přehledné a dobře použitelné rozhraní a zajistila by možnost sledování důležitých metrik pro efektivnější finanční řízení společnosti a projektů. Nová aplikace by umožňovala zobrazit důležité informace týkající se například odpracovaných, odepsaných nebo vyfakturovaných hodin pro projektové manažery, manažery oddělení nebo vedení společnosti.

Hrozby

Jako hrozbu současného řešení lze vnímat použitou technologii Ruby on Rails, kdy v případě nějaké větší chyby by mohlo dojít k tomu, že problém bude časově a finančně náročné odstranit z důvodů chybějících specialistů na tuto technologii. Existuje tu také hrozba dočasné nedostupnosti aplikace v případě poškození vlastního serveru společnosti, na kterém aplikace běží.

Tabulka č. 1: Výstup SWOT analýzy současného řešení
(Zdroj: Vlastní zpracování)

Silné stránky	Slabé stránky
• Synchronizace dat s aplikací Toggl a Redmine • Export výkazů do CSV a XLSX souborů • Vlastní řešení běžící na vlastním zálohovaném serveru • Spolehlivá evidence veškerého stráveného času na projektech	• Chybějící některé metriky pro finanční řízení společnosti • Nemožnost upravovat položky ve vytvářeném výkazu • Použité technologie • Nemožnost jednoduchého přidání nových funkcionalit do aplikace • Chybějící responzivní zobrazení
Příležitosti	**Hrozby**
• Zefektivnění finančního řízení společnosti a projektů • Časová a ekonomická úspora	• Časově a finančně náročná oprava případné chyby aplikace • Nedostupnost aplikace v případě poškození vlastního serveru

2.11. Shrnutí

V analýze konkurenčních aplikací byly představeny dvě aplikace, které se obecně zaměřují na finanční řízení projektů a lidí. Aplikace Costlocker nenabízí příliš přívětivé uživatelské rozhraní a v aplikaci není možné vytvářet výkazy práce pro externí dodavatele. Aplikace Timely naopak nabízí přehledné uživatelské rozhraní, avšak neumožňuje import úkolu pomocí API. Obě aplikace také neumějí využívat aplikaci Toggl v plném rozsahu, a tak by bylo nutné tuto aplikaci v rámci celé firmy přestat používat.

Na základě provedených analýz, zjištěných nedostatků současného řešení a specifických požadavků na aplikaci se vedení společnosti rozhodlo k vytvoření vlastní nové webové aplikace. Vývoj vlastní aplikace je pro společnost nejvhodnějším řešení vzhledem k potřebným specifickým funkcím a možnosti aplikaci v budoucnu rozšiřovat o další potřebné funkcionality. Společnost očekává vyšší počáteční náklady na realizaci nové aplikace, avšak tyto náklady mohou být postupem času ve výsledku nižší než při využívání konkurenčních aplikací, které jsou distribuovány jako placené služby.

3. VLASTNÍ NÁVRH ŘEŠENÍ, PŘÍNOS PRÁCE

Hlavní část práce se zabývá návrhem a vývojem samotné webové aplikace Timey. Při návrhu a implementaci bylo vycházeno z poznatků zjištěných v analýze problému a současné situace.

3.1. Architektura aplikace

Architektura aplikace vychází z konceptu SPA a je vytvářena v programovacím jazyce JavaScript (ES6) s využitím frameworku Next.js a knihoven React.js a Material UI. Tyto technologie byly zvoleny na základě vlastních zkušeností autora s jejich používání a také z důvodu, že jsou ve společnosti Proof & Reason využívány na různých projektech, a tak v budoucnu nebude problém s dalším rozvojem aplikace. Zvolené technologie nabízí vysokou modularitu a umožňují vytvářet znovupoužitelný, jednoduše rozšiřitelný a udržitelný kód.

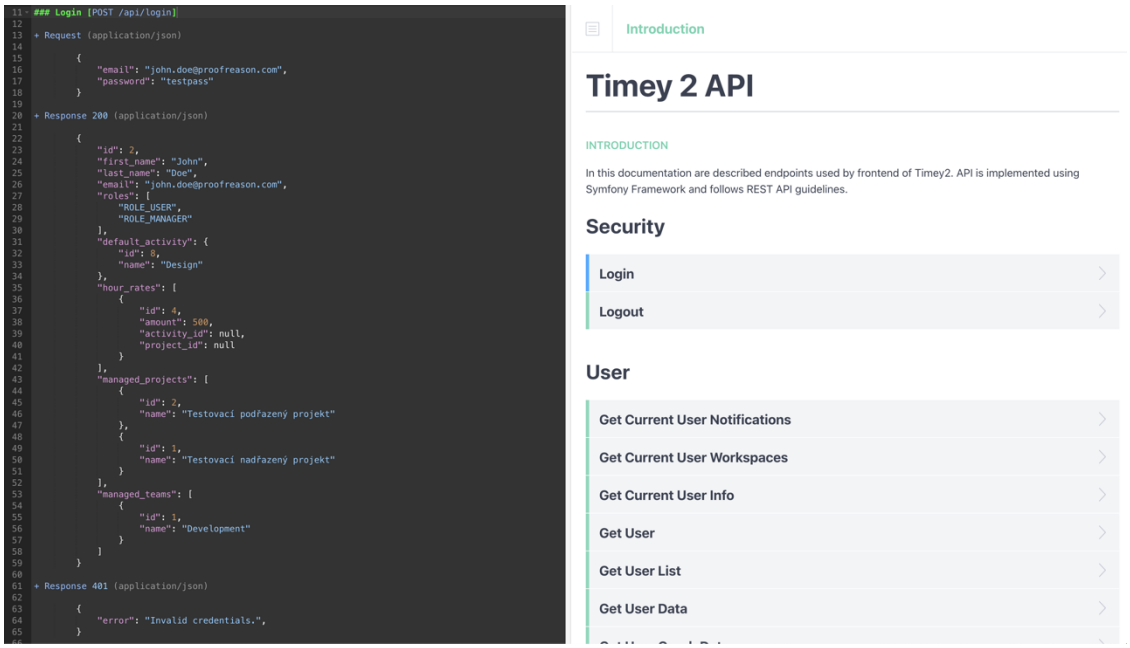
Použité technologie přinášejí také výhodu v podobě minimálních nároků na serverový hardware, jelikož v době sestavení aplikace se díky použití Next.js a funkce SSG vytváří statické HTML a JS soubory, které je možné umístit na CDN a na serveru tak nemusí běžet žádný proces. Nevýhodou tohoto přístupu může být delší doba úvodního načtení aplikace, jelikož aplikace musí dodatečně načítat data z externího zdroje a poté vygenerovat HTML stránky.

Jediným zdrojem dat, se kterým aplikace pracuje, je samostatné REST API, které společnost vyvíjí v programovací jazyce PHP a frameworku Symfony. REST API poskytuje veškerá potřebná data pro fungování aplikace, ať už se jedná o data ohledně uživatelů, projektů, výkazů nebo o analytická data. Tyto data jsou získávána a agregována na základě synchronizace s tiketovacím systémem Redmine a aplikací Toggl Track.

3.1.1. Komunikace s REST API

Komunikace s REST API probíhá pomocí HTTP protokolu, a to vždy pomocí konkrétního koncového bodu. Koncový bod je jeden konec komunikačního kanálu identifikovaný URL adresou společně s dostupnými HTTP metodami, které lze při

požadavku na tento koncový bod používat. Všechny dostupné koncové body API, ke kterým je v aplikaci možné přistupovat, jsou definovány v dokumentaci API. Tato dokumentace se vytváří automaticky při vývoji API a je dostupná ve službě Apiary popisované v kapitole 1.3.4.



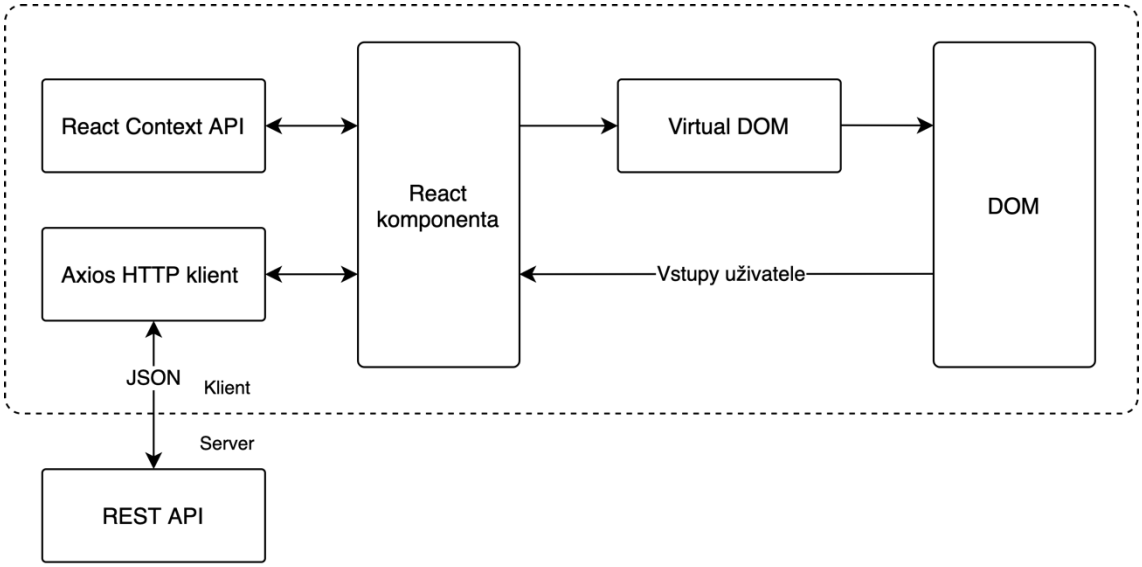
Obrázek č. 11: Ukázka dokumentace API v Apiary
(Zdroj: [18])

Jak je možné vidět na obrázku č. 11, u koncového bodu mohou být uvedeny parametry, které mají být odeslány společně s požadavkem. Dále je zde uveden příklad datové struktury odezvy, kterou koncový bod vrací při odpovědi na požadavek. Všechny požadavky a odpovědi jsou prováděny ve formátu JSON, který architektura REST doporučuje a je ideální volbou vzhledem k použití v JS aplikaci.

Pro jednodušší práci s API byla vybrána knihovna Axios [27], která poskytuje například automatický převod dat z JS objektů na JSON nebo naopak. Dále také umožňuje zpracování chybových hlášek na globální úrovni aplikace, díky tomu může být uživatel aplikace informován o chybách při komunikaci s API a zároveň kód aplikace zůstává přehledný, jelikož se tyto stavy nemusí řešit zvlášť u každého požadavku.

3.1.2. Tok dat v aplikaci

Před vytvářením samotné React aplikace bylo potřeba se zamyslet nad způsobem, jakým bude sdílena stavová logika mezi jednotlivými komponentami aplikace. React nabízí tři hlavní přístupy: High Order Components (HOC), Render Props a nejnovější přístup tzv. Hooks. Každý z přístupů má své výhody i nevýhody. Při použití HOC je nutné obalovat komponentu funkcí vyššího řádu, a to v případě častého používání může ovlivnit čitelnost kódu a také vždy nemusí být jasné, jaká data jsou do obalené komponenty předávána a odkud pochází. U Render Props může být problém s nutností častého zanořování komponent, a to může vést ke složité struktuře komponent. Tyto problémy se snaží řešit Hooks, které umožňují oddělit datovou logiku od komponenty a zároveň tuto logiku sdílet mezi všemi komponentami nezávisle na úrovni, ve které je komponenta používaná. Tento přístup byl také zvolen při vytváření aplikace.



Obrázek č. 12: Znázornění toku dat v aplikaci
(Zdroj: Vlastní zpracování)

Pro sdílení a správu globálního stavu aplikace bylo použito React Context API. Globální stav aplikace může uchovávat informace například o přihlášeném uživateli nebo notifikačních hláškách a k těmto informacím je možné přistupovat z jakékoliv komponenty. React Context API je součástí knihovny React a tak není potřeba instalovat další externí knihovnu. Mezi jeho další výhody patří jednoduchá syntaxe a možnost používat Hooks při přistupování ke globálnímu stavu z komponenty.

3.2. Zabezpečení aplikace

Zabezpečení aplikace je řešeno na dvou úrovních, na úrovni samotné aplikace a na úrovni REST API. Samozřejmostí je zabezpečené spojení pomocí HTTPS. Veškerý obsah aplikace je dostupný až po autentizaci uživatele, která probíhá pomocí e-mailu a hesla. Princip autentizace je založen na odeslání HTTP POST požadavku na koncový bod REST API, kdy v případě úspěšné autentizace odpovídá koncový bod stavovým kódem 200 společně s parametrem *set-cookie* v hlavičce odpovědi, který obsahuje identifikátor aktuální relace uchovávané na straně serveru. Tento identifikátor je uložen do cookies v prohlížeči a je následně odesílán v hlavičce každého požadavku na REST API. V případě pokusu o přístup k REST API bez tohoto identifikátoru je přístup zamítnut.

Toto řešení bylo ze strany společnosti zvoleno jako nejjednodušší a dočasné, a to z důvodu úspory času. V aktuálním řešení je na serveru uchováván stav o přihlášeném uživateli, avšak dle architektury REST by měl být server bezstavový pro jednoduché škálování REST API. Proto bude v budoucnu autentizace prováděna pomocí tzv. bearer tokenu, který nevyžaduje uchovávání stavu na serveru a také bude umožněno přihlášení pomocí Google účtu.

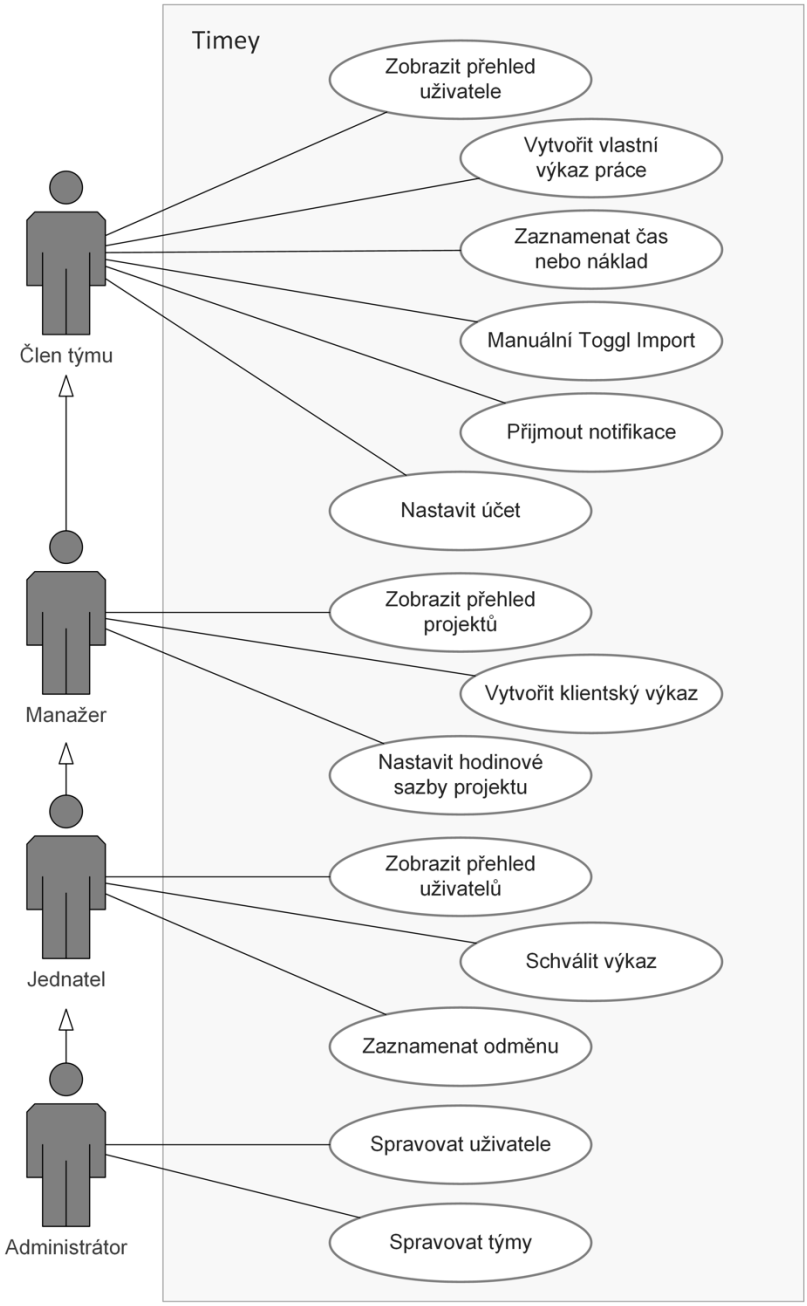
3.2.1. Uživatelské role

Na základě analýzy uživatelů současné aplikace byly definovány čtyři role, které vymezují práva uživatele. První rolí je administrátor, který má práva k provádění veškerých operací v aplikaci a také může nastavovat týmy a vytvářet nové uživatele. Druhou rolí je jednatel, ten může schvalovat výkazy členů realizačních týmů, udělovat odměny a sledovat přehledy všech projektů a lidí. Třetí rolí je manažer, mezi jeho práva patří možnost vytvářet klientské výkazy práce a sledovat přehledy projektů, u kterých je projektovým manažerem. Poslední rolí je člen týmu, který může vykazovat odpracovaný čas na projektech, přidávat dodatečný odpracovaný čas a dodatečné náklady, vytvářet své výkazy práce a sledovat své přehledy o vykázané práci. Na základě těchto rolí se mění některé prvky uživatelského rozhraní aplikace a také je omezován přístup ke koncovým bodům REST API.

3.3. Znázornění funkční části aplikace pomocí diagramů

V této části jsou pomocí diagramu případů užití a vývojových diagramů znázorněny vybrané funkční části aplikace. Vzhledem k rozsahu práce není možné znázornit veškeré funkční části aplikace, proto byly vybrány jen některé funkcionality a moduly.

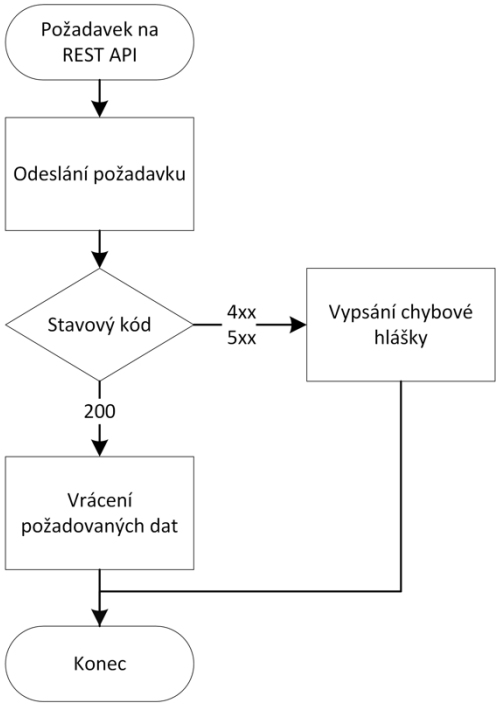
3.3.1. Diagram případů užití



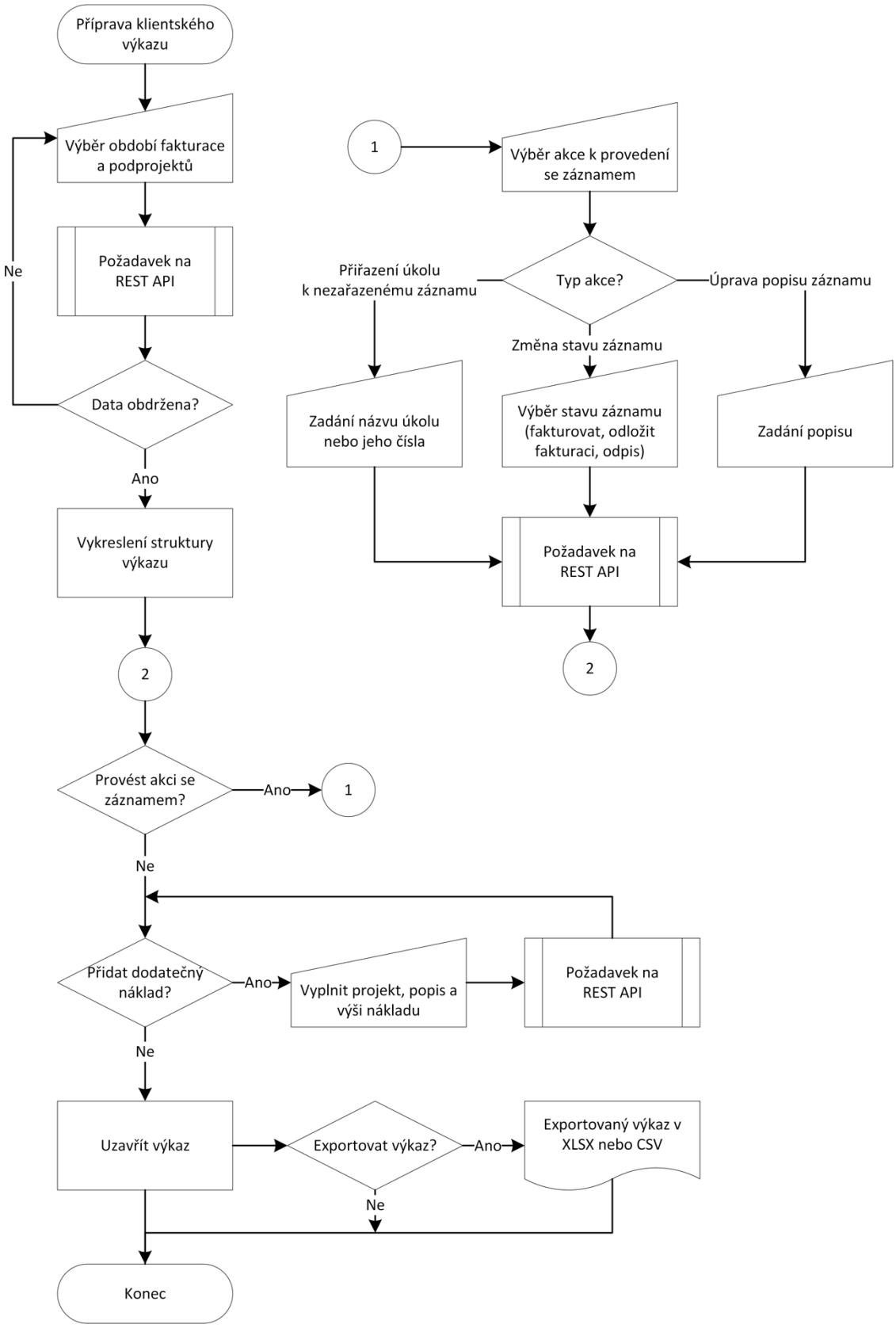
Obrázek č. 13: Diagram případů užití hlavních funkcionalit a modulů
(Zdroj: Vlastní zpracování)

3.3.2. Vývojové diagramy

Pro znázornění byl vybrán podproces odeslání požadavku na REST API společně s hlavním procesem přípravy klientského výkazu.



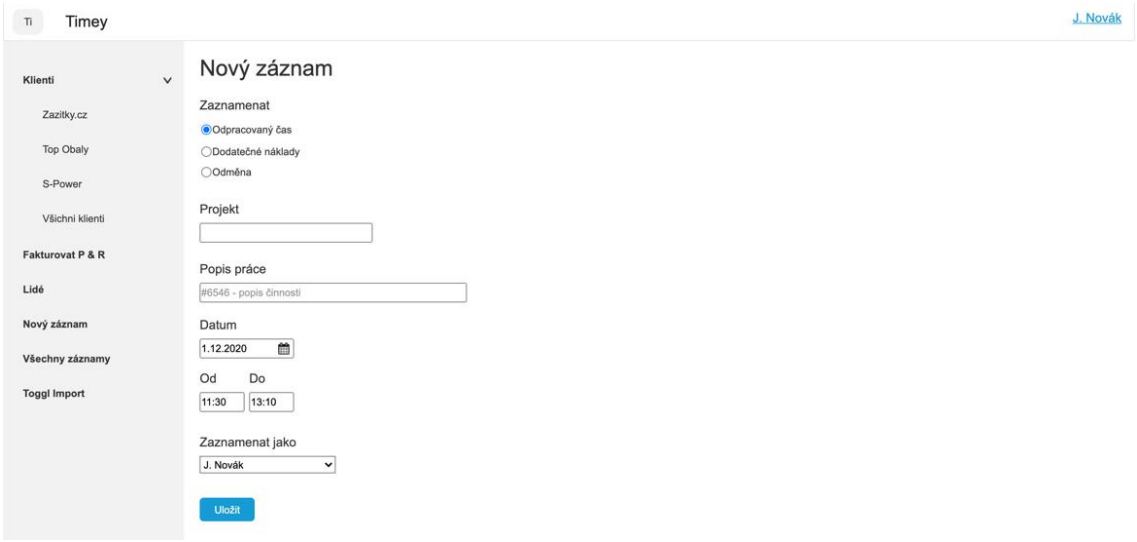
Obrázek č. 14: Vývojový diagram odeslání požadavku na REST API
(Zdroj: Vlastní zpracování)



Obrázek č. 15: Vývojový diagram přípravy klientského výkazu
(Zdroj: Vlastní zpracování)

3.4. Návrh a implementace uživatelského rozhraní

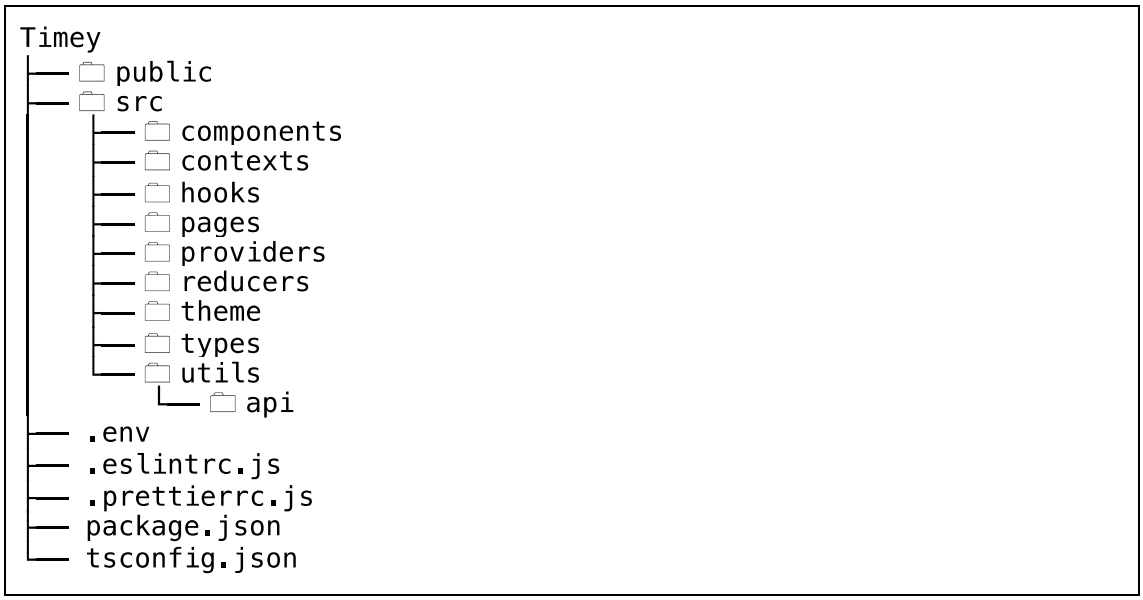
V této části práce bude představen samotný návrh a implementace uživatelského rozhraní pomocí již dříve zmíněných hlavních technologií React.js, Next.js a Material UI. Uživatelské rozhraní bylo navrhnuto a implementováno na základě prototypu, který společnost vytvořila. Tento prototyp definuje základní rozvržení stránek a rozmístění jednotlivých prvků uživatelského rozhraní aplikace.



Obrázek č. 16: Ukázka prototypu aplikace
(Zdroj: Vlastní zpracování dle společnosti Proof & Reason, s.r.o.)

3.4.1. Obecná nastavení

Prvním krokem bylo vytvoření základní adresářové struktury a nainstalování potřebných závislostí. Základní struktura projektu vychází ze struktury definované frameworkem Next.js, ten požaduje vytváření stránek do složky *./src/pages* a ukládání statických souborů (jako jsou například fonty) do složky *./public*. Další soubory jsou řazeny dle jejich typu, kdy například všechny typové definice se nacházejí ve složce *./src/types* a všechny komponenty jsou umístěny do složky *./src/components*. Komponenty jsou dále organizovány dle domény tak, aby například všechny komponenty související s klientským výkazem byly ve stejné složce.



Obrázek č. 17: Základní adresářová struktura aplikace
(Zdroj: Vlastní zpracování)

Na nejvyšší úrovni aplikace se nachází soubor *./src/pages/_app.tsx*. Ten zajišťuje vykreslení jednotlivých stránek definovaných v *./src/pages*. V *_app.tsx* jsou také definovány globální React Context Providers, které zajišťují dostupnost určitých kontextů (globální stavy a funkcionality) na nižších úrovních aplikace. Na nejvyšší úrovni je také řešena autentizace uživatelů, kdy v případě nepřihlášeného uživatele je vykreslen přihlašovací formulář.

Pro udržení jednotného formátování kódu, kvality kódu a přecházením chybám spojených například se zapomenutým středníkem, byl v projektu nastaven nástroj pro analýzu statického kódu a nástroj pro formátování kódu. Konkrétně se jedná o nástroje ESLint [29] a Prettier [30], které jsou v době psaní této práce nejčastěji používány při vývoji JS aplikací [31].

3.4.2. Styl uživatelského rozhraní

Při vytváření stylu uživatelského rozhraní bylo vycházeno z firemního vizuálu, který je založen na jednoduchosti, modrých barvách a specifickém písmu. Dále vzhledem k použití knihovny Material UI se do stylu uživatelského rozhraní projevují rysy Material Designu vytvořeného společností Google.

Knihovna Material UI nabízí sadu předdefinovaných komponent pro tvorbu uživatelského rozhraní, jedná se například o tlačítka, vstupní pole, tabulky a další. Tato

knihovna byla zvolena z více důvodů. Hlavním důvodem jsou komponenty vytvořené v React.js, které nabízí velké množství přizpůsobení a mají dostupné typové definice v TypeScriptu. Knihovna má také velmi rozsáhlou dokumentaci, která popisuje způsoby použití jednotlivých komponent a jejich dostupné parametry, které mění chování komponenty. Další výhodou je časová i finanční úspora, jelikož vytvoření komponent v podobné kvalitě jako nabízí Material UI by vyžadovalo nemalé množství práce. Nevýhodou knihovny mohou být její vyšší nároky na výkon zařízení a její datová náročnost, ale výše uvedené výhody tyto nevýhodu jistě převýší i vzhledem k použití knihovny na interním projektu a cílové skupině, která bude aplikaci používat.

Material UI umožňuje definovat vlastní styly komponent, a to jak na globální úrovni pomocí tématu, tak i přímo u samotné komponenty. V rámci globálního tématu je možné měnit výchozí parametry komponent, barevnou paletu, písmo, stíny a další parametry popsané v dokumentaci knihovny. Vlastní styly přímo u komponent je možné definovat více způsoby. Zvolena byla definice pomocí parametru *sx*, který je nově v Material UI od verze 5.0.0. Tento parametr umožňuje využívat proměnné definované v globálním tématu a také nabízí zkrácený zápis CSS vlastností, díky kterému zůstává kód relativně přehledný. Výhodou je také podpora zanořování selektorů, podobně jako v klasickém CSS.

3.4.3. Komponenty

V Reactu existují dva typy komponent, Functional Component (bezstavová) a Class Component (stavová). Jak už z názvů vyplývá, první typ používá běžnou JS funkci, druhý typ používá ES6 třídu, která rozšiřuje třídu *Component* z knihovny React. Rozdíl mezi nimi je tedy především syntaktický, kdy třídy jsou komplexnější a jinak přistupují ke vnitřnímu stavu a vlastnostem komponenty. Vzhledem k rozhodnutí používat React Hooks, jsou všechny komponenty v aplikaci psány jako funkcionální. To přináší řadu výhod, funkcionální komponenty především vyžadují méně kódu pro zápis, jsou přehlednější, jednodušší na pochopení a také se snadněji testují.

```
import { Typography } from '@material-ui/core';
import Link from 'next/link';
import React from 'react';

const Logo = () => (
    <Link href="/" passHref>
        <Typography
            variant="h6"
            color="primary.dark"
            component="a"
            noWrap
            sx={{ letterSpacing: 2 }}
        >
            <strong>TIMEY</strong>
        </Typography>
    </Link>
);

export default Logo;
```

Obrázek č. 18: Funkcionální komponenta vykreslující logo aplikace
(Zdroj: Vlastní zpracování)

3.4.4. Komunikace s REST API

Na základě dokumentace API v Apiary bylo potřeba v aplikaci vytvořit pomocné funkce, které slouží k vytváření požadavků na koncové body REST API a využívají k tomu sdílenou instanci knihovny Axios. Tyto funkce jsou používány napříč celou aplikací a jsou volány například při interakci uživatele s rozhraním aplikace.

```
import axios from 'axios';

const Backend = axios.create({
    baseURL: process.env.NEXT_PUBLIC_BACKEND_URL,
    withCredentials: true,
});

export default Backend;
```

Obrázek č. 19: Sdílená instance HTTP klienta Axios
(Zdroj: Vlastní zpracování)

Pomocné funkce byly vytvořeny dle dokumentace pro všechny koncové body REST API. Tento přístup umožňuje jednoduchou správu všech koncových bodů používaných

v aplikaci na jednom místě. Všechny funkce pracující s REST API mají také otypované vstupní parametry a očekávaný výstup funkce.

```
export const ProjectReportApi = {
  create: async (data: {
      from: string;
      to: string;
      projects: number[];
      source_project_id: number;
  }): Promise<AxiosResponse<ProjectReport>> => {
      return await Backend.post('/api/project-report', data);
  },
  ...
}
```

Obrázek č. 20: Pomocná funkce pro vytvoření nového výkazu
(Zdroj: Vlastní zpracování)

Vytvořené pomocné funkce je následně možné používat v samotných komponentách. Na obrázku č. 21 je možné vidět použití API při vytváření nového výkazu.

```
const createReport = async () => {
  const date = new Date();
  const from = format(startOfMonth(addMonths(date, -1)), 'yyyy/MM/dd');
  const to = format(endOfMonth(addMonths(date, -1)), 'yyyy/MM/dd');

  await ProjectReportApi.create({
      from,
      to,
      source_project_id: projectId,
      projects: [projectId, ...filteredProjects],
  }).then((response) => {
      const report = response.data;
      if (report) {
          Router.push('/project-report/' + report.id).then(() =>
              addSuccess('Výkaz úspěšně vytvořen!'),
          );
      }
  });
};
```

Obrázek č. 21: Funkce pro vytvoření nového výkazu
(Zdroj: Vlastní zpracování)

Uchování načtených dat v mezipaměti

Komponenty často mohou současně přistupovat ke stejným koncovým bodům REST API, to může vést k zbytečnému přetěžování API, jelikož při každém novém vykreslení komponenty může vznikat nový požadavek na API. Tento problém řeší React Hook knihovna SWR (zkratka z anglických slov stale-while-revalidate) [32], která pracuje ve třech hlavních krocích. Nejdříve vrací data z mezipaměti (cache), poté provádí nový požadavek (revalidace dat) a následně nabízí aktuální data. V případě více požadavků na stejný koncový bod API se stejnými parametry se požadavek provede pouze jednou, ale zároveň bude zajištěna aktuálnost dat ve všech komponentách. Knihovna přináší i řadu dalších výhod jako je revalidace dat při změně karty prohlížeče nebo možnost vyvolání manuální revalidace dat, kterou je možné využít například pro načtení aktualizovaného seznamu položek po odeslání formuláře.

3.4.5. Validace formulářů

Aby bylo předcházeno situacím, kdy budou částečně nebo špatně vyplněné formuláře odeslány na REST API, byla ve všech formulářích v aplikaci nastavena validace polí. Pro jednodušší práci s formuláři jsem zvolil knihovnu *react-hook-form* [33], která, jak již z názvu vyplývá, používá pro práci s formuláři React Hooks. Knihovna umožňuje nastavení vlastních pravidel pro validaci polí včetně vlastních chybových hlášek. Dále také umožňuje validovat skupinu dynamicky přidaných polí, která byla využita například u formuláře pro nastavení hodinových sazeb pro konkrétní aktivity a jejich přidávání a případné mazání.

3.4.6. Grafy

Pro potřeby zobrazení některých metrik ve formě grafů byla do projektu přidána React knihovna Recharts [34]. Tato knihovna byla vybrána především kvůli možnosti jednoduše vytvářet responzivní animované grafy různých typů, velkému množství přizpůsobení a možnosti definice vlastních stylů. Grafy jsou sestavovány komponentovým přístupem, kdy nejdříve je definována komponenta určující typ grafu a následně jsou do této komponenty vkládány potřebné osy, sloupce, legenda a další prvky.

3.5. Popis modulů

V této části jsou popsány stěžejní funkcionality aplikace Timey. Zcela jistě v práci není prostor pro detailní popis všech funkcionalit, z toho důvodu budou popsány pouze hlavní části aplikace, které tvoří stěžejní funkcionalitu řešení. Popis některých modulů je doplněn o snímky z aplikace, které umožňují lépe vyjádřit funkcionalitu než textový popis.

3.5.1. Modul uživatele

Tento modul zajišťuje veškeré funkcionality týkající se uživatele, jako je přihlášení, odhlášení a autorizace na základě role při zobrazování jednotlivých modulů aplikace. Na uživatele můžeme pohlížet ze dvou úhlů pohledu, z pohledu samotného přihlášeného uživatele a z pohledu role administrátora.

Vytvoření nového uživatele

Role administrátor může vytvořit nového uživatele. Při vytváření vyplňuje jméno, příjmení, e-mail a heslo. Dále pak roli, výchozí aktivitu (Design, Development, Management a další), kterou bude uživatel primárně vykonávat a dále tým, do kterého bude uživatel zařazen. Následně je možné nastavit hodinovou sazbu pro jeho výchozí aktivitu a hodinové sazby pro případné další aktivity.

Nastavení uživatele

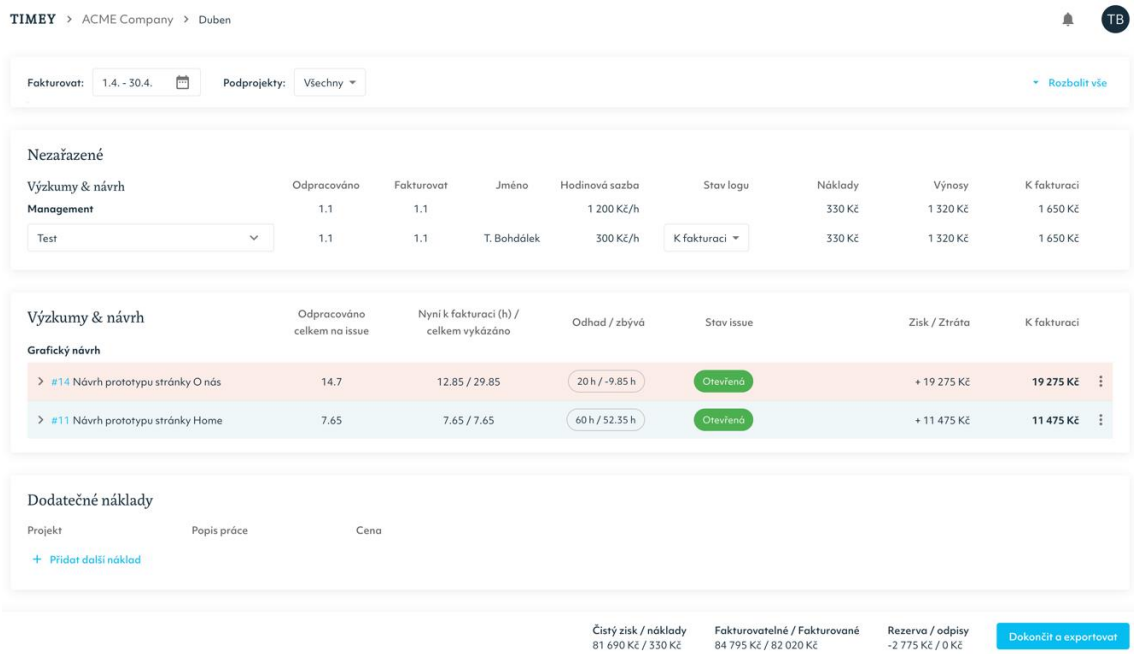
Přihlášený uživatel má k dispozici modální okno nastavení účtu. Pro synchronizaci dat s Redmine a aplikací Toggl musí mít uživatel nastavený API klíč z Redmine účtu, Toggl API token a Toggl Workspace, ve které bude zaznamenávat odpracovaný čas. Dále má uživatel také možnost nastavit údaje pro službu Fakturoid, konkrétně se jedná o e-mail, název účtu a API klíč.

3.5.2. Modul klientský výkaz

Tento modul slouží pro práci s výkazem práce na klientských projektech a tvoří hlavní součást aplikace, na kterou navazují další podpůrné moduly.

Vytvoření nového výkazu

Nový výkaz může vytvořit uživatel s rolí manažer, a to pouze u projektů, na kterých je projektovým manažerem. Při vytvoření nového výkazu je nastaven výchozí filtr období na minulý měsíc a zároveň je nastaven hlavní projekt, ze kterého je výkaz vytvářen včetně jeho případných podprojektů. Po vytvoření výkazu je toto výchozí nastavení možné změnit.



Obrázek č. 22: Příprava klientského výkazu
(Zdroj: Vlastní zpracování)

Kontrola jednotlivých pracovních záznamů

Výkaz je strukturován do dvou hlavních částí, nezařazené záznamy a podprojekty. V nezařazených záznamech se zobrazují všechny záznamy, které byly k projektu vytvořeny, ale neobsahují povinné identifikační číslo úkolu z Redmine. V druhé části výkazu jsou záznamy řazeny dle podprojektů, cílových verzí, úkolů a aktivit. Každý úkol a aktivita obsahuje souhrny časů a částek, které jsou vypočítány na základě pracovních záznamů a tyto souhrny jsou dále využity v celkovém souhrnu výkazu.

U každého pracovního záznamu je možné změnit popis práce, fakturovaný čas a stav logů, který určuje, zdali se má záznam fakturovat, nefakturovat nebo odepsat. U jednotlivých úkolů je také k dispozici kontextová nabídka umožňující odepsat celý

úkol. Nepřiřazeným záznamům je možné dodatečně přiřadit úkol pomocí modálního okna a fulltextového vyhledávání v seznamu úkolů v projektech.

Přidání dodatečných nákladů

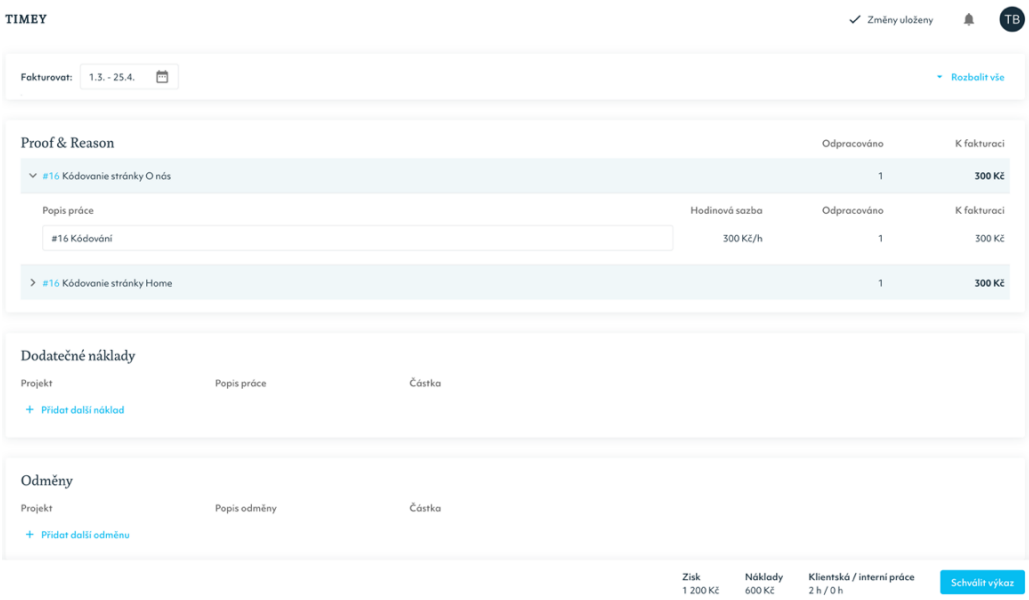
Do výkazu je možné přidat dodatečné náklady, které se mají klientovi vykázat. Každý záznam obsahuje projekt nebo podprojekt, kterého se náklad týká. Dále je vyplňován popis nákladu a částka.

Uzavření a export výkazu

Po uzavření výkazu je možné provést export výkazu do formátů XLSX a CSV, který projektový manažer posílá klientovi na schválení. Po schválení výkazu klientem je možné k výkazu přiřadit číslo faktury.

3.5.3. Modul výkaz člena týmu

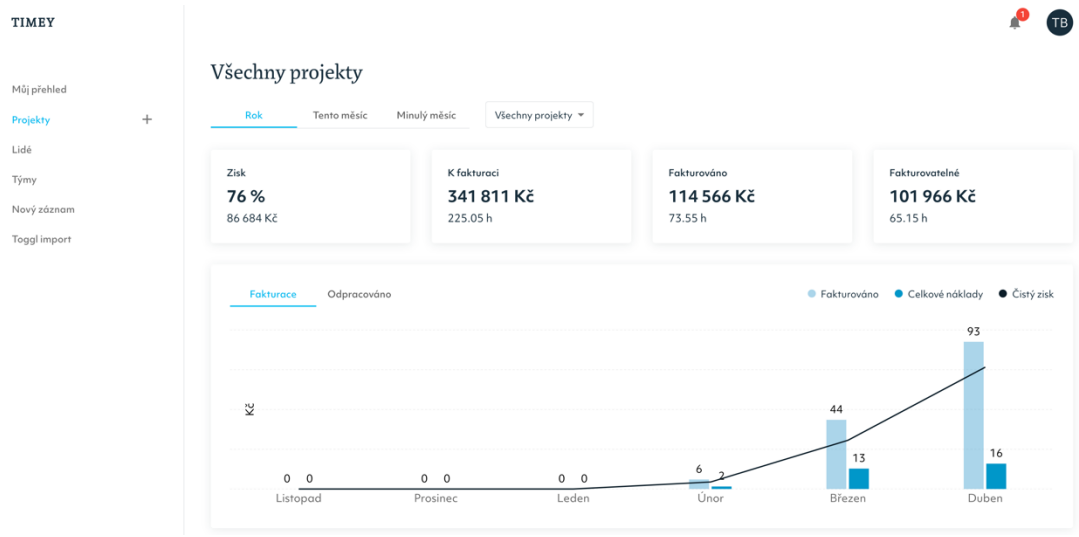
Tento modul je založen na stejném základu jako modul klientský výkaz a vytvoření výkazu probíhá podobným způsobem. Jednotlivé časové záznamy ve výkazu jsou hierarchicky řazeny dle projektů a úkolů. K výkazu má možnost uživatel přidat dodatečné náklady. Po zkontrolování záznamů odesílá člen výkaz ke schválení. Následně má role jednatel možnost tento výkaz schválit a případě přidat mimořádné odměny. Po schválení výkazu má člen týmu možnost vytvořit fakturu ve Fakturoidu (v případě vyplněných údajů v nastavení účtu) nebo může exportovat výkaz do XLSX nebo CSV, stejně jako u klientského výkazu. V případě vytvoření faktury ve Fakturoidu může následně fakturu odeslat k proplacení.



Obrázek č. 23: Výkaz člena týmu z pohledu role jednatel
(Zdroj: Vlastní zpracování)

3.5.4. Modul přehled projektů

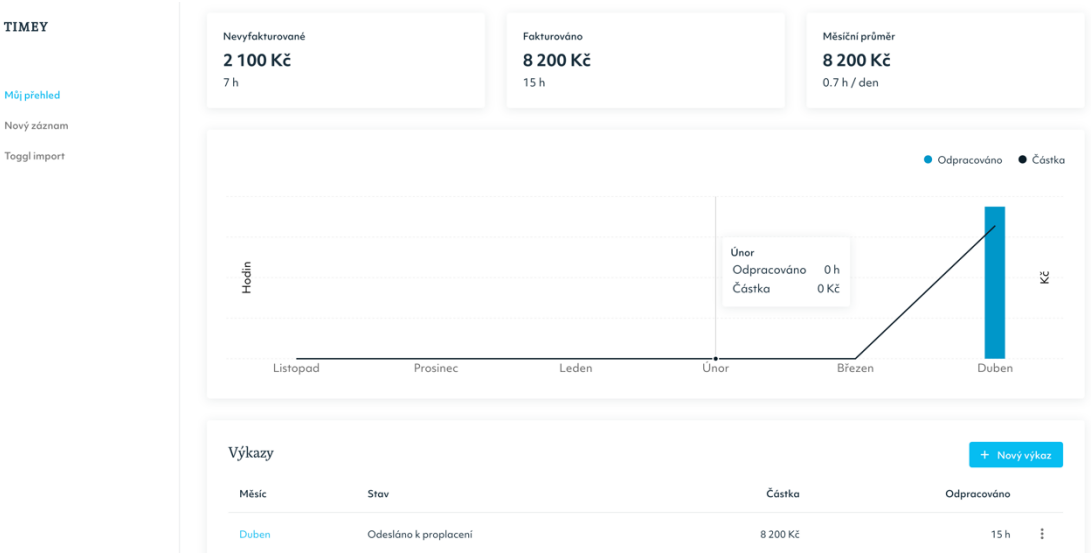
Tento modul slouží pro získání přehledu o stavu projektů a také umožňuje zobrazit detaily jednotlivých projektů. Přístup k těmto přehledům je omezen na základě rolí, kdy role manažer si může zobrazit spravované projekty a role jednatel má přístup ke všem projektům. Přehled nabízí nejdůležitější shrnující údaje o projektech a grafy historie fakturace a odpracovaného času. V rámci přehledu je také možné provádět filtraci časového období a projektů. V případě přehledu všech projektů se na konci stránky nachází seznam projektů, pro které jsou metriky zobrazované. Pokud je zobrazen detail projektu, jsou zde vypsané vytvořené výkazy.



Obrázek č. 24: Přehled všech projektů
(Zdroj: Vlastní zpracování)

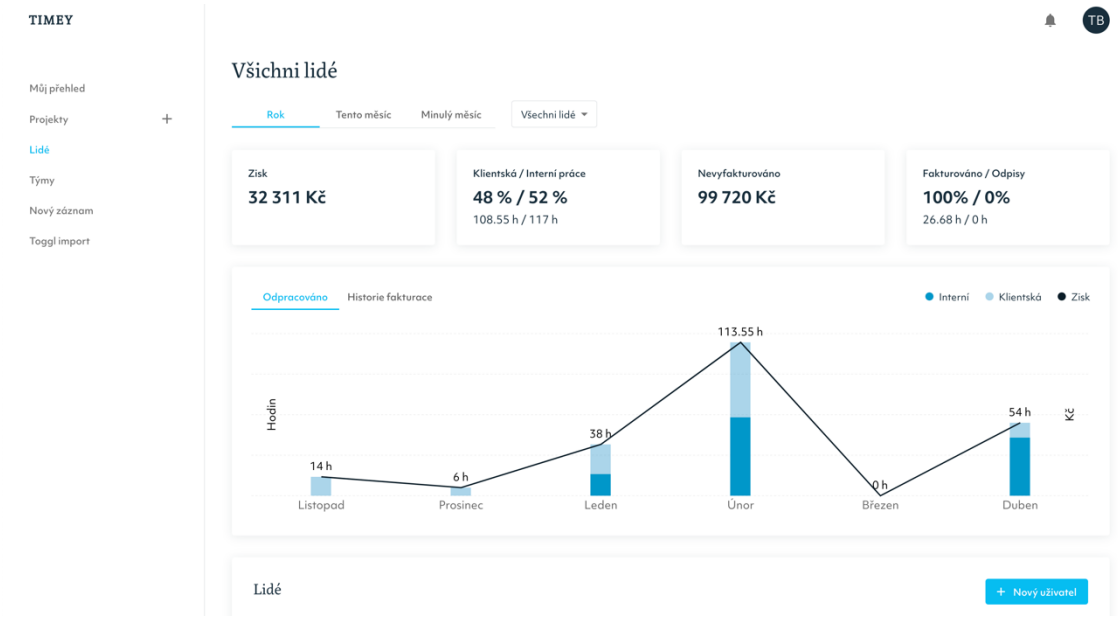
3.5.5. Modul přehled uživatelů

Tento modul slouží ke sledování důležitých metrik týkajících se uživatelů Timey. Modul se zobrazuje jako úvodní obrazovka po přihlášení uživatele do aplikace a poskytuje shrnující údaje o fakturaci a její historii. Dále zobrazuje seznam vytvořených výkazů, kdy u každého výkazu je možné vyvolat kontextovou nabídku, která nabízí dostupné akce k výkazu. Tyto akce jsou zobrazovány na základě aktuálního stavu výkazu.



Obrázek č. 25: Úvodní obrazovka uživatelské role člen týmu
(Zdroj: Vlastní zpracování)

V případě role jednatel modul slouží k získání přehledu o stavu pracovní výkonnosti všech nebo konkrétních členů tymů. Modul umožňuje zobrazování metrik na základě filtru časového období a také výběru konkrétních členů tymů. Mezi nejdůležitější sledované metriky patří zisk, poměr klientské a interní práce a nevyfakturovaná práce.



Obrázek č. 26: Přehled všech uživatelů z pohledu uživatelské role jednatel
(Zdroj: Vlastní zpracování)

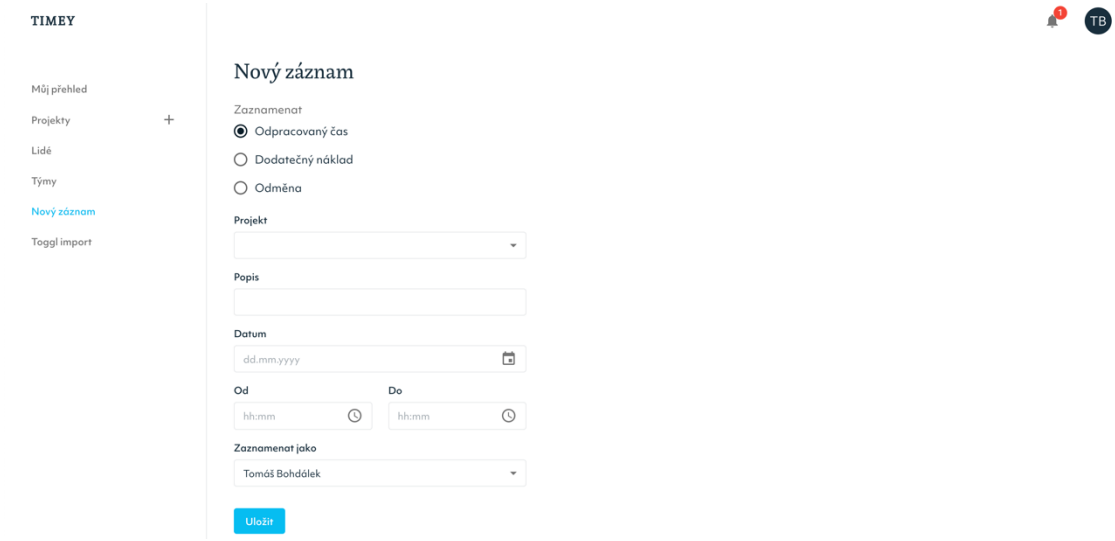
3.5.6. Modul notifikace

Tento modul slouží k upozornění uživatele na určité události. Počet notifikací s jejich seznam se nachází v hlavičce stránek. U role člena týmu se může jednat o upozornění na chybu při importu logů z aplikace Toggl do Timey nebo o schválení výkazu jednatelem. V případě role jednatel se zde může objevit upozornění na neschválený výkaz, který je potřeba schválit. Po otevření seznamu notifikací a kliknutí na notifikaci je uživatel přesměrován na entitu, které se notifikace týká. V seznamu notifikací je také možné označit všechny notifikace jako přečtené.

3.5.7. Modul nový záznam

Tento modul umožňuje vytvářet nové záznamy, které se následně projevují do vytvářených výkazů. Všechny role mají možnost manuálně zaznamenat odpracovaný

čase a vytvořit dodatečný náklad. Role jednatel má pak navíc možnost vytvořit odměnu, která bude udělena určitému uživateli.



Obrázek č. 27: Vytvoření nového záznamu
(Zdroj: Vlastní zpracování)

3.5.8. Modul Toggl import

Tento modul umožňuje uživateli vyvolat manuální import logů z aplikace Toggl a dále zajišťuje zobrazení reportu z posledního proběhnutého importu. Report informuje o úspěšnosti či neúspěšnosti importu konkrétních logů a v případě neúspěchu je uživateli vypsán konkrétní důvod, z jakého se import neprovedl.

3.6. Testování aplikace

Důležitou součástí vývoje aplikace je její testování. Tato část práce s zabývá testováním aplikace v průběhu implementace a dále uživatelským testováním, které bylo prováděno s budoucími uživateli aplikace.

Testovací prostředí

Aby bylo možné aplikaci průběžně testovat během implementace mimo vývojové prostředí a také provádět uživatelské testování, bylo pro tyto účely vytvořeno testovací prostředí pracující s testovacími daty. V tomto prostředí běží samotná aplikace, REST API a nová instance Redmine. Sestavení a nasazení aplikace do testovacího prostředí probíhá automaticky pomocí Gitlab Continuous Integration.

3.6.1. Testování v průběhu implementace

V rámci implementace uživatelského rozhraní byly průběžně testovány hlavní části aplikace. Z důvodu úspory času při první fázi implementace uživatelského rozhraní testování probíhalo manuálně a pouze ve webovém prohlížeči Chrome (verze 89) a na mobilních zařízeních s operačním systémem iOS 14 pomocí služby BrowserStack. V prohlížeči Chrome bylo využíváno DevTools, které nabízí řadu nástrojů pro testování aplikací. Mezi nejvíce používané nástroje patřil nástroj pro sledování síťové aktivity (odesílání požadavků na REST API a příjem odpovědí) a nástroj pro simulaci mobilních zařízení. Vzhledem k použitým technologiím je očekávána funkčnost aplikace ve všech moderních webových prohlížečích (Google Chrome, Firefox, Safari a Microsoft Edge).

3.6.2. Uživatelské testování

Pro získání zpětné vazby k uživatelskému rozhraní a odhalení případných chyb, bylo provedeno uživatelské testování. Testování probíhalo podle předem připravených scénářů pro všechny uživatelské role (člen týmu, projektový manažer, jednatel a administrátor) a testování se zúčastnili pracovníci a jednatel společnosti. Testovací scénáře byly sestaveny z úkolů, které dané role budou v aplikaci nejčastěji vykonávat.

U každé akce v testovaných scénářích bylo provedeno vyhodnocení. Úspěch byl označen jako *ANO*, částečný úspěch – *ANO**, neúspěch – *NE*. V případě neúspěchu nebo částečného úspěchu je důvod uveden v poznámce u dané akce.

Administrátor

Tabulka č. 2: Vyhodnocení testování role administrátor
(Zdroj: Vlastní zpracování)

	Akce	Úspěch
1.	Vytvoření nového uživatele a nastavení existujícího uživatele.	ANO
2.	Vytvoření nového týmu a přiřazení uživatele k existujícímu týmu.	ANO
3.	Nastavení hodinových sazeb pro projekty a všechny uživatele.	ANO

Jednatel

Tabulka č. 3: Vyhodnocení testování role jednatele
(Zdroj: Vlastní zpracování)

	Akce	Úspěch
1.	Přidání odměny za práci členovi týmu.	ANO
2.	Zobrazení notifikace v případě výkazu, který čeká na schválení.	ANO
3.	Schválení výkazu člena týmu.	ANO
4.	Nastavení hodinové sazby projektu nebo člena týmu.	ANO
5.	Zobrazení přehledu všech uživatelů, filtrování dle období a jména uživatele, zobrazení různých metrik jako je zisk a poměr klientské a interní práce.	ANO
6.	Zobrazení přehledu všech projektů, filtrování dle období a podprojektů, zobrazení různých metrik jako je částka k fakturaci, zisk projektu a historie fakturace.	ANO

Člen týmu

Tabulka č. 4: Vyhodnocení testování role člen týmu
(Zdroj: Vlastní zpracování)

	Akce	Úspěch
1.	Nastavení Redmine API klíče, Toggl API tokenu a údajů k Fakturoid účtu. Možnost výběru Toggl Workspace.	ANO
2.	Vytvoření nového nákladu.	ANO
3.	Zobrazení poslední proběhnuté synchronizace a vyvolání manuální synchronizace aplikace Toggl s Timey.	ANO
4.	Vytvoření nového výkazu, možnost přidání dodatečného nákladu, přiřazení úkolu k nezařazenému záznamu a upravení popisu záznamu. Odeslání výkazu ke schválení. *Poznámka: Nefunkční výběr projektů při přidávání dodatečného nákladu.	ANO*
5.	Zobrazení notifikace po schválení výkazu jednatelem.	ANO
6.	Vytvoření faktury ve Fakturoidu, její stažení a případné smazání.	ANO
7.	Odeslání faktury k proplacení.	ANO
8.	Zobrazení přehledu fakturace za minulé měsíce, vytvořených výkazů, nevyfakturované částky a dalších metrik.	ANO

Projektový manažer

Tabulka č. 5: Vyhodnocení testování role projektový manažer
(Zdroj: Vlastní zpracování)

	Akce	Úspěch
1.	Nastavení hodinové sazby projektu.	ANO
2.	Vytvoření nového klientského výkazu, výběr fakturačního období, možnost přiřazení úkolu k nezařazenému záznamu, přidání dodatečného nákladu, upravení popisu záznamu a možnost odepsání nebo nefakturování záznamu. *Poznámka: Nefungující vyhledávání úkolu dle jeho čísla.	ANO*
3.	Uzavření výkazu a jeho export do formátu XLSX a CSV.	ANO
4.	Přiřazení čísla faktury k výkazu a úprava čísla faktury.	ANO
5.	Zobrazení přehledu projektu, vyfakturovaných částek, odpracovaných hodin a dalších metrik. Zobrazení vytvořených výkazů.	ANO

3.7. Budoucí vývoj aplikace Timey

Celá aplikace byla vyvíjena s ohledem na modulárnost a snadnou rozšiřitelnost o nové funkcionality. Hlavním cílem do budoucnosti je spuštění aplikace do ostrého provozu, které by mělo proběhnout během několika následujících měsíců, a to po dořešení hlavních nedostatků, které se naskytly během vývoje a testování aplikace. Důležitým krokem před spuštěním bude také migrace dat ze stávající aplikace do nového Timey.

Pro zvýšení zabezpečení účtů bude přidán způsob přihlášení do aplikace pomocí Google účtů, které nabízejí možnost dvoufázového ověření. Dále bude upraven způsob autentizace uživatele, který byl popsán v kapitole 3.2.

Mezi hlavní funkcionality, které budou v průběhu dalšího vývoje přidány do aplikace, patří hromadné úpravy ve výkazech, kdy bude možné ve výkazu označit více projektů, cílových verzí, úkolů nebo záznamů a nad nimi bude možné provádět hromadné operace. To povede k rychlejší přípravě výkazu a ušetření práce projektovým manažerům.

Jako doplněk k notifikacím bude přidán na úvodní obrazovku aplikace seznam úkolů, které čekají na zpracování, aby se minimalizovalo zapomínání. Dále bude přidána

integrace služby iDoklad, která bude alternativou k Fakturoidu a uživatel tak bude mít možnost volby, jakou službu bude primárně používat.

3.8. Ekonomické zhodnocení

V této části bude provedeno zhodnocení nákladů na celou aplikaci včetně návrhu prototypu, implementace REST API a dalších činností, které byly při realizaci této aplikace potřeba, ale nebyly součástí této práce. Náklady na jednotlivé činnosti byly vypočítány jako mzdové náklady na hodinu práce krát počet odpracovaných hodin.

Náklady na provoz frontendové aplikace vytvářené v této práci jsou pro společnost nulové, jelikož je aplikace hostována u služby Netlify nabízející plán zdarma, který je pro tuto aplikaci plně dostačující. Náklady na provoz REST API je složité určit, protože společnost využívá větší infrastrukturu, na které běží velké množství projektů a společnosti je fakturován celkový pronájem této infrastruktury. Odhadem se ale cena může dle ceníku poskytovatele infrastruktury pohybovat okolo 2 400 Kč / rok.

Tabulka č. 6: Celkové náklady na realizaci
(Zdroj: Vlastní zpracování)

Položka	Cena
Analytické práce a návrh prototypu	170 398 Kč
Implementace REST API	74 820 Kč
Kódování šablon a implementace UI	28 376 Kč
Administrativa (projektové řízení, pravidelné status updaty, …)	40 677 Kč
Provozní náklady na 1 rok	2 400 Kč
Celkem	**316 671 Kč**

3.9. Přínosy práce

Hlavním přínosem práce je vytvoření jednoduché, přehledné a snadno rozšiřitelné aplikace využívající moderní technologie, která splňuje požadavky společnosti Proof & Reason, s.r.o., výrazně zjednodušuje práci při vykazování odvedené práce na projektech a podporuje finanční řízení společnosti.

Přínosem pro uživatele aplikace je její jednodušší používání díky přehlednému uživatelskému rozhraní. Aplikaci je možné pohodlně používat i na mobilních zařízeních

a díky integraci nástroje pro generování faktur umožňuje vystavovat faktury přímo v aplikaci a není tak nutné přecházet do jiné aplikace.

Vzhledem k tomu, že tato aplikace bude sloužit pouze jako interní nástroj a není v plánu ji nabízet třetím stranám, její finanční přínosy není možné jednoduše určit. Po spuštění aplikace do ostrého provozu bude však možné po určité době provést zhodnocení, kolik času nová aplikace šetří oproti stávající aplikaci.

Tabulka č. 7: Výstup SWOT analýzy nové aplikace
(Zdroj: Vlastní zpracování)

Silné stránky	Slabé stránky
• Řešení splňující požadavky společnosti • Jednoduše rozšiřitelné řešení využívající moderní technologie • Přehledné a responzivní uživatelské rozhraní • Zobrazování metrik pro řízení projektů a finanční řízení společnosti • Možnost editace vytvářených výkazů	• Nemožnost volby aktivity při vytváření nového záznamu • Notifikační hlášky nejsou dostatečně výrazné a uživatel je může přehlédnout • Nemožnost vyhledávání v projektech
Příležitosti	**Hrozby**
• Implementace nových technologií	• Nedostupnost Toggl API

Při srovnání výstupu SWOT analýzy nové aplikace s analýzou současného řešení (kapitola 2.10) je možné vidět výrazné zlepšení situace.

ZÁVĚR

Cílem této práce bylo navrhnout a vyvinout frontendovou část webové aplikace pro společnost Proof & Reason, s.r.o., která bude prezentovat data pro řízení projektů, vedení týmů a finanční řízení společnosti, a bude umožňovat přípravy a editace výkazů práce přímo v aplikaci.

V úvodní části práce došlo k identifikaci problému a stanovení cíle. V následující části byla popsána teoretická východiska, ve kterých byly objasněny pojmy a metody, jež byly využívány v dalších částech této práce. Jednalo se především o popis webových technologií a nástrojů pro vývoj webových aplikací.

Další část práce se zabývala analýzou problému a současné situace, byla zde představena společnost Proof & Reason, s.r.o. a byly provedeny nezbytné analýzy před zahájením návrhu a vývoje aplikace. V analýze současného stavu byly zjištěny nedostatky současného řešení, dále následovala analýza uživatelů a infrastruktury současného řešení. Následně zde byly zmapovány požadavky společnosti na novou aplikaci. Nakonec byla provedena analýza konkurenčních aplikací a SWOT analýza současného řešení.

Poslední a zároveň stěžejní část práce se věnovala vlastnímu návrhu řešení na základě zjištěných poznatků z analýzy problému a současné situace. Nejdříve byla představena architektura aplikace a její zabezpečení. Následně bylo navrhnuto a implementováno uživatelské rozhraní na základě požadavků společnosti a popsány hlavní moduly aplikace. Aplikace byla testována v průběhu implementace na přítomnost technických chyb a po dokončení implementace proběhlo i testování použitelnosti uživatelského rozhraní se všemi uživatelskými rolemi. Závěr této části se věnuje budoucímu vývoji aplikace, ekonomickému zhodnocení a přínosům práce.

Vytvořená aplikace řeší zjištěné nedostatky současného řešení a také splňuje požadavky společnosti. Aplikace byla nasazena do testovacího provozu, ve kterém bude probíhat její další testování. Na základě zpětné vazby od uživatelů se bude aplikace dále vyvíjet a rozšiřovat o další potřebné funkcionality tak, aby během několika následujících měsíců mohlo dojít k migraci dat ze současného řešení do nového a aplikace mohla být nasazena do ostrého provozu.

SEZNAM POUŽITÝCH ZDROJŮ

[1] What Is a Web Application? How It Works, Benefits and Examples. *Indeed* [online]. 2021 [cit. 2021-02-20]. Dostupné z: https://www.indeed.com/career-advice/career-development/what-is-web-application

[2] SOSINSKY, Barrie A. *Mistrovství - počítačové sítě: [vše, co potřebujete vědět o správě sítí]*. Brno: Computer Press, 2010. ISBN 978-80-251-3363-7. Dostupné také z: https://ndk.cz/uuid/uuid:f72865a0-f6db-11e7-a97b-005056827e51

[3] Hypertext Transfer Protocol -- HTTP/1.1. *W3* [online]. 1999 [cit. 2021-03-03]. Dostupné z: https://www.w3.org/Protocols/rfc2616/rfc2616.html

[4] What is HTTPS?. *SSL.com* [online]. 2020 [cit. 2021-02-23]. Dostupné z: https://www.ssl.com/faqs/what-is-https/

[5] Application Programming Interface (API). *IBM* [online]. 2020 [cit. 2021-02-25]. Dostupné z: https://www.ibm.com/cloud/learn/api

[6] REST APIs. *IBM* [online]. 2019 [cit. 2021-02-25]. Dostupné z: https://www.ibm.com/cloud/learn/rest-apis

[7] Structuring the web with HTML. *MDN Web Docs* [online]. 2021 [cit. 2021-02-21]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/HTML

[8] Document Object Model (DOM). *MDN Web Docs* [online]. 2021 [cit. 2021-02-24]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model

[9] Learn to style HTML using CSS. *MDN Web Docs* [online]. 2021 [cit. 2021-03-12]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/CSS

[10] All CSS specifications. *W3.org* [online]. 2021 [cit. 2021-03-12]. Dostupné z: https://www.w3.org/Style/CSS/specs.en.html

[11] JavaScript. *MDN Web Docs* [online]. 2021 [cit. 2021-03-16]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/JavaScript

[12] FLANAGAN, David. *JavaScript: the definitive guide : master the world's most-used programming language*. Seventh edition. Beijing: O'Reilly, 2020. ISBN 978-149-1952-023.

[13] Typed JavaScript at Any Scale. *TypeScript* [online]. [cit. 2021-03-17]. Dostupné z: https://www.typescriptlang.org/

[14] React - A JavaScript library for building user interfaces. *ReactJS* [online]. [cit. 2021-03-19]. Dostupné z: https://reactjs.org/

[15] What is ReactJS: Introduction To React and Its Features. *Simplilearn* [online]. 2021 [cit. 2021-03-19]. Dostupné z: https://www.simplilearn.com/tutorials/reactjs-tutorial/what-is-reactjs

[16] Next.js by Vercel - The React Framework. *Next.js* [online]. [cit. 2021-03-22]. Dostupné z: https://nextjs.org/

[17] Material-UI: A popular React UI framework. *Material-UI* [online]. [cit. 2021-03-23]. Dostupné z: https://material-ui.com/

[18] Platform for API Design, Development & Documentation. *Apiary* [online]. [cit. 2021-03-23]. Dostupné z: https://apiary.io/

[19] KOCH, Miloš a Bernard NEUWIRTH. *Datové a funkční modelování*. Vyd. 4., rozš. Brno: Akademické nakladatelství CERM, 2010. ISBN 978-80-214-4125-5.

[20] Use Case Diagram. *Creately* [online]. 2020 [cit. 2021-03-23]. Dostupné z: https://creately.com/blog/diagrams/use-case-diagram-tutorial/

[21] How to Do a SWOT Analysis. *WordStream* [online]. 2017 [cit. 2021-03-24]. Dostupné z: https://www.wordstream.com/blog/ws/2017/12/20/swot-analysis

[22] Strength, Weakness, Opportunity, and Threat (SWOT) Analysis. *Investopedia* [online]. 2021 [cit. 2021-03-24]. Dostupné z: https://www.investopedia.com/terms/s/swot.asp#what-is-swot-analysis

[23] Stratégové digitálních produktů. In: *Proof & Reason* [online]. [cit. 2021-01-20]. Dostupné z: https://proofreason.com/

[24] Time Tracking Software for Any Workflow. *Toggl Track* [online]. [cit. 2021-04-21]. Dostupné z: https://toggl.com/track/

[25] Timesheet & time tracking app. *Costlocker* [online]. [cit. 2021-01-22]. Dostupné z: https://costlocker.com/

[26] Fully Automatic Time Tracking. *Timely* [online]. [cit. 2021-01-22]. Dostupné z: https://memory.ai/timely

[27] Promise based HTTP client for the browser and node.js. *Axios* [online]. [cit. 2021-04-01]. Dostupné z: https://github.com/axios/axios

[28] *ESLint - Pluggable JavaScript linter* [online]. [cit. 2021-04-02]. Dostupné z: https://eslint.org/

[29] Opinionated Code Formatter. *Prettier* [online]. [cit. 2021-04-02]. Dostupné z: https://prettier.io/

[30] *State of JS 2020: Other Tools* [online]. [cit. 2021-04-02]. Dostupné z: https://2020.stateofjs.com/en-US/other-tools/utilities

[31] React Hooks library for data fetching. *SWR* [online]. [cit. 2021-04-03]. Dostupné z: https://swr.vercel.app/

[32] Simple React forms validation. *React Hook Form* [online]. [cit. 2021-04-03]. Dostupné z: https://react-hook-form.com/

[33] A composable charting library built on React components. *Recharts* [online]. [cit. 2021-04-03]. Dostupné z: https://recharts.org/en-US/

SEZNAM POUŽITÝCH ZKRATEK

API	Application Programming Interface
CDN	Content Delivery Network
CSS	Cascading Style Sheets
DOM	Document Object Model
ES6	ECMAScript 6
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
JS	JavaScript
REST	Representational State Transfer
SPA	Single Page Application
SSG	Static Site Generation
TS	TypeScript
UI	User Interface
URL	Uniform Resource Locator

SEZNAM POUŽITÝCH OBRÁZKŮ

SEZNAM POUŽITÝCH TABULEK